Speech to Text to Semantics: A Sequence-to-Sequence System for Spoken Language Understanding

John Dodson

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2020

Reading Committee:

Gina-Anne Levow, Chair

Philip Borawski

Program Authorized to Offer Degree:
Linguistics

University of Washington

Abstract

Speech to Text to Semantics:
A Sequence-to-Sequence System for Spoken Language Understanding

John Dodson

Chair of the Supervisory Committee:
Gina-Anne Levow
Linguistics

Spoken language understanding entails both the automatic transcription of a speech utterance and the identification of one or more semantic concepts being conveyed by the utterance. Traditionally these systems are domain specific and target industries like travel, entertainment, and home automation. As such, many approaches to spoken language understanding solve the task of filling predefined semantic slots, and cannot generalize to identify arbitrary semantic roles.

This thesis addresses the broader question of how to extract predicate-argument frames from a transcribed speech utterance. I describe a sequence-to-sequence system for spoken language understanding through shallow semantic parsing. Built using a modification of the OpenSeq2Seq toolkit, the system is able to perform speech recognition and semantic parsing in a single end-to-end flow. The proposed system is extensible and easy to use, allowing for fast iteration on system parameters and model architectures.

The system is evaluated through two experiments. The first experiment performs a speech to text to semantics transformation and uses n-best language model rescoring to generate the best transcription sequence. The second experiment executes the same transformation process, but generates transcriptions through shallow language model fusion. Both experiments evaluate several combinations of speech recognition models and semantic parsers.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Completion of this thesis would not have been possible without guidance and cooperation from the Big Data Analytics Solutions department at L3Harris Technologies ComCept Division, and the faculty of the UW Linguistics department. I extend my sincere thanks.

Chapter 1
INTRODUCTION

A key facet in building artificially intelligent systems is providing the computational means to understand human language, and enabling a machine to exhibit generalized spoken language understanding (SLU) has been at the forefront of AI research for many decades [24]. Indeed, finding robust techniques to identify, for example, *who* did *what* to *whom* given some speech utterance is a crucial step in facilitating such an understanding. This thesis investigates one such technique through the design and implementation of a neural system for mapping speech to semantically annotated transcriptions.

The system described herein is designed with an end-to-end philosophy, in that it transcribes speech and classifies predicate-argument roles in a single shot without requiring hand-engineered features. It accomplishes this by treating SLU as a multi-stage sequence-to-sequence task and uses encoder-decoder neural network architectures to generate data representations at each stage of the pipeline. The pipeline for this task is comprised of three distinct and configurable modules: a speech recognizer, a language model, and a semantic role labeler.

1.1 Motivation

Spoken language data is notoriously difficult to leverage in higher-level natural language processing (NLP) tasks like classification and tagging; many automatic speech recognition (ASR) systems focus solely on generating word transcriptions, without giving regard to concepts such as punctuation boundaries and syntax. Moreover, data sources are often noisy and unreliable in physical world environments, and ASR tools tend to perform rather poorly as a result. This can be attributed to both the complexity of modeling natural human speech

and the inability for many ASR systems to generalize to out-of-domain speech phenomena ([49],[61]).

Certain semantically-oriented speech tasks like dialogue slot-filling suffer from similar problems. Nonetheless, dialogue systems have demonstrated promising results in extensible SLU across domains like travel, media, and more. For example, Yu, et al. [93] describes a task-oriented state tracking system which is adaptable to new dialogue domains through a combination of rule-based and statistical modeling. The model is evaluated across two question answering domains, namely, restaurant search and tourist information, and is found to be domain extensible. There are several other similar examples of handling generalized and non-task-specific conversational dialogue systems which map speech to text to semantics ([89], [72], [13], [35], among others).

The broader question of how to derive meaningful and consistent semantic information from spoken language remains largely unanswered. While the recent advancements in conversational slot-filling are critically important to dialogue-based tasks, other speech tasks which require higher levels of semantic abstraction might not necessarily benefit from the same advancements [8]. For example, tasks like predicate-argument identification entail more universal semantic roles than a dialogue system operating within the context of travel support [34]. Systems that perform tasks like semantic parsing must be generalizable and robust to the many potential issues of an ASR transcript.

The research presented herein addresses the technical challenges of building and evaluating an end-to-end system for semantically annotating speech transcripts. I describe a cohesive architecture for jointly transcribing speech and identifying full predicate-argument frames within the transcription. I evaluate the system performance by cross-validating with unseen speech corpora. The system is modular and allows for quick experimentation with different types of models, with a particular focus on models within the sequence-to-sequence domain that can be represented through encoder-decoder architectures.

Specifically, this research investigates:

- Joint cross-domain speech recognition and semantic analysis,
- Modular system compositions, and
- Practical training procedures for multi-task systems.

I seek to answer two central questions through this investigation. The first question is whether or not end-to-end systems are suitable for this type of task. End-to-end machine learning has become a prevalent focus for nearly every NLP discipline, and popular models tend to learn direct alignments between input data and corresponding labels without requiring hand-engineered features. Semantic role labeling (SRL) models are traditionally dependent on linguistically-informed features like syntax, and generating those features would require significant processing for data such as speech transcriptions. To mitigate this issue, my system takes inspiration from recent developments in syntax-agnostic SRL, where models are not reliant on discrete linguistic features.

The second question is to understand the levels at which certain model architectures can interoperate to provide salient outputs. Encoder-decoder models learn in a two-step process: firstly, they encode data into a latent vector representation; and secondly, they decode that representation into output symbols. The fidelity of the output symbols is heavily influenced by the type of network used to encode the data. Particular attention is given to output samples of each model architecture used in the experimentation.

1.2 Preliminary Concepts

This section provides a review of fundamental concepts in spoken language processing and computational semantics. The experimentation described in subsequent chapters of this thesis is inspired by recent work in neural speech processing and semantic classification, and the remainder of this chapter begins to lay the necessary foundation for deeper understanding of both disciplines.

1.2.1 Automatic Speech Recognition

Speech recognition is the process of transcribing waveform data into discrete sequences of words. This process traditionally utilizes two primary components: an acoustic model for aligning spectral features to grapheme sequences, and a language model for aligning grapheme sequences with word transcriptions. In the simplest case, speech decoding can be mathematically formalized:

$$\hat{W} = \arg\max_{W \in D} P(W|O) \tag{1.1}$$

where $W = w_1, w_2, ..., w_n$ is a sequence of words for all $w_i \in D$, and $O = o_1, o_2, ..., o_n$ is a set of acoustic observations. Using Bayes' Rule, equation $P(W|O)$ may be computed as follows:

$$P(W|O) = \frac{P(O|W)P(W)}{P(O)} \tag{1.2}$$

Equation 1.2 effectively defines the usage of both the acoustic and language models. During the training process, the two models are tuned to accurately estimate $P(O|W)$ and $P(W)$, respectively.

Acoustic Modeling

An acoustic model is a collection of statistics about the sounds which make up a particular word. The most widely adopted methodology for acoustic modeling is through a combination of Gaussian mixture models (GMMs) and hidden Markov models (HMMs) [29]. GMMs are immensely useful for modeling complex data distributions, and with enough data they can model probabilities with high accuracy using the Expectation-Maximization (EM) algorithm [92].

For speech data, however, the GMM alone is insufficient; GMMs have no concept of temporality and are thus unable to model sequence information. The GMM-HMM paradigm mitigates this shortcoming with the addition of the HMM component, which provides the

ability to model the sequential nature of speech. An HMM models sequences of states such that for some state $s_t \in S$, the probability of symbol $o_k \in O$ occurring at step t is $b_i(k) = P(X_t = o_k | s_t = i)$, where X_t is an observable event sequence. For GMM-HMM acoustic models, each state transition represents a phoneme or grapheme which helps constitute the pronunciation of a word. State emissions contain a GMM probability density function:

$$b_j(\mathbf{x}) = \sum_{k=1}^{M} c_{jk} N(\mathbf{x}, \mu_{jk}, \Sigma_{jk}) \tag{1.3}$$

where c_{jk} is a component weight value, and $N(\mathbf{x}, \mu_{jk}, \Sigma_{jk})$ is a Gaussian density function with observation vector $\mathbf{x}$, mean vector μ_{jk}, and covariance matrix Σ_{jk}.

The GMM-HMM model is particularly important because an observation sequence comprised of acoustic data is not part of a discrete space [46]. In outputting GMM probability density functions with every state emission, the model is able to seamlessly map the observation data from a continuous space into a discrete space. In the case of speech recognition, this means training a GMM-HMM model to map spectral feature vectors to sequences of labels. The model would estimate $b_j(\mathbf{x})$ for each frame of the input signal.

The downside to the GMM-HMM approach is that training a good model is somewhat inefficient when the data is on a non-linear manifold in vector space. Research has shown that human speech data indeed lies on a lower dimensional manifold embedded within a higher dimensional feature space [28]. Similarly, GMM-HMM models are unable to process large chunks of contextual information, potentially missing long-distance dependencies which could benefit the target output sequence. Deep learning is one mitigation technique for these shortcomings.

Deep learning has taken ASR research by storm; effectively all modern speech systems use some type of neural network at the acoustic level as an alternative to GMMs, since they can both model nonlinear data and capture contextual information during training. Hybrid systems, such as the ones described by Li et al. [58] and Dahl, et al. [22], use DNN-HMM acoustic models to learn frame-level alignments between acoustic features and grapheme

sequences. While neural networks are inherently better classifiers than GMMs, the hybrid systems nonetheless suffer from inefficiently complex training mechanisms and the inability to support robust large-vocabulary speech recognition [37].

End-to-end speech recognition is an area of research which seeks to solve the shortcomings of hybrid systems and simplify the entire ASR pipeline [5]. End-to-end models accomplish this by providing architectures with are more cohesive and flexible. For example, the model described by Graves & Jaitly [37] is a recurrent neural network (RNN) encoder which maps spectral feature inputs to latent vector representations, and then uses a separate neural network model to decode the latent vectors as grapheme sequences. Because of the separability of the encoder and decoder, the model can map a variable-length input sequence to a variable-length output sequence. RNN models iterate over equations 1.4 and 1.5, which are adapted from Graves & Jaitly [37]:

$$h_t = g(\mathbf{W}_{ih}x_t + \mathbf{W}_{hh}h_{t-1} + b_h) \tag{1.4}$$

$$y_t = \mathbf{W}_{ho}h_t + b_o \tag{1.5}$$

where input vector x is a sequence of input features, h_t denotes the latent vector output at timestep t, and $\mathbf{W}$ is a trainable weight matrix. g denotes the specific RNN function, which can be anything from a Long Short-Term Memory (LSTM) to a Gated Recurrent Unit (GRU) ([44], [16]). Utilizing multiple layers of RNNs has been shown to outperform hybrid systems ([37], [4]).

Language Modeling

The language model component of an ASR system estimates the conditional probability of a token given the set of previous tokens:

$$P(W) = \prod_{i=1}^{n} P(w_i|w_1, w_2, ...w_{i-1}) \tag{1.6}$$

When the size of vocabulary D is large, equation 1.6 becomes computationally expensive to calculate. Practical applications of language modeling often apply an equivalence function $\Phi(W_{k-1})$ which bounds the context of the feature space to $W_{k-1} = w_1, w_2, ..., w_{k-1}$. This type of context constraint makes the language modeling problem much simpler to compute. For example, a language model using a trigram equivalence function estimates the probability of a token given only the last $n-2$ tokens:

$$P(W) = \prod_{i=1}^{n} P(w_i | w_{i-2}, w_{i-1}) \tag{1.7}$$

Setting higher values for n allows the model to consider more context. Estimating equation 1.7 is indeed more practical than equation 1.6, but models are still significantly limited in their ability to generalize to previously unseen tokens. Neural language models seek to mitigate the generalization issue by jointly learning both a probability function for the language model and a feature vector embedding for each word in the vocabulary, such that contextually similar words are mapped closer together in vector space [6]. The experiment in Bengio, et al. [6] is shown to support out-of-vocabulary words by predicting a potential feature vector based on the context of the unseen words.

1.2.2 Semantic Role Labeling

Semantic classification is the task of identifying semantic relationships among constituents in a sentence [34]. SRL is a shallow parsing technique and sequence labeling task, where a token string is aligned with corresponding labels indicating the semantic role of each token. Semantic roles traditionally encompass various predicate types and argument types.

(1) Variations of semantic alignments for the predicate 'gave'

a. Bob gave Alice a gift .

[**giver** Bob] gave [**receiver** Alice] [**entity-given** a gift] .

b. Bob gave a great tour of the building .

[**giver** Bob] gave [**entity-given** a great tour of the building] .

c. The senator gave a great speech to last night's crowd .

[**giver** The senator] gave [**entity-given** a great speech] to [**receiver** last night's crowd] .

SRL tasks can be subcategorized into predicate identification, argument identification, or a combination of the two. The seminal work by Gildea & Jurafsky [34] focused on argument identification by chaining together multiple classifiers. Some modern approaches provide a unified model to jointly predict predicates and arguments ([42], [11]). This thesis focuses on the joint task of full frame predicate-argument identification.

Semantic Roles

In the examples provided by (1), the semantic roles of `giver`, `receiver`,and `entity-given` annotate the context sentences, filling the expected roles of the predicate `gave`. The collection of a predicate and its roles is called a semantic frame.

Identifying the actors which constitute a semantic frame is typically a two-step process: firstly, the model identifies the context predicate; and secondly, the model infers the arguments associated with that predicate. This process is aligned with Fillmore [31], who proposes that multiple types of roles exist which are specific to their corresponding predicate. That kind of verbal analysis assumes that any given frame has argument roles which may not be present for a different frame.

Proper predicate-argument characterization is a pervasive issue for computational semantics in general. Specific roles like **giver** and **receiver** are only applicable in certain contexts, and training a classifier to accurately align every possible role is implausible. PropBank labeling [62] is a common semantic annotation convention which specifies a more widely applicable label set. Argument labels are represented through an incremental numbering schema, namely, **Arg-0**, **Arg-1**, and so forth. Using PropBank styling, predicate usages are grouped with associated argument types in framesets, which specify both the syntactic usage of the predicate as well as the associated arguments:

(2) Frameset excerpts for predicate 'gave'

a. Frameset give.01, 'transfer'

Arg-0: giver

Arg-1: thing given

Arg-2: entity given to

Ex: [**Arg-0** Bob] gave [**Arg-2** Alice] [**Arg-1** a gift] .

b. Frameset give.17, 'yield'

Arg-0: thing yielding

Arg-1: to what

Ex: [**Arg-0** Bob] will eventually give in to [**Arg-1** the pressure of his work] .

The examples provided by (2) are adapted from the PropBank frame file repository. Palmer, et al. [62] note that the argument naming convention is extensible to an any number of predicates; the exact mapping from argument label to semantic role is defined per predicate, but a classifier would only need to identify the high-level label. This dramatically decreases the size of the label space, making the task much more tenable.

Machine Learning for SRL

Some of the earliest successful attempts at SRL involved machine learning techniques, including classical methods like maximum entropy [7], support vector machines [10], and log-linear models [39]. SRL has also become a common application for neural network architectures. Collobert, et al. [21] provides an early architecture comprised of n-gram word selections, word vectors, and feed-forward computation for successful (best reported $F_1 = 74.15$) semantic parsing. In recent years, RNNs and convolution neural networks (CNNs) have also been successfully applied to SRL [84].

Traditional SRL models use syntactic features to inform a downstream classification model [90]. Several neural approaches to SRL encode syntactic labels as part of a feature vector ([88], [51], among others). For example, a model might represent words as concate-

nated vectors $\mathbf{w} = \mathbf{w}^e \oplus \mathbf{w}^{pos}$, where $\mathbf{w}^e \in \mathbb{R}^{d_w}$ are word embeddings and $\mathbf{w}^{pos} \in \mathbb{R}^{d_{pos}}$ are part of speech embeddings. The additional context provided by $\mathbf{w}^{pos}$ helps the model learn that words with certain parts of speech are frequently associated with certain semantic roles. He, et al. [41] defines the SRL task as a prediction of the best label sequence $\hat{\mathbf{y}} \in \Upsilon$ associated with the embedding vector:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \Upsilon} f(\mathbf{w}, \mathbf{y}) \tag{1.8}$$

where $f(\mathbf{w}, \mathbf{y})$ computes a probability distribution over the label set.

The specific syntactic features vary depending on the model architecture, and no set standard exists. In the simplest case, generating the alignments in (1), for example, can be done by estimating $P(l|\theta)$, where l is the semantic role and θ is a set of parameters. In Gildea & Jurafsky [34], the parameter set includes features such as the constituent head word, phrase type, and parse tree paths because the authors claim that such features help to characterize the syntax-semantics relationship between predicate and argument. Alternatively, Kasai, et al. [51] presents a state-of-the-art SRL model using tokens, parts of speech, lemmas, and syntactic features derived from dependency parses. Some models, such as the one described in Strubell, et al. [82], are trained to jointly predict predicates, arguments, and syntactic labels.

1.3 Roadmap

Chapter 2 is a deep dive into two sets of encoder-decoder neural networks: three ASR models and two SRL models. Focus is placed on evaluating different variations of encoder-decoder architectures and how they function as end-to-end solutions for their respective tasks. After reviewing each model, I introduce a different approach to semantically annotating speech transcripts [79], which I use as a baseline system.

Chapter 3 begins with an introduction to the OpenSeq2Seq toolkit [54] for sequence-

to-sequence experimentation[1]. Then, I discuss the primary contributions of this thesis: an augmented OpenSeq2Seq pipeline for bootstrapping an arbitrary number of postprocessing models for tasks like language model rescoring and semantic tagging. I also describe a novel implementation of the popular Listen, Attend and Spell model to be compatible with the toolkit. The chapter concludes with a discussion on the system compositions which are used to evaluate the speech to text to semantics task, including the concepts of data augmentation and model fusion for establishing an end-to-end flow.

Chapter 4 provides an experimental analysis of the system's performance. I briefly overview the training environment hardware and review the comprehensive training procedures for both the ASR and SRL model sets. I then describe the implementation of the baseline system and present all experiment results. The chapter concludes with an analysis of the results and potential error mitigation techniques.

Chapter 5 concludes the thesis by reviewing the major contributions and experimentation results. The primary conclusions are that (i) end-to-end approaches to the ASR-SRL task are viable, and spoken language data can indeed be annotated with semantic roles without requiring intermediate feature extraction; and (ii) more sophisticated system structures to incorporate techniques like joint multi-task training or deep fusion could further improve the state of the art.

[1]The base toolkit is freely available at https://github.com/NVIDIA/OpenSeq2Seq

Chapter 2
MODEL SURVEY

The previous chapter overviewed the foundations of speech processing and shallow semantic parsing. With the necessary groundwork in place, this chapter surveys recent literature in encoder-decoder models for the ASR and SRL tasks. The two foci of the chapter are model descriptions and prior research efforts in semantically parsing speech transcriptions. I focus the model discussions entirely on encoder-decoder architectures, providing a description of each model and how they function as end-to-end solutions for their respective tasks. The final section of this chapter introduces work from Shrestha, et al. [79], which I use as a baseline system.

2.1 End-to-End ASR

For the speech recognition module, three encoder-decoder variations are considered: DeepSpeech2, Jasper, and Listen Attend and Spell. This section provides a deep dive into the architectures, training procedures, and nuances of each model.

2.1.1 DeepSpeech2

DeepSpeech2 is an end-to-end encoder-decoder model from Baidu Research [2], and one of the first successful ASR systems built entirely with a neural network. The model learns a direct alignment between spectrogram features and output graphemes. The network accepts a time-windowed vector of audio features $x_t^{(i)}, t = 0, ..., T^{(i)} - 1$ as input and learns an output distribution $y_{(i)}$ corresponding to the input utterance. The label distribution at each time step is $p(l_t|x)$, where l_t is an entry in the lexicon of the target language. The model uses a GPU-enabled Connectionist Temporal Classification (CTC) loss function [36] to learn the

character-level alignment.

DeepSpeech2 contributed several major breakthroughs in ASR research. Firstly, the authors note that the model is extensible to multiple languages, with successful experiments on both English and Chinese corpora. Secondly, the model has become a mainstream ASR commodity; training a sufficient speech-to-text model requires no expertise in the target language and several implementations are available for further experimentation or production use. Finally, and perhaps most importantly, the DeepSpeech2 authors contribute evidence that useful end-to-end models need substantial compute capacity to support large-scale distributed training, as well as large training corpora. The authors note that the English DeepSpeech2 model had a total of 11,940 hours of labeled speech and 8 million utterance samples in total. It trains over 20 epochs.

The reference implementation of DeepSpeech2 is very large, with 11 layers in the network: 3 layers of 2D CNNs, 7 bidirectional RNNs, and a single fully-connected layer to output the grapheme probability distribution. Figure 2.1 illustrates the general architecture of a DeepSpeech2 model Throughout each layer, batch normalization [48] is used as a regularization technique and accelerate training times.

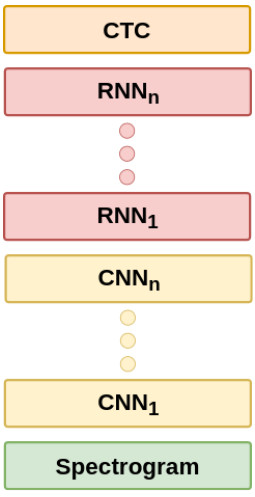

Figure 2.1: DeepSpeech2 architecture. The spectrogram features are given as input to a hybrid CNN-RNN encoder, and decoded into character outputs through a CTC layer.

The DeepSpeech2 encoder is a hybrid CNN-RNN architecture. The CNNs employ 2D temporal convolutions to model time and frequency information, and feed the convolved vector into a unidirectional multi-layer RNN. According to the DeepSpeech2 authors, the unidirectional RNNs are placed on top of a row convolution operation, which can make predictions at a given timestep without requiring the entire input sample. The decoder is a feedforward network into the CTC computation.

2.1.2 Jasper and 1D Convolutions

The Jasper (Just Another Speech Recognizer) model is a recent end-to-end CNN-based model [57] which achieves a new state of the art WER for the LibriSpeech corpus [63]. Jasper is unique in many ways: not only is the model significantly larger than DeepSpeech2, with 54 total layers in the most accurate model, but it is also reported to be much more efficient to train, supports real-time audio decoding, and has implementations that run on hardware with low size, weight, and power. Jasper's efficiency is attributed to the components within its architecture; the authors focused the architecture on operations suitable for distributed GPU computation. The core operation within Jasper is the 1D convolution. Unlike 2D convolution operations, which are largely popular for image processing since imagery data have patterns across 2 dimensions (i.e., height and width), 1D convolutions model sequential data that have a single spatial dimension. 1D convolutions can efficiently model time series data much like RNNs. For example, given an input vector v and kernel g, the 1D convolution operation is defined as:

$$(v * g)(i) = \sum_{j=1}^{m} g(j) \cdot v(i - j + \frac{m}{2}) \tag{2.1}$$

where m is the length of g (adapted from [94]). Intuitively, this operation slides the kernel across the input vector and applies an element-wise dot product between f and g.

1D convolutions are nearly identical to RNN operations, with some notable differences.

A convolution can only model sequential data within a finite receptive field, dictated by the size of the kernel. This often makes CNN-based models undesirable for both variable-length sequence-to-sequence tasks and modeling long-distance dependencies. While RNNs are the most common solution to sequence modeling, they suffer from phenomena like the vanishing gradient problem and are difficult to train for large models [65]. One of the most beneficial properties of convolution operations is that they can be easily parallelized and generally train faster than their RNN counterparts [9].

Jasper uses 1D convolutions in a block architecture; each layer is encapsulated in a block which contains several sub-block operations. Each block has a different kernel size and thus models different features at each level. Architectural variations of Jasper are denoted as BxR, where B is the number of blocks and R the number of sub-blocks. An illustration of the Jasper architecture is provided in Figure 2.2, adapted from the authors' architecture diagram.

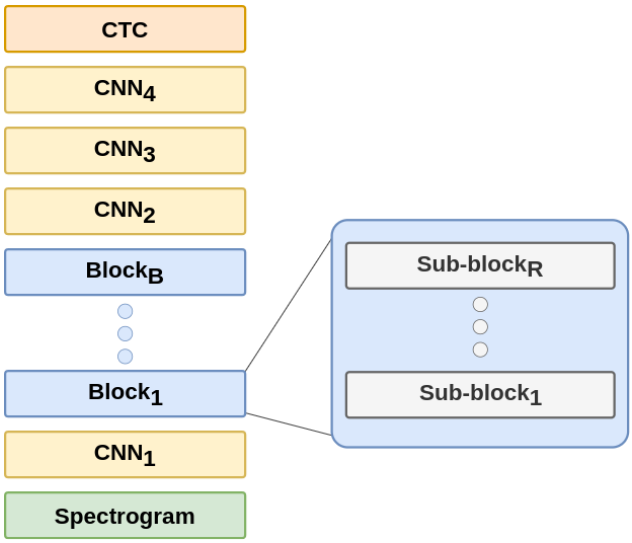

Figure 2.2: Jasper BxR architecture. Every sub-block contains a series of Conv1D + BatchNorm + ReLU + Dropout operations, repeated R times. Similarly to DeepSpeech2, Jasper uses a CTC decoder to output grapheme tokens.

Similarly to DeepSpeech2, the Jasper model requires significant computational resources

to adequately train. The authors note that Jasper's architecture helps facilitate efficient GPU utilization, such that each block can allocate its sub-block operations to a single GPU. The reference implementation of Jasper10x5 reports using 8 GPUs for 400 epochs with the LibriSpeech corpus. Jasper also uses the CTC loss computation to generate character alignments.

2.1.3 Listen, Attend & Spell

Listen Attend and Spell (LAS) is an encoder-decoder architecture inspired by sequence-to-sequence models with attention mechanisms [12]. LAS uses bidirectional RNNs for both the encoder and decoder components, and the latent encoded vector undergoes an attention operation which maps the variable-length input sequence to a fixed-length representation. The decoder aligns the latent vector with an output sequence of graphemes. This type of architectural paradigm has been previously successful in tasks like machine translation [17] and question-answering [91], where the input and output vectors often have different lengths.

The LAS architecture is presented in Figure 2.3. The two primary modules are the Listener, a bidirectional RNN encoder, and the Speller, an attention-equipped RNN decoder. LAS approaches end-to-end ASR with a fundamentally different philosophy than DeepSpeech2 or Jasper; while those two models (and other CTC-based models) assume that output tokens are conditionally independent of each other, LAS learns an implicit language model within the decoder, and is thus able to generate salient transcriptions without the need for CTC. Specifically, the model learns a probability distribution for an output character y_i given the previous tokens and the input vector $\mathbf{x}$:

$$P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|y_{i-1}, ..., y_0, \mathbf{x}) \tag{2.2}$$

The equation in 2.2 is given by the $CharacterDistribution$ function, which is parameterized by an attention vector $\mathbf{c}$ and RNN state s_i, such that at every timestep, $P(y_i|y_{<i}, \mathbf{x}) = CharacterDistribution(s_i, \mathbf{c})$.

LAS is able to model variable-length input sequences by leveraging its attention mechanism. The authors note that a direct mapping from input sequence to latent vector results in slow convergence and poor results. The proposed solution to this problem is the pyramidal-RNN encoder, where each layer of the RNN provides a reduction in the time dimension by a factor of 2. The authors claim that this allows the model to learn a salient latent vector from a smaller input representation.

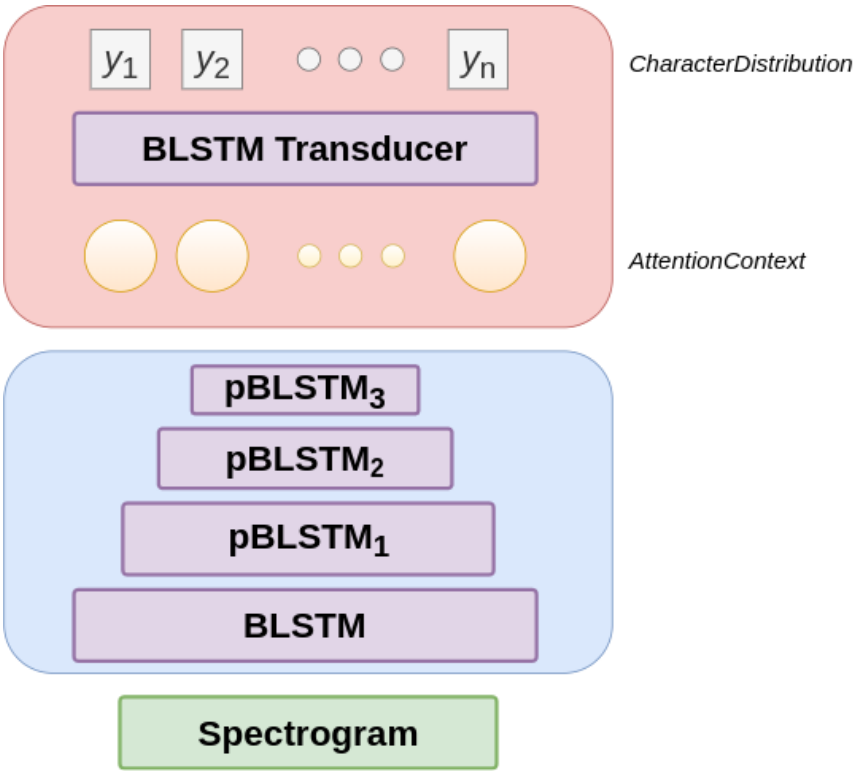

Figure 2.3: Listen, Attend and Spell architecture. The encoder contains one BLSTM and a stack of pyramidal-BLSTM layers. The decoder uses the output of *Listen* and generates an attention vector **c** from the *AttentionContext* function. The BLSTM transducer feeds into *CharacterDistribution* to generate grapheme probabilities at each timestep.

The reference implementation of LAS uses a 3-layer pyramidal-RNN encoder and a 2-layer RNN decoder. The authors use a 2000-hour dataset and report a training time of two weeks. The standard LAS model does not outperform its baseline (best reported WER on clean

speech: 10.3), but an improved architecture presented by Chiu, et al. [15] is reported to be competitive with state-of-the-art ASR models. Improvements include multi-headed attention and various optimization refinements such as asynchronous training and label smoothing.

2.2 Syntax-agnostic SRL

For the semantic role labeling module, Google's frame-semantics parser SLING and an implementation of the model described in Cai, et al. [11], both syntax-agnostic, are considered.

2.2.1 SLING

SLING is an end-to-end system for dynamically producing semantic parses without intermediate syntactic knowledge [73]. The model is a transition-based parser, where the output of the decoder is a sequence of $(state, decision)$ tuples. The SLING system constructs a frame graph to identify semantic frames within a sentence, using its encoder-decoder model to predict the most likely semantic connections. SLING can also be trained to jointly predict named entities.

SLING provides semantic parses using only lexical features from the input sequence. The encoder is a bidirectional LSTM which accepts word, affix, and shape embeddings as features. The decoder is a multi-layered stack of Transition Based Recurrent Units (TBRU), which dynamically update an internal state vector to describe the current state of the frame graph. TBRUs were introduced as the core component of the DRAGNN model [53] which used them to predict intermediate parse tree structures for the dependency parsing task. SLING can therefore be considered a type of semantic analogue to DRAGNN. Other systems, like DeepCx, take inspiration from the transition-based approaches to parsing [26]. The SLING architecture is presented in Figure 2.4.

The model supports a special type of attention mechanism called the attention buffer, which represents stateful information from previous outputs. The authors note the significance of this custom mechanism as it allows SLING to continually refine and re-prioritize existing frames based on the output of its TBRU layer.

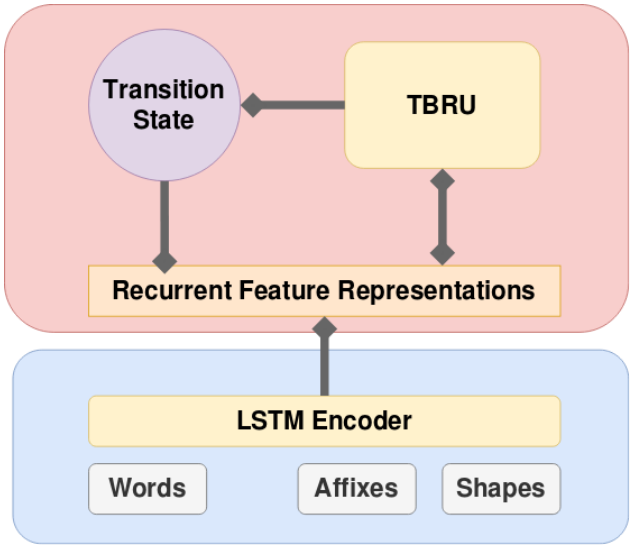

Figure 2.4: SLING architecture. The model maintains an internal transition state to keep track of the frame parses. This state continually informs the recurrent features fed into the TBRU layer.

The SLING system maintains an internal semantic ontology that defines roles, predicates, and named entity types. The model uses this ontology during training. For example, a SLING parser trained to output Propbank annotations would have an ontological definition for each predicate frameset. The predicate *gave* might have entries for **/pb/give.01** and **/pb/give.17**, among others. Argument roles would be denoted as **/pb/arg0**, **/pb/arg1**, and so on. A number of transition types are supported in SLING. Transitions like **EVOKE** and **CONNECT** are parameterized functions expecting type definitions within the SLING ontology, and the system uses the outputs of these functions to compose its semantic graph. The examples in (3) are adapted from Ringgaard, et al. [73] and illustrate the SLING output progression.

Each transition step incrementally builds semantic frames from the sentence. In (3a), the first **EVOKE** establishes the token *Bob* as an entity of type **/saft/person**. The second **EVOKE** establishes *gave* as a Propbank predicate, and **CONNECT** provides a semantic linking between *Bob* and *gave*.

(3) SLING output examples

a. Bob ate the pizza.

EVOKE(/saft/person,1)
SHIFT
EVOKE(/pb/ate, 1)
CONNECT(0, /pb/arg0, 1)
SHIFT
SHIFT
EVOKE(/saft/consumer_good, 1)
CONNECT(0, /pb/arg1, 1)
SHIFT
STOP

b. Bob gave Alice a pizza.

EVOKE(/saft/person,1)
SHIFT
EVOKE(/pb/ate, 1)
CONNECT(0, /pb/arg0, 1)
SHIFT
EVOKE(/saft/person,1)
CONNECT(0, /pb/arg2, 1)
SHIFT
SHIFT
EVOKE(/saft/consumer_good, 1)
CONNECT(0, /pb/arg1, 1)
SHIFT
STOP

The SLING API comes with utilities for interactive parsing, retraining on new data, and defining new ontologies.

2.2.2 Biaffine Syntax-agnostic SRL

The research in Cai, et al. [11] introduces a fully syntax-agnostic semantic role labeler (hereafter BSAT, the biaffine syntax-agnostic tagger) which outperforms state of the art syntax-aware models. Their model uses a bidirectional LSTM encoder to generate two distinct latent vectors: one for predicate representation and one for potential argument representations. The decoder uses biaffine attention [25] to score the most likely argument associations in each predicate-argument word pairing.

The model performs joint predicate-argument identification by producing two vector representations for each potential pairing:

$$\mathbf{h}_i^{(pred)} = ReLU(\mathbf{W}^{(pred)} g_i + \mathbf{b}^{(pred)}) \tag{2.3}$$

$$\mathbf{h}_i^{(arg)} = ReLU(\mathbf{W}^{(arg)} g_i + \mathbf{b}^{(arg)}) \tag{2.4}$$

where $g_i = g_i^f \oplus g_i^b$ is the concatenation of the forward and backward LSTM states at the current timestep. The authors note that the ReLU operation provides the necessary nonlinearity to model the potentially ambiguous features for each role in that particular context. Typical SRL models have external syntactic knowledge to categorize predicates and arguments, but the syntax-agnostic approach must rely on implicit mappings to differentiate which tokens belong in which semantic roles. Such relationships are potentially nonlinear.

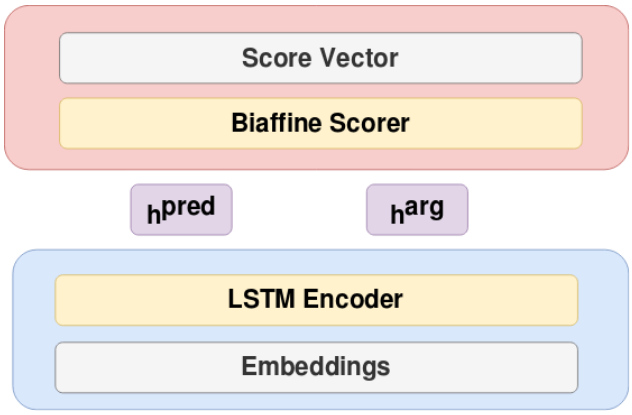

Figure 2.5: BSAT architecture. The encoder produces two distinct latent representations given as input to the biaffine scorer mechanism. The output of the model is a score vector indicating predicate-argument likelihood pairings.

The model architecture is presented in Figure 2.5. As input, the LSTM encoder expects a rich feature vector for each token. For every token in the input sequence, the model generates an embedding vector $e = e^{(r)} \oplus e^{(p)} \oplus e^{(l)} \oplus e^{(pos)} \oplus e^{(i)}$, which captures randomly initialized embeddings $e^{(r)}$, pretrained word embeddings $e^{(p)}$, part of speech embeddings $e^{(pos)}$, lemma embeddings $e^{(l)}$, and predicate indication embeddings $e^{(i)}$. The decoder is inspired by Dozat

& Manning [25] and Luong, et al. [59], which use a type of attention mechanism to model uneven class distribution likelihoods $P(y_i = c|x_i)$.

Class sparsity is a pervasive problem in many language tagging tasks. Tasks like named entity recognition (NER) and slot filling suffer from a very uneven label distribution. Classical probabilistic models like conditional random fields have been successfully leveraged in end-to-end neural models and are shown to improve metrics when the class distribution is uneven ([32], [55]). Rather than rely on a classical model to improve their system, Cai et al. [11] uses the biaffine attention scorer to generate predicate-argument likelihoods in one pass. Specifically, the mechansim outputs a score vector and selects the label with the highest score as its prediction. The authors describe the score computation as:

$$\mathbf{s}_{ij} = \mathbf{h}_i^{\intercal(arg)} \mathbf{W}^{(role)} \mathbf{h}_j^{(pred)} + \mathbf{U}^{(role)} (\mathbf{h}_i^{(arg)} \oplus \mathbf{h}_j^{(pred)}) + \mathbf{b}^{(role)} \tag{2.5}$$

where $\mathbf{W}^{(role)}$, $\mathbf{U}^{(role)}$, and $\mathbf{b}^{(role)}$ are learnable weights. The reference model is trained on the CoNLL-2009 [40] dataset in English and Chinese, with a best reported F1 score of 89.6.

2.3 Previous Work in ASR-SRL

SRL for ASR is still a nascent problem. There has been little substantial progress in semantic role labeling for automatic speech recognition outputs, especially compared to slot-filling or dialogue modeling systems. This lack of progress can be attributed to several important factors: as Favre, et al. [30] observes, speech recognition transcripts often lack important linguistic features needed to adequately predict semantic phenomena, including sentence boundaries, punctuation, and even properly formed words. If an ASR system produces low-fidelity outputs, one can expect similarly low-fidelity semantic annotations as a result.

Fortunately, the word error rate (WER) of a transcription sequence can be minimized given a properly trained ASR system. The problem of sentence boundaries in SRL has been previously explored in Shrestha, et al. [79], in which the authors propose an SRL system classifying *segments* rather than sentences. The segment approach is analogous to

a sliding window over a speech transcript, where a given predicate might overlap several windows. Eventually, when a transcription is fully windowed, the SRL system would have classified the predicate and all of its associated arguments in separate segments. This type of incremental frame building is similar in theory to SLING's process.

The solution presented in Shrestha, et al. [79] uses a windowing algorithm to generate new predicate segments in order to minimize the amount of times the SRL model is used. In their two-pass system, the authors first identify a predicate by using the SRL model to classify the semantics in the current segment. Once a predicate is detected, their solution identifies all segments in which the predicate occurs and merges them according to a predefined heuristic, and again applies the SRL model to detect the associated arguments for the context predicate within the merged segment. The system uses a *take-shortest* or *take-longest* heuristic to create the final segment Table 2.1 illustrates the algorithm with a window size of 3.

Iteration	Window State
1	**IN SEPTEMBER THE** COMPANY RECEIVED NINETY SIX MILLION
2	IN **SEPTEMBER THE COMPANY** RECEIVED NINETY SIX MILLION
3	IN SEPTEMBER **THE COMPANY RECEIVED** NINETY SIX MILLION
4	IN SEPTEMBER THE **COMPANY RECEIVED NINETY** SIX MILLION
5	IN SEPTEMBER THE COMPANY **RECEIVED NINETY SIX** MILLION
6	IN SEPTEMBER THE COMPANY RECEIVED **NINETY SIX MILLION**
Segment	**THE COMPANY RECEIVED NINETY SIX**

Table 2.1: Example iterations of the SRL segmenting algorithm with a window size of 3

As shown in Table 2.1, the final segment created for the predicate `RECEIVED` contains only a portion of the full transcription sequence. The segment captures the first argument for the predicate, `COMPANY` , but fails to merge with the object `MILLION` due to its small window size. Larger window sizes lets the algorithm consider more context and identify more argument

roles.

Favre, et al. [30] proposes a series of tools for evaluating the semantic annotation of speech recognition transcripts in the same way one would evaluate dependency parses. They provide an evaluation using several off-the-shelf SRL tools to label a portion of the Ontonotes v3 corpus [45].

Model	P	R	F1
baseline	**0.2646**	**0.1865**	**0.2188**
win-5-L	**0.2452**	**0.1825**	**0.2093**
win-8-L	0.2344	0.1775	0.2020
win-10-L	0.2266	0.1724	0.1958
win-13-L	0.2176	0.1659	0.1883
win-15-L	0.2139	0.1636	0.1854
win-17-L	0.2096	0.1603	0.1817
win-20-L	0.2062	0.1593	0.1797
win-23-L	0.2045	0.1565	0.1773
win-25-L	0.2024	0.1547	0.1754
win-28-L	0.2023	0.1544	0.1751
win-30-L	0.2008	0.1532	0.1738

Table 2.2: Full frame predicate-argument metrics for the baseline ASR-SRL system. The baseline system from Favre, et al. [30] outperforms the best model from Shrestha, et al. [79]

Shrestha, et al. [79] uses Favre, et al. [30] as a baseline system and fails to surpass it in full frame predicate-argument identification. Table 2.2 illustrates these results.

2.4 Summary

This chapter introduced a series of model architectures for the ASR and SRL tasks. DeepSpeech2, Jasper, and LAS were described in detail as end-to-end ASR solutions. Each of

the three speech recognition systems adhere to the encoder-decoder paradigm, but leverage vastly different architectures: DeepSpeech2 uses a hybrid CNN-RNN encoder mechanism to extract spectral features and encode sequence information into a latent representation. Jasper exploits the 1D convolution operation in a CNN-based encoder, which functions identically to RNN operations but can be easily distributed and is faster to train. LAS exclusively uses RNN layers in both the encoder and decoder.

For SRL, this chapter reviewed SLING and the BSAT model created by Cai, et al. [11]. SLING uses a TBRU to output transition prediction updates to its internal semantic graph, and then uses the internal graph to parse semantic frame information from a raw sentence. BSAT is unique in that it encodes the source data to two distinct latent representations describing the predicate token and all potential predicate-argument pairs. The two latent vectors are decoded using a biaffine scoring function to produce a probability distribution for each predicate-argument pairing.

Finally, this chapter reviewed work from Shrestha, et al. [79], which describes an alternative system for producing semantic annotations over speech transcript data.

Chapter 3

SPEECH TO TEXT TO SEMANTICS: AN END-TO-END APPROACH

The previous chapter introduced several deep learning models for automatic speech recognition and semantic role labeling. This chapter describes a system for applying those models to the end-to-end speech tagging task with the OpenSeq2Seq toolkit. Specifically, the chapter objectives are to explain:

1. OpenSeq2Seq postprocessor extension: OpenSeq2Seq is designed to be extensible such that different modules can be easily added to the toolkit. I describe an extensible module for adding arbitrary postprocessors to derive additional features from the outputs of the primary encoder-decoder model.

2. OpenSeq2Seq LAS extension: The toolkit contains several implementations of popular ASR models, including DeepSpeech2 and Jasper. I describe an OpenSeq2Seq implementation for a LAS-like model with a more complicated attention mechanism inspired by recent research in end-to-end ASR.

3. Model compositions for single-shot speech tagging: The compositions of each model in the system are more sophisticated than a simple ordered pipeline of processing steps. From data augmentation to shallow model fusion, I discuss how to tie each of the various components together to form a cohesive end-to-end flow.

The motivation for evaluating different arrangements of ASR-SRL models is to better characterize different facets of the speech to text to semantics task. The major contributions of this thesis are described in the following sections of this chapter.

3.1 Extending OpenSeq2Seq

The system I describe in this section is a an extension of OpenSeq2Seq [54], a neural network toolkit designed for rapid experimentation with sequence-to-sequence models. The toolkit is built with Tensorflow [1], which expresses mathematical models as computation graphs; it symbolically represents machine learning operations as vertices and the communication between operations as edges. While these graphs can be used to express generic mathematical computations, in the case of deep learning the inbound and outbound data are tensors which can capture stateful information about the model (i.e., weight matrices). Tools like Tensorflow allow for the construction of sophisticated graph topologies to control the ordering of operation executions [56]. Other deep learning libraries like Pytorch [66] and MXNet [14] use computation graphs to represent neural network models.

OpenSeq2Seq expands upon Tensorflow's expressiveness and provides logical abstractions for complex deep learning concepts; specifically, the notions of data provisioning, encoder subgraphs, and decoder subgraphs are abstracted such that each of the three components can be architected individually but utilized together as a global computation graph. With OpenSeq2Seq, users can create encoder-decoder models for machine translation, speech recognition, speech synthesis, and more [54]. The OpenSeq2Seq authors provide a huge toolbox of reusable components for crafting encoder-decoder models. For example, there are out-of-the-box encoder architectures for Transformer models [86], time-delay neural network (TDNN) models [87], and many more. Additionally, OpenSeq2Seq can fully exploit Tensorflow's computation graph architecture and support distributed training across multiple GPUs. Figure 3.1 depicts the flow of an OpenSeq2Seq-based system.

The authors designed the toolkit to be modular and high-level. Architecting a custom model with OpenSeq2Seq entails the creation of a configuration script which informs all the components and hyperparameters of a model. Likewise, the configuration also specifies parameters for the three main OpenSeq2Seq modes: training, evaluation, and inference. A configuration script might contain a JSON structure with entries similar to:

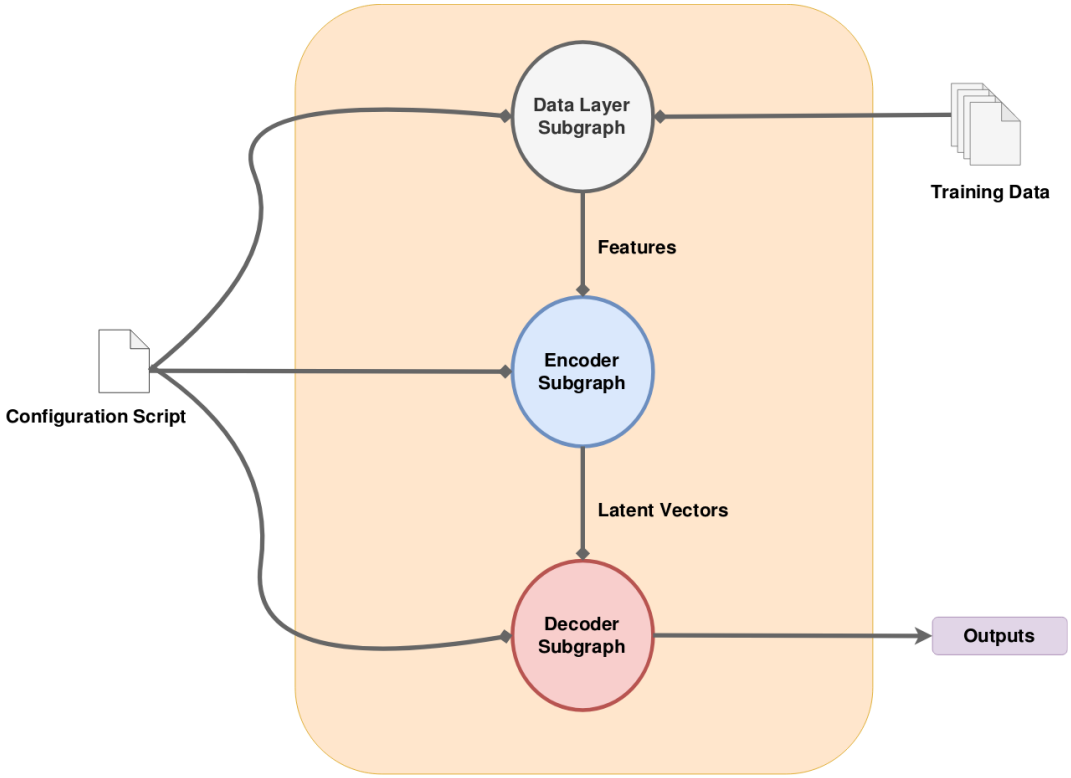

Figure 3.1: An OpenSeq2Seq system flow is structured like a directed acyclic graph (DAG). The architecture of the DAG is specified in the configuration script. This flow is used for training, evaluation, and inference of OpenSeq2Seq architectures.

```
...
'encoder': DeepSpeech2Encoder,
    'encoder_params': {
        ...
    },
'decoder': TransformerDecoder,
    'decoder_params': {
        ...
    },
...
```

Appendix A contains a full configuration JSON and some descriptions for a DeepSpeech2 model implementation.

3.1.1 Extension: Bootstrapping Additional Models via PostProcessors

OpenSeq2Seq has a mechanism to bootstrap downstream language models as an external resource and re-score the outputs of the primary encoder-decoder model. For the speech recognition task, the LM is used to find the n-best transcription sequences given the output of the ASR model. The external LM is pretrained apart from the OpenSeq2Seq system and added as a postprocessing operation. Intuitively, this process serves as a biasing function to generate the most likely transcriptions applicable to the targeted domain [71]; the LM is external from the DAG and can thus be trained on entirely separate data from the primary ASR model. OpenSeq2Seq natively supports ARPA-formatted n-gram LMs with KenLM [43] or a pretrained Transformer-XL model [23]. Figure 3.2 illustrates the bootstrapping flow.

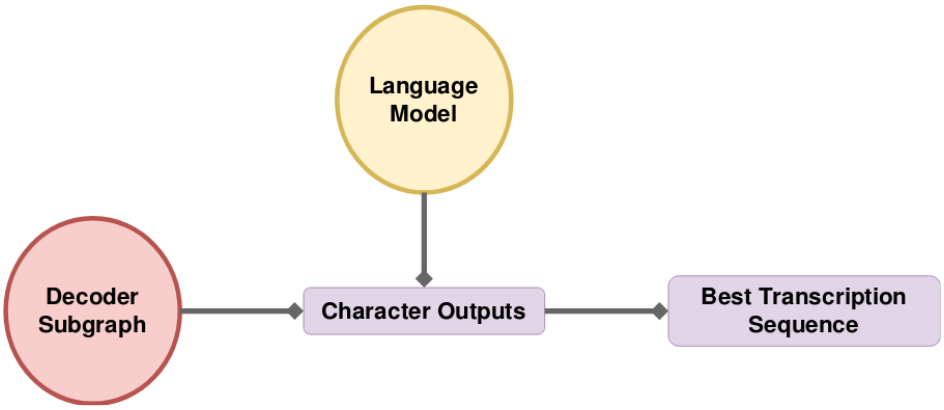

Figure 3.2: The bootstrapped language model rescores the initial output sequence from the OpenSeq2Seq decoder and generates new transcription sequences.

The bootstrapping concept can be extended beyond LMs to support any number of postprocessing steps for the core DAG flow. In order to enable the system to perform speech to text to semantics transformations, I provide an additional bootstrapping extension to

postprocess the transcription sequence by providing it as input to an SRL model. The postprocessing components are appended to the flow similarly to the natively supported LM resource, apart from one major difference: because OpenSeq2Seq's native LM integration provides transcription sequences as output, it can be used during all of training, evaluation, and inference to help achieve better WER metrics. Outputs from postprocessors like SRL have no relevance to WER and do not influence the speech recognition process, so the postprocessor additions are only supported during evaluation or inference. The joint output of the system is thus (i) the most likely speech transcription and (ii) the semantic label alignment for the transcription. Figure 3.3 illustrates the new flow.

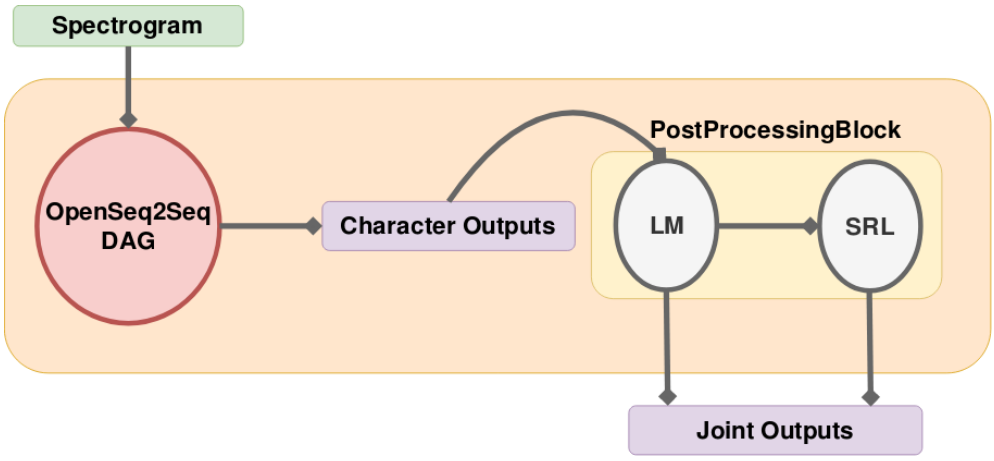

Figure 3.3: The extension to OpenSeq2Seq allows the system to bootstrap additional models downstream from the decoder and enables joint output sequences.

The extension, called a `PostProcessingBlock`, can be used to add an arbitrary number of external resources and derive additional information from the output sequence. With this extension in the JSON configuration, users can cascade postprocessors by integrating the logic directly into the OpenSeq2Seq pipeline. Unlike the natively supported LM rescorer, this extension is run automatically and requires no manual inputs beyond the implementation of a single Python class per postprocessor. A `PostProcessingBlock` configuration specifies both the desired processor to use as well as the data source for the processor. For a single processor,

the source is the output sequence of the encoder-decoder model provided by OpenSeq2Seq. In order to support more complicated flows, users can redirect outputs of other postprocessors as inputs to downstream postprocessors. For example, the augmented configuration script is exemplified as follows:

```
...
'eval_params' = {
    'data_layer': Speech2TextDataLayer,
    'data_layer_params': {
        'dataset_files': [
            ...,
        ],
        'shuffle': False,
    },
    'postprocessing_block': {
        'enable_joint_outputs': True,
        'processors': [
            {'processor': TransformerXLProcessor, 'data_source_idx': -1,
             'fuse': True, 'lambda': 0.4,
            },
            {'processor': SlingProcessor, 'data_source_idx': 0},
            ...
        ]
    },
}
...
```

As shown by the JSON entry, the configuration contains a sequence of two postprocessors: one Transformer-XL for LM rescoring, and one for invoking SLING. The `'data_source_idx'`

expects a value which corresponds to an index for the block entries. An index value of -1 indicates that the processor operates on the primary encoder-decoder output sequence. The `'fuse'` and `'lambda'` fields are hyperparameters which are discussed further in subsequent sections of this chapter. An LM in the `PostProcessingBlock` functions the same as the native LM rescoring mechanism. Within the augmented OpenSeq2Seq toolkit, I provide postprocessor implementations for Transformer-XL, SLING, and BSAT.

3.1.2 Extension: Listen, Attend and Spell with Location-Awareness

Currently, OpenSeq2Seq has predefined configuration files for several variations of DeepSpeech2 and Jasper models. As part of its toolbox of encoders and decoders, OpenSeq2Seq also contains definitions for a `ListenAttendSpellEncoder` and a `ListenAttendSpellDecoder`, but no accompanying configuration file for a standard LAS model. For the experimentation in Chapter 4, I define and utilize a configuration for a LAS-like model which uses both the LAS encoder and LAS decoder provided by OpenSeq2Seq[1]. Evaluation metrics and training times for the model are discussed in Chapter 4.

The encoder architecture for the LAS model is defined in terms of the following configuration sample:

```
'encoder': ListenAttendSpellEncoder,
    'encoder_params': {
        'convnet_layers': [ ],
        'recurrent_layers': [
          { 'type': 'gru', 'num_layers': 1,
            'hidden_dim': 512, 'dropout_keep_prob': 0.8,
            'pool': True, 'pool_size': [2], 'stride': [2],
          },
          { 'type': 'gru', 'num_layers': 1,
```

[1]Part of a core research & development deliverable while the author was at L3Harris Technologies.

```
            'hidden_dim': 256, 'dropout_keep_prob': 0.8,
            'pool': True, 'pool_size': [2], 'stride': [2],
          },
          { 'type': 'gru', 'num_layers': 2,
            'hidden_dim': 256, 'dropout_keep_prob': 0.8,
            'pool': False, 'pool_size': [-1], 'stride': [-1],
          },
        ],
    },
```

The LAS encoder exclusively uses RNN-based layers, similarly to the original LAS model described in Chapter 2. After each RNN layer, the encoder uses convolution operations to simulate the temporal reduction from the pyramidal-RNN layer. The LAS encoder accepts either LSTM or GRU RNNs. Although the authors of LAS chose LSTM-based encoders, this implementation uses a 4-layer GRU encoder, with reductions in the time dimension across the first and second layers. This choice is partially inspired by the end-to-end ASR system presented in Bahdanau, et al. [4], wherein the authors describe an RNN encoder-decoder which uses GRU operations in a 4-layer encoder structure.

The decoder subgraph is defined according to:

```
'decoder': ListenAttendSpellDecoder,
    'decoder_params': {
        'tgt_emb_size': 256,
        'pos_embedding': True,
        'rnn_type': 'gru',
        'hidden_dim': 256,
        'num_layers': 1,
        'dropout_keep_prob': 0.8,
        'attention_params': {
```

```
            'attention_dim': 256,
            'attention_type': 'chorowski',
            'use_coverage': True,
            'num_heads': 4,
        }
    },
...
```

The LAS decoder implements the function *AttendAndSpell* as defined by Chan, et al. [12]. The target embedding size parameter specifies the dimension of the vocabulary embeddings. For this model, the 26-character English alphabet is used as the target vocabulary. The decoder implementation in OpenSeq2Seq supports positional encodings, which are intended to model token order [86] and assist in producing high-fidelity output sequences. The main decoder architecture is a single-layer GRU with a hidden dimension of 256. The output of the decoder is a softmax operation which predicts the most likely character sequence. This operation is denoted as the `CharacterDistribution` by Chan, et al. [12].

The attention parameters presented in this implementation are inspired by the model in [4], which leverages location-awareness in its attention mechanism. The `chorowski` attention type is known as location-aware, as it creates a positional vector $f_{i,j} = \mathbf{F} * \alpha_{i-1}$ for each position j of the previous alignment vector α_{i-1} [18]. By convolving the vector with parameter matrix $\mathbf{F}$, the attention mechanism learns to extract salient features from previous alignments. The equations in 3.1 and 3.2 are adapted from Bahdanau, et al. [4] and Chorowski, et al. [18], and demonstrate how attention scores are generated from $f_{i,j}$.

$$e_{i,j} = \mathbf{w}^{\intercal} \tanh(\mathbf{W}\mathbf{s}_{i-1} + \mathbf{V}\mathbf{h}_j + \mathbf{U}\mathbf{f}_{i,j} + \mathbf{b}) \tag{3.1}$$

$$a_{i,j} = \frac{\exp(e_{i,j})}{\sum_j^J \exp(e_{i,j})} \tag{3.2}$$

where $\mathbf{W}$, $\mathbf{V}$, and $\mathbf{U}$ are weight matrices, and $\mathbf{w}$ is a weight vector. The operation $\mathbf{W}\mathbf{s}_{i-1}$

computes a matrix from the previous RNN timestep $\mathbf{s}_{i-1}$, and $\mathbf{Vh}_j$ generates a matrix from the current content vector $\mathbf{h}_j$. The addition of $\mathbf{Uf}_{i,j}$ lets the scorer account for previous alignments. Finally, the alignment vector for the current timestep is taken from the softmax-normalization in 3.2.

The implication of using location-aware attention in speech processing is that the model will eventually learn to attend to only the salient features of the audio signal from its latent vector encoding while simultaneously accounting for signal history. Attention functions which can robustly capture location contexts have been demonstrated to outperform standard content-based attention [18]. Chiu, et al. [15] observes that extending the LAS model to support multi-headed attention helps it isolate clean speech from degraded background noise by allocating noisy features to a single attention head, and ultimately benefits performance metrics. In my LAS implementation, I couple the location-awareness of `chorowski` attention with the multi-headed processing and report a WER competitive to the base LAS model.

OpenSeq2Seq's toolbox of attention tools contains functions for `chorowski` attention and multi-headed processing.

3.2 System Compositions for Single-shot Speech Tagging

To properly construct an end-to-end flow for the speech tagging task, the experimental system is set up such that it can transform speech to text to semantics in a single pass. By leveraging the concepts discussed in Section 3.1, when the OpenSeq2Seq DAG is loaded into evaluation or inference mode the ASR outputs are automatically routed to the desired series of postprocessors for further analysis, which can include predicate-argument identification. This pipeline is the experimental framework used to evaluate a set of system compositions for the ASR-SRL task.

I evaluate 6 different combinations of ASR and SRL models, such that each of the three supported ASR models are evaluated alongside both SRL models. For every system composition, a Transformer-XL model is used as the intermediate LM. I compare the different evaluations to a baseline system which uses a standard HMM-GMM ASR model and an algo-

rithm similar to the one discussed in [79]. In general, the system is designed to characterize three facets of ASR-SRL experimentation: (i) whether or not end-to-end neural systems can outperform traditional systems for this particular task; (ii) which encoder-decoder architectures are most conducive for downstream processing given a balanced training procedure; and (iii) the effect of model coupling in the end-to-end system. Section 3.2.1 elaborates on strategies for model coupling.

In terms of system architecture, OpenSeq2Seq is the implementation platform for each ASR model. I provide a novel configuration for LAS and a heavily modified configuration for DeepSpeech2. To implement Jasper10x5, I utilize a pretrained model released by the authors [57] and the corresponding open-sourced configuration file. Every configuration file is augmented with the `PostProcessingBlock` field in the evaluation and inference sections of the JSON.

All ASR models are equipped with similar training, evaluation, and inference parameters to the Jasper10x5 configuration recipe: the models have a data augmentation step akin to SpecAugment [64], wherein audio features undergo time warping and time-frequency masking to help make the model robust to noise and benefit model generalization. The SpecAugment authors report competitive WER metrics for LibriSpeech and Hub5'00 (LDC2002S09, LDC2003T02) with an augmented LAS model. The SLING and BSAT models are trained on the same dataset. Chapter 4 describes training procedures in greater detail.

3.2.1 Model Fusion Strategies

Each supported system composition for transcribing speech and identifying semantic structure adopts a philosophy of shallow model fusion [38], whereby the ASR model and the LM compute a joint decoding function similar to the one described in Stahlberg, et al. [81] and Toshniwal, et al. [85]:

$$\hat{y} = \texttt{argmax}_y \log P_{ASR}(\mathbf{y}|\mathbf{x}) + \lambda \log P_{LM}(\mathbf{y}) \tag{3.3}$$

where $\log P_{ASR}(\mathbf{y}|\mathbf{x})$ is the log probability distribution given by the ASR decoder and $\lambda \log P_{LM}(\mathbf{y})$ is the weighted log probability distribution of the best LM hypothesis. Similar to standard LM rescoring operations, this type of log-linear interpolation with the LM is typically invoked at inference time using standard decoding algorithms like beam search or greedy search [50].

Shallow fusion has become a common mechanism for integrating LMs into end-to-end ASR models ([95], [19], [50], [5]). Other variations of model fusion provide a tighter coupling between the LM and the core encoder-decoder model, such as deep fusion [38] and cold fusion [80]. In both deep fusion and cold fusion, the LM is integrated into the primary computation graph to be used during training, and thus influences how the ASR model learns. For shallow fusion, Equation 3.3 could potentially be aggregated into a single computation graph or support more sophisticated scoring mechanisms. For example, an experiment in Kannan, et al. [50] performs shallow fusion between a LAS model and an external LM with a weighted penalty coefficient $\gamma c(\mathbf{x}, \mathbf{y})$:

$$\hat{y} = \texttt{argmax}_y \log P_{ASR}(\mathbf{y}|\mathbf{x}) + \lambda \log P_{LM}(\mathbf{y}) + \gamma c(\mathbf{x}, \mathbf{y}) \tag{3.4}$$

$$c(\mathbf{x}, \mathbf{y}) = \sum_j \log(\min(\sum_i a_{i,j}, 0.5)) \tag{3.5}$$

where $a_{i,j}$ is a probability from the attention vector at the frame-level. Sriram et al. [80] and Chorowski, et al. [19] both posit that by including frame-level values in a penalty calculation, the decoder is biased toward outputting full transcripts over shorter sequences. Equation 3.4 can be extended to combine more complex sequences of scores: Seki, et al. [75] discusses shallow fusion with an RNN-LM which performs log-linear interpolation with CTC prefix scores, decoder scores, and LM scores. Shallow LM fusion may be applied with n-gram models or neural LMs.

Currently, only shallow model fusion is supported in the `PostProcessingBlock` for OpenSeq2Seq. Shallow fusion is traditionally applied at inference time, which is more closely aligned with the purpose of the overall extension. The motivation for integrating the shallow fusion strategy is twofold: (i) model fusion is noted as a potential improvement for the system described in Bahdanau, et al. [4], wherein the authors suggest that an external LM trained on larger text corpora could be fused for improved performance; and (ii), a tight integration of the ASR and LM models ultimately makes the system closer to being truly end-to-end. Figure 3.4 depicts the end-to-end flow coupled with LM fusion.

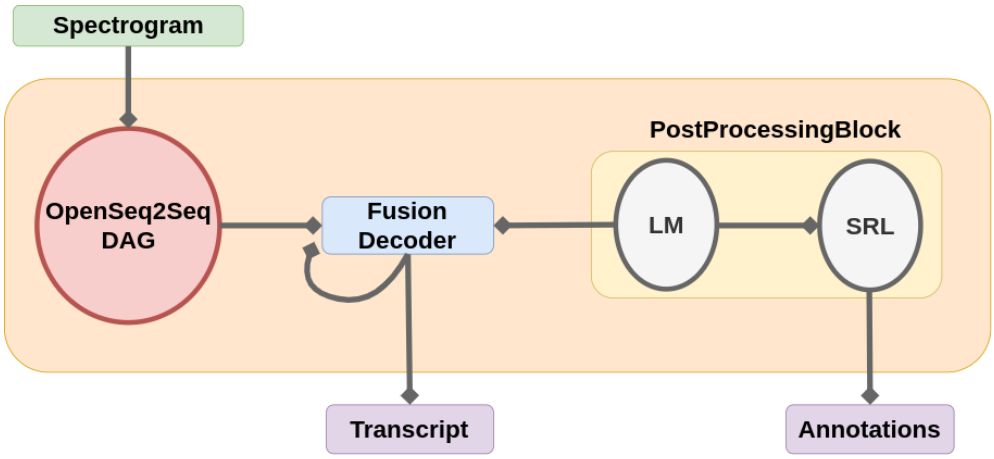

Figure 3.4: Fusion decoding with an LM postprocessor.

As part of the `PostProcessingBlock` extension, a shallow fusion decoder has been implemented to perform the same kind of log-linear interpolation described in Equation 3.3. Currently greedy search is the only supported decoding algorithm. To enable shallow fusion, the extension configuration must set the boolean flag `'fuse'` and specify a weight value. Fusion is only supported between the core OpenSeq2Seq DAG and an LM postprocessor. If fusion is disabled in the LM configuration, the system will default to using the LM for standalone rescoring. The SRL postprocessors remain decoupled from the core computation graph and function as external resources.

3.3 Summary

Chapter 3 has described the major contributions presented in this thesis. I have described an OpenSeq2Seq system for rapidly building end-to-end pipelines for speech transcription and joint semantic annotation. The system uses a custom extension for adding additional downstream models to postprocess the output of the primary encoder-decoder network. The `PostProcessingBlock` extension allows users to specify a set of postprocessing operations which can derive additional features from the output sequence. Furthermore, postprocessors can be arranged such that the output of one processor can feed into another processor. For the task of speech transcription tagging, the supported extensions include a `TransformerXLProcessor` and two SRL operations: the `SLINGProcessor` and `BSATProcessor`. The extension is usable at the evaluation and inference stages.

Beyond the `PostProcessingBlock`, I presented a novel implementation of the LAS model which is compatible with the OpenSeq2Seq DAG. This model takes advantage of several components contained natively within the toolkit, including encoder and decoder subgraphs designed for LAS-like models. This implementation of LAS takes inspiration from [15] and uses multi-headed attention with a locative attention mechanism designed by [18].

The final section of this chapter discussed practical and philosophical motivations for the various end-to-end system combinations. In order to fairly evaluate each system compositions, each model within the ASR and SRL model sets undergo an identical training and evaluation procedure with the same data augmentation steps inspired by SpecAugment. Each system composition uses Transformer-XL as the intermediate LM. Finally, the broader question of how to properly establish a comprehensive end-to-end neural network flow was partially addressed. Techniques like model fusion are one way to tightly couple two distinct computation graphs as a single cohesive network. The `PostProcessingBlock` can be enabled with shallow model fusion to perform joint decoding using probability distributions from both the ASR model and the LM.

Chapter 4

EXPERIMENTS & ANALYSIS

Having now defined the end-to-end architecture and its various features, Chapter 4 describes the experimental setup used to evaluate the speech to text to semantics process. Specifically, this chapter explains:

1. Experimental environment: Because OpenSeq2Seq allows GPU distributed training, the entirety of this system takes advantage of a multi-GPU environment to accelerate training times.

2. Training & evaluation procedures: In order to fairly evaluate the different model combinations, I use a common training procedure for each of the ASR and SRL models. ASR models are pretrained on clean data and are then fine-tuned on a more challenging dataset. This process is necessary since the end-to-end evaluation corpus containing speech data, transcriptions, and semantic annotations is an out-of-domain conversational speech corpus and proves to be quite challenging. The SLING and BSAT models are each trained on the CoNLL-2012 dataset [69]. I additionally describe the construction of the baseline system.

3. Results & discussion: To conclude the chapter, I list the evaluation results for the joint outputs of the end-to-end system as compared to the baseline system. In all cases, the end-to-end system outperforms its baseline. Specifically, model compositions using both CTC decoders and SLING perform most competitively on synthetically tagged data in terms of WER and F1. The system performs significantly worse on the conversational speech corpus. I discuss potential error mitigation techniques.

4.1 Experimentation Environment

All training procedures described in this chapter are conducted on an NVIDIA DGX Station [1] with 4 Tesla V100 GPUs and 256GB of RAM. Like the base OpenSeq2Seq implementation, my extended system uses Horovod [77] to enable asynchronous training across multiple GPUs.

4.2 Training & Evaluation Procedures

This section elaborates on the training and evaluation procedures for both the ASR and SRL model sets.

4.2.1 ASR Training Procedure

The targeted evaluation corpus for the end-to-end system is a portion of the CallHome conversational English corpus. A set transcripts from this corpus is provided as part of the OntoNotes v5 release [45] and has PropBank annotations. This is the same dataset used to evaluate the Shrestha, et al. [79] system. CallHome is a particularly challenging dataset, since the speech is conversational and not always clean [74]. Moreover, the corpus only contains 120 hours of speech data, which is rather small for training end-to-end neural network implementations. To mitigate this issue, the ASR models are trained with a complex multi-stage procedure that ensures each model is trained sufficiently in one domain, and then fine-tuned to adapt to the more challenging domain.

For all stages of the ASR training procedure, I use 64-dimensional log-mel filterbank features and a data augmentation mechanism, inspired by SpecAugment, which applies time and frequency masks of size 2 to the features.

---

[1] Hardware provided by L3Harris Technologies, ComCept division. More information on the hardware is available at https://www.nvidia.com/en-us/data-center/dgx-station/

Pretraining

The reference implementations of the DeepSpeech2, Jasper, and LAS models undergo vastly different training processes. For example, as noted in Chapter 2, the original DeepSpeech2 and Jasper models are trained in a distributed environment with vastly different training requirements (20 epochs and 400 epochs, respectively) and on different datasets. The original LAS implementation is similarly distributed, but trains until convergence on an internal dataset.

The procedure starts with a pretraining step using clean speech data from the LibriSpeech corpus. The ASR models train on the clean data until convergence. Table 4.1 shows the results of the pretraining step. Note that because I use an off-the-shelf Jasper model that has already been trained on LibriSpeech, it does not need further pretraining. I use the train-clean splits and train-other-500 split for the pretraining step, with a total corpus size of about 1324 hours of English speech.

Model	Epochs	GPUs	Training time (hours)	WER
DeepSpeech2	96	4	218	8.38
LAS	125	4	539	16.57
Jasper	N/A	N/A	N/A	3.61

Table 4.1: ASR pretraining procedure results over 1324 hours of LibriSpeech data

DeepSpeech2 converged fastest, with only 96 epochs of training time across 4 GPUs. LAS took significantly longer, at 125 epochs and nearly 3.5 weeks of training time on 4 GPUs. Even with the GPU distribution, per-batch processing time was observed to be significantly longer with LAS. The off-the-shelf Jasper model comes pretrained to production quality for English ASR. Jasper and DeepSpeech2 use CTC loss calculation, and LAS uses a weighted cross-entropy sequence loss. The models all use polynomial loss decay with a learning rate floor of 0.000001.

For evaluating the pretrained models, I use the dev-clean split from LibriSpeech. The initial WER metrics are raw with no LM rescoring. The DeepSpeech2 model performed competitively with OpenSeq2Seq's open-source alternative, which has a 6.71 WER[2]. The raw LAS WER is 2.47% worse than the best-reported reference model with a 14.1 WER, and 10.77% worse than the best LM-free LAS in Chiu et al. [15]. The difference could be attributed to the amount of data used to train this model versus the amount used to train the reference models: Chiu et al. [15] train the improved LAS on 12,500 hours of data, and the amount of data in the LibriSpeech corpus might be insufficient for the LAS architecture.

Fine-tuning

The main training cycle is intended to fine-tune the pretrained models such that they adapt to an alternate domain. To that end, the main training cycle uses two datasets: (i) the SI-284 split from the WSJ corpus [67], a broadcast news dataset containing English read speech, and (ii) a portion of the CallHome corpus. Both of these datasets are more challenging than the clean LibriSpeech splits.

Data from the CallHome corpus is dual channeled; to properly utilize the dataset, each audio file in the corpus is split and aligned to is corresponding transcript. This was largely a manual process, with some utilization of Kaldi [68] tooling. The resulting training set contains 105 hours of English speech. Figure 4.1 illustrates the WER results at 10-epoch increments during the fine-tuning process.

As expected, every model WER is increased when adapted to the new domain. The losses for DeepSpeech2 and Jasper diverged early in the training process but ultimately converged to a reasonable level. The LAS loss is observed diverging throughout the entirety of the training process. Each model is fine-tuned for 50 additional epochs beyond the initial pretraining step.

---

[2]More information is available at https://nvidia.github.io/OpenSeq2Seq/html/speech-recognition.html

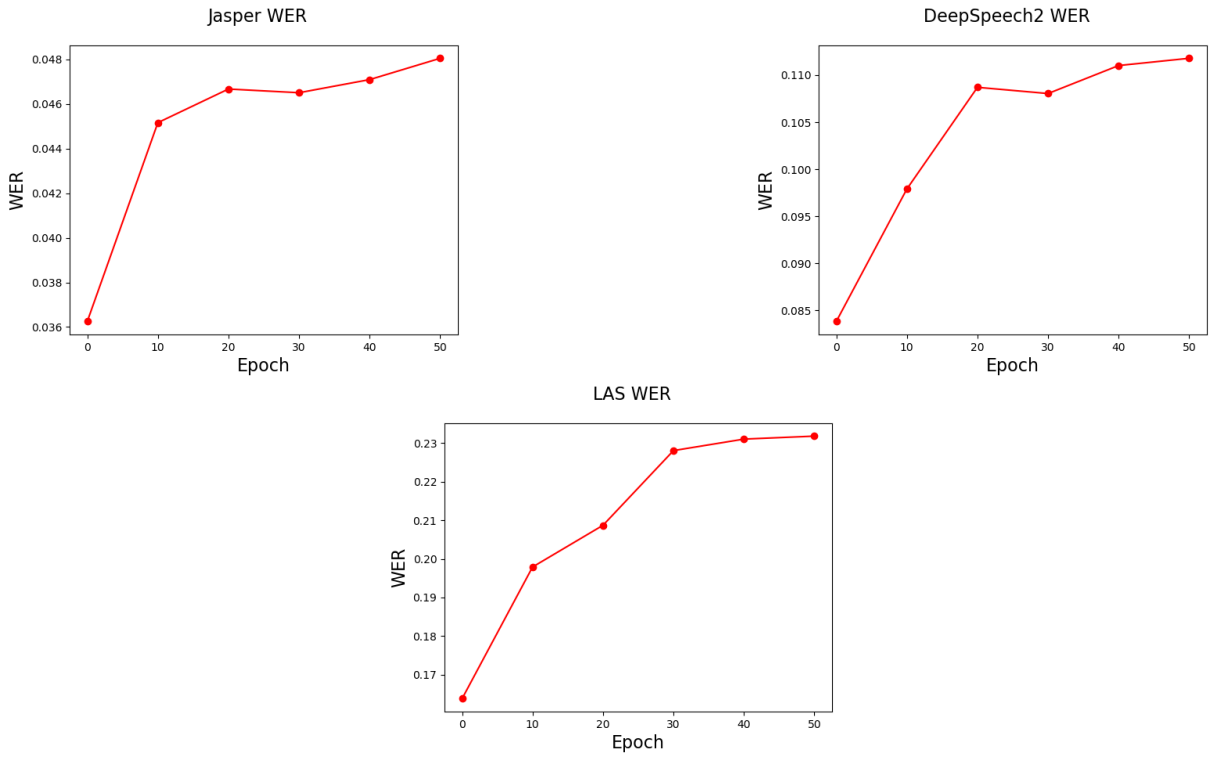

Figure 4.1: Raw fine-tuned evaluation WERs

4.2.2 Synthetically Annotated WSJ Transcripts

In addition to evaluating the CallHome corpus for both speech recognition and semantic annotation, I evaluate the full system on an internal synthetic dataset derived from the WSJ eval92 split. For each transcript in the dataset, I use the AllenNLP [33] implementation of He, et al. [41]'s SRL model to generate PropBank annotations and create a CoNLL-formatted corpus. The synthetic corpus is then manually reviewed for errors and corrected. The synthetic corpus contains 333 lines of annotated transcript data.

4.2.3 SRL Training Procedure

The procedure for training the SRL models is not quite as advanced as the ASR procedure; there is no need for domain adaption since the SRL training data is from the same corpus as the end-to-end evaluation data. As a result, I use the CoNLL-2012 data set to train both SLING and BSAT with the standard train split.

Neither SRL model implementation supports distributed GPU training, so both models are retrained using CPU resources from the same hardware described in Section 4.1. Table 4.2 provides the training results for both models.

Model	F1
SLING	78.48
BSAT	74.31

Table 4.2: SRL training procedure results over 50000 steps on the CoNLL-2012 corpus

The SLING authors report that their best model converged after 120,000 steps with an F1 of about 90 for frame parsing. Their argument role parsing is reported to be much worse, converging with an F1 of roughly 60. The results reported in Table 4.2 are for the best full-frame parses (all predicate and argument labels). More steps could potentially yield higher scores, but both models were observed converging slightly before training was stopped.

For the BSAT model, Cai, et al. [11] use a rich embedding representation which includes part-of-speech embedddings. In my implementation, I alter the embeddings to exclusively use pretrained word embeddings, lemma embeddings, and predicate-indicator embeddings as described in Chapter 2. The complete embedding representation is thus $e = e^{(p)} \oplus e^{(l)} \oplus e^{(i)}$. Including the part-of-speech embeddings would require an additional system postprocessor to assign the corresponding labels upstream from the SRL model, which is beyond the scope of this research task.

4.2.4 Baseline

The baseline system used to benchmark this experiment is an implementation of the SRL algorithm inspired by Shrestha, et al. [79]. The baseline uses a trigram HMM-GMM ASR model using the Kaldi toolkit following the 's5' WSJ recipe[3]. The LM is a 4-gram ARPA-formatted model created from the same recipe. The baseline ASR model is trained similarly to the procedure described in Section 4.2.1, using both clean splits from LibriSpeech as well as the SI-284 split from WSJ. I evaluate the baseline ASR training on two evaluation corpora, the results of which are given in Table 4.3.

Corpus	WER
LibriSpeech dev-clean	7.99
WSJ dev93	10.48

Table 4.3: Baseline ASR system results for two evaluation corpora

As hyperparameters for the segment selection algorithm, I evaluate three window sizes following the reference implementation: 5, 8, and 10. I use the *take-shortest* heuristic. Unlike Shrestha et al. [79], I find that using larger window sizes yields better semantic annotations. This difference is likely due to the types of semantic parsers used to infer the predicate-

[3]https://github.com/kaldi-asr/kaldi/tree/master/egs/wsj/s5

argument annotations; SLING and BSAT are sequence models, and thus can properly handle long input sequences. Section 4.3.3 discusses this phenomenon further. Table 4.4 presents the results of the algorithm given the ground-truth transcription sequences.

	SLING		BSAT	
Window	CallHome	WSJ eval92	CallHome	WSJ eval 92
10	**54.38**	**67.68**	44.07	**61.91**
8	50.69	67.39	**46.13**	55.73
5	37.88	52.48	39.72	48.58

Table 4.4: Baseline SRL F1 results given ground-truth transcripts

4.2.5 Aligning Hypotheses with Gold Standard Annotations

I follow a convention similar to the one described by Favre, et al. [30] for aligning hypothesis annotations to their gold standards. The authors note that because the hypothesis annotations for speech transcripts might not be directly aligned with their reference annotations, using standard CoNLL tools for evaluation is not a viable option. As such, I use their open-sourced evaluation scripts[4] to generate WER and F1 scores.

To align references and hypotheses, Favre, et al. [30] perform a forced alignment between head-words of the two sequences. Figure 4.2 is adapted from Favre, et al. [30] and provides an example alignment. In the example, the head-words are aligned, but the speech recognition system generates an insertion w_{h2} between the hypothesis head w_{h3} and its corresponding relation w_{h1}. This insertion disrupts alignment between other sequence features. To compensate, a set of matching heuristics is enforced such that if a semantic relation arc has a properly aligned predicate, the arc is considered correct. The authors note that this decreases F1 correlation with WER, but is a more straightforward evaluation mechanism

[4]https://code.google.com/archive/p/srleval/

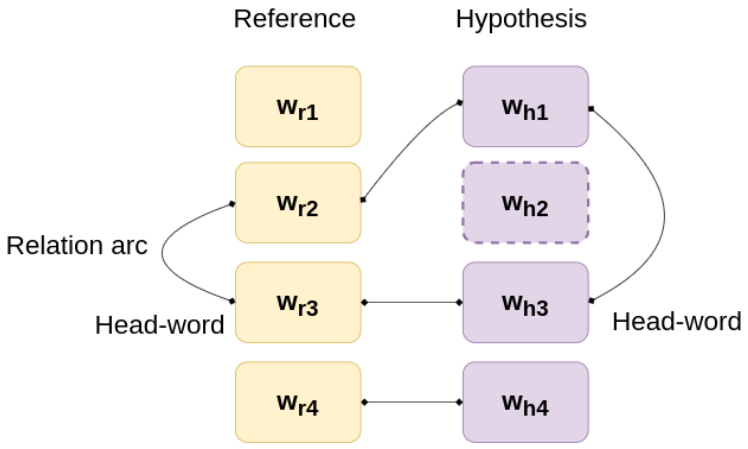

Figure 4.2: Example reference and hypothesis alignment

than attempting to align at the annotation-level. The latter approach would entail remediating insertions and deletions in the hypothesis to ensure the proper annotations are given to the proper tokens.

In order to properly use the modified `srleval` scripts, I annotate the hypothesis and reference sequences with syntactic dependency labels using SyntaxNet [3]. The evaluation script requires full syntax annotations in CoNLL'09 format to align references and hypotheses by head-word. From there, the evaluation mechanism computes micro- and macro-averaged F1 values for the syntactic and semantic feature sets.

For each computed F1 score, I report only the semantic measurements. Semantic frames are considered correct when both the predicate and all accompanying arguments are properly identified.

4.3 Experiments

This section quantitatively and qualitatively presents the results of two separate end-to-end experiments. The first experiment evaluates the neural speech to text to semantics transformation process with n-best rescoring in comparison with the baseline system. The second experiment re-evaluates the transformation pipeline with shallow fusion enabled during beam

search decoding. The chapter concludes with a discussion of the results and potential error mitigation techniques for future research in this domain.

Model	CallHome	WSJ eval92
baseline-10	30.8	11.31
DeepSpeech2	15.75	9.29
Jasper	**10.18**	**6.42**
LAS	18.83	16.99

Table 4.5: Experiment 1 - End-to-end WER metrics for the CallHome and WSJ eval92 evaluation corpora

	CallHome			WSJ eval92		
Model	P	R	F1	P	R	F1
baseline-10 + SLING	0.437	0.233	0.304	0.679	0.482	0.564
baseline-10 + BSAT	0.399	0.186	0.254	0.614	0.426	0.503
DeepSpeech2 + SLING	0.495	0.431	0.461	0.568	0.502	0.533
DeepSpeech2 + BSAT	0.463	0.411	0.435	0.549	0.494	0.520
Jasper + SLING	**0.599**	**0.568**	**0.583**	**0.671**	**0.658**	**0.664**
Jasper + BSAT	0.557	0.532	0.544	0.643	0.610	0.626
LAS + SLING	0.427	0.418	0.422	0.455	0.426	0.44
LAS + BSAT	0.374	0.251	0.300	0.403	0.349	0.374

Table 4.6: Experiment 1 - Full-frame annotation metrics over the generated transcription sequences for CallHome and WSJ eval92

4.3.1 Experiment: End-to-End Transformation with LM Rescoring

Tables 4.5 and 4.6 detail the experiment results in terms of WER and F1, respectively, for the CallHome corpus and the synthetically annotated eval92 split from WSJ. The baseline ASR system is rescored with the 4-gram ARPA LM, and all neural system transcripts are generated by rescoring with an (un-fused) Transformer-XL model pretrained on WSJ and fine-tuned with parts of CallHome. The baseline segment selection is set to a window of size 10 (denoted as 'baseline-10'). The baseline system is evaluated using CPU resources and each neural system combination uses 1 Tesla v100 GPU to speed up evaluation time.

Model	CallHome	WSJ eval92
DeepSpeech2	14.43	9.08
Jasper	**8.77**	**6.03**
LAS	17.42	15.95

Table 4.7: Experiment 2 - End-to-end WER metrics using a pretrained shallow-fused Transformer-XL language model

4.3.2 Experiment: End-to-End Transformation with Shallow Fusion Decoding

Tables 4.7 and 4.8 describe WER and F1 results when each neural network model uses a shallow fusion beam search decoder to generate transcription sequences. I use a λ value of 0.4 and a beam width of 100. Similarly to the first experiment, the neural systems use a single GPU during evaluation. Note that the baseline ASR system does not support shallow fusion, so I omit it from this experiment.

The fusion decoder implemented for this experiment computes a linear interpolation between the Transformer-XL LM and the ASR log-probability distributions, which can be the native log-linear probability tensors emitted from Tensorflow's CTC decoder or the softmax probability distributions emitted from the LAS implementation.

	CallHome			WSJ eval92		
Model	P	R	F1	P	R	F1
DeepSpeech2 + SLING	0.502	0.441	0.469	0.573	0.515	0.542
DeepSpeech2 + BSAT	0.422	0.472	0.445	0.563	0.509	0.534
Jasper + SLING	0.572	0.558	0.564	**0.691**	**0.677**	**0.683**
Jasper + BSAT	**0.61**	**0.586**	**0.597**	0.657	0.629	0.642
LAS + SLING	0.433	0.401	0.416	0.463	0.441	0.451
LAS + BSAT	0.361	0.195	0.253	0.411	0.352	0.379

Table 4.8: Experiment 2 - Full-frame annotation metrics over the output of the Transformer-XL model for CallHome and WSJ eval92

4.3.3 Results Analysis

The final section of this chapter concludes with an analysis and discussion of the experimentation results. In both experiments, system combinations using the Jasper model outperform all others in both WER and F1. DeepSpeech2 performs comparably, with only a 4.43% relative WER difference on CallHome and a 2.87% relative WER difference on eval92. SLING seems to provide the best F1 metrics, which is not surprising since it outperforms BSAT in terms of both SRL training and evaluation metrics.

Section 1.1 described two central questions for this thesis: 1) could an end-to-end syntax-agnostic system be adequate for the speech to text to semantics task, and 2) what variations of encoder-decoder architectures are most suitable for such a system.

As an empirical answer to the first question, the experimentation results indicate that end-to-end systems could indeed be suitable for generating semantically annotated speech transcriptions, even without relying on linguistically-informed features like syntax or punctuation. The answer to the second question is more multi-faceted: although the results indicate that CTC ASR models with SLING yield the best performance, it would be unfair to claim

that RNN-based models like LAS are unviable. A more personalized training procedure for the OpenSeq2Seq LAS might yield better results; it is well known that CNN-based models are easier to train than RNN models. Similarly, SLING consistently outperforms BSAT, but the retrained BSAT model used in these experiments is worse than the reference implementation reported by Cai, et al. [11], which has state-of-the-art performance. A future research endeavour could properly investigate model tradeoffs for this end-to-end system and provide a more concrete answer to the second central question of this thesis.

The end-to-end results are encouraging overall. Certain areas like fusion decoding and the LAS model performance could still be improved, but evidence shows the viability of a neural speech to text to semantics transformation pipeline. This section concludes the chapter with an analysis of (i) the performance of the OpenSeq2Seq LAS implementation, (ii) the shallow fusion decoder, and (iii) the viability of the baseline algorithm.

Improving LAS Results

The LAS model did not perform quite as well as DeepSpeech2 or Jasper, and only outperformed the baseline system on the CallHome evaluation corpus. As a follow-on to this experiment, my implementation of LAS could be enhanced with additional optimizations in Chiu, et al. [15] such as label-smoothing or wordpiece integration. Evidence shows that a longer training time could also result in better performance:

As we can see in Figure 4.3, the training loss converges to a local minimum quite early. A more personalized training procedure for LAS could include a manual annealing step, such as the one described for the RNN encoder-decoder in Bahdanau, et al. [4], which restarts the optimizer with a lower learning rate value. The learning rate is continually decreased until the log-likelihood value stops improving. The reference implementation of Bahdanau, et al. [4]'s model has 2 annealing steps[5]. Such a procedure could help the model to continue converging beyond the local minimum.

[5]https://github.com/rizar/attention-lvcsr

Figure 4.3: LAS training loss after every epoch

Effects of Shallow Fusion

The primary goal of enabling shallow fusion is to generate higher-fidelity transcriptions which could benefit more accurate semantic tagging. As shown in Table 4.7, the shallow fusion decoder in this system does not drastically improve WERs. For the end-to-end task, the semantic annotation results do improve. Systems using the Jasper model still outperform the other combinations; Jasper+BSAT and Jasper+SLING show the best performance on the CallHome and eval92 evaluations, respectively.

The WER results seem to be in contrast with other fusion experiments; for example, the experiments with shallow fusion decoding in Zeyer, et al. [95] report a 27% WER improvement by integrating an LSTM LM trained on a byte pair encoding (BPE) vocabulary [76]. BPE is a type of subword encoding commonly used in machine translation tasks ([76], [78], [20], among others). Evidence shows that using BPE units to decompose the target sequences can greatly benefit model accuracies, especially when used alongside other units like characters [27].

Apart from modifying the target unit set, incorporating a more sophisticated interpola-

tion is one potential way to improve outputs from the fusion decoder. Applying coverage penalties like the one described in Kannan, et al. [50] or Chorowski & Jaitly [19] is one potential improvement for the decoder. Inaguma, et al. [47] compute $P_{ASR}(\mathbf{y}|\mathbf{x})$ as a log-linear interpolation of the CTC probability distribution and the softmax distribution from the decoder, and then perform shallow fusion with an RNN-LM. In their monolingual experiments, shallow fusion shows obvious improvements over the baseline. By comparison, the fusion implementation evaluated for this experiment is less sophisticated, but nonetheless provides a worthy baseline for future fusion experimentation with OpenSeq2Seq.

Baseline Evaluation

One of the major differences between the baseline system and the end-to-end neural pipeline is that the baseline flow did not provide the entirety of the transcription sequence as input to the SRL models in one pass, but instead used Shrestha, et al.[79]'s segmentation algorithm to generate windowed snippets as an incremental process. For classical SRL systems, the incremental approach seems valid; the experiments in Shrestha, et al. [79] used the ASSERT SRL system [70] to generate semantic annotations. ASSERT is an SVM classifier and thus does not have the benefits of LSTM-centric systems like SLING or BSAT, namely, it does not innately model long-distance dependencies within sequences. By feeding in transcriptions at the segment-level, the baseline system occasionally misses long-distance dependencies. When observing the results of the baseline compared to any of the neural systems, this is one of the most apparent issues. An example of this phenomenon is provided in Table 4.9, which depicts output annotations from the baseline-10 system in comparison to the gold standard.

The example in Table 4.9 is centered around finding the semantic frame anchored at the ROOT predicate `RECEIVED`. The hypothesis sequence is able to correctly identify all frame arguments to the left of the predicate, but only identifies parts of ARG-4 and none of ARG-2. This misclassification is due to the segment window moving out of context for `RECEIVED`, and subsequent inputs to the SRL model do not contain the target predicate. Upon deeper inspection of the output sequence, the hypothesis annotations do correctly identify other

semantic frames, such as the frame anchored at `AWARD`.

As shown by the earlier results in Table 4.2, neither SRL model is perfect at sequence tagging, so it would not be fair to discount the viability of a segmentation algorithm like the one used in the baseline system. Certain applications of a speech to text to semantics system, like real-time audio streaming, could potentially benefit from a windowing-based algorithm to select relevant portions of the streaming output for semantic annotation. But for single-shot sequence transformations where the system has access to the entirety of the transcription, segment selection seems detrimental to the annotation results.

4.4 Summary

This chapter has discussed the experimentation procedures used to evaluate an end-to-end system for semantically annotating spoken language data, and provided an analysis of the experiment results. The ASR components in the end-to-end OpenSeq2Seq system underwent a multi-stage training procedure to compensate for the necessary cross-domain evaluation with the CallHome corpus. One of the only accessible corpora that has both audio, transcriptions, and semantic annotations is a portion of CallHome. Using a mixture of LibriSpeech, WSJ, and CallHome, the speech recognition modules are trained to be competitive with several other published ASR models. The SLING and BSAT SRL models are each retrained on the standard CoNLL-2012 dataset and perform comparably with one another.

I execute two separate experiments for this thesis. The first experiment assesses the viability of an end-to-end neural transformation pipeline to map spoken language data to semantically annotated text. The experiment is evaluated alongside a traditional baseline system using a GMM-HMM ASR model and a segmentation algorithm to select relevant portions of the resulting transcription as input to an SRL model. The second experiment evaluates the same end-to-end transformation with decoder that performs shallow fusion between the ASR and LM components. I evaluate compositions of ASR and SRL models and find that systems using CTC decoders with a SLING semantic parser outperform the baseline by a reasonable margin. The LAS model performs quite poorly by comparison, though could

potentially be improved through longer training times and additional optimizations.

Enabling shallow fusion in the decoder did not heavily influence the result metrics, but showed similar results to decoding with n-best LM rescoring. This phenomenon contradicts related research in shallow fusion, where enabling fusion with beam search decoding will typically improve WERs for sequence-to-sequence models. There are some optimizations which could improve OpenSeq2Seq shallow fusion in future research. One potential improvement is modifying the LM unit set to support BPE or other subword units has been shown to yield better results during fusion decoding. Alternatively, incorporating more a sophisticated interpolation has been demonstrated to improve decoding.

The experiments have indicated that the baseline system is unable to properly handle long-distance semantic dependencies. Because the baseline algorithm processes transcriptions at the segment-level, rather than the full transcription, certain frame elements are not properly identified when the segment window is too small. Given that modern SRL models like SLING and BSAT are LSTM-based, using entire transcription sequences is as input is a more natural way to perform semantic annotation.

In general, the results of these experiments are quite encouraging and have demonstrated the viability of an end-to-end transformation to generate semantically annotated speech transcriptions. Transcription sequences can be properly annotated without needing syntactic feature enrichment, assuming the system is equipped with an adequate ASR model and a syntax-agnostic SRL model.

Token	Hyp Annotation	Ref Annotation
IN	(ARG-TMP*	(ARG-TMP*
SEPTEMBER	*)	*)
THE	(ARG-0*	(ARG-0*
COMPANY	*)	*)
RECEIVED	(V*)	(V*)
NINETY	(ARG-1*	(ARG-1*
SIX	*	*
POINT	*	*
EIGHT	*	*
MILLION	*	*
DOLLARS	*)	*)
AS	(ARG-4*	(ARG-4*
ITS	*	*
SHARE	*)	*
OF	_	*
DAMAGES	_	*)
FROM	_	(ARG-2*
A	_	*
BREACH	_	*
OF	_	*
CONTRACT	_	*
AWARD	_	*
RELATED	_	*
TO	_	*
A	_	*
COAL	_	*
PARTNERSHIP	_	*)

Table 4.9: Baseline-10 + SLING hypothesis sequence and corresponding gold standard with a semantic frame at the ROOT predicate

Chapter 5

CONCLUSION

This thesis presents an end-to-end system for transforming speech to text to semantics, without requiring an intermediate representation dependant on syntactic annotation. This type of generalized spoken language understanding is still a nascent research area: advancements in the fields of spoken dialogue systems or conversational AI are typically geared to one or more topic areas in the form of slot-filling. Slot-filling systems solve the task of parsing transcribed text to fill predefined semantic slots, and thus cannot generalize to identify arbitrary semantic roles. The system described in this thesis uses shallow semantic parsing, or SRL, to annotate full frame predicate-argument roles across the transcription sequence using Propbank-style annotations.

The introduced system uses a series of sequence-to-sequence neural networks. The foundation of the pipeline is OpenSeq2Seq, a modular toolkit for building encoder-decoder models built on top of the Tensorflow deep learning platform. Using a set of predefined building blocks within the toolkit, the system supports three ASR models with distinct architectures: DeepSpeech2, Jasper, and LAS. DeepSpeech2 and Jasper both use CTC decoders to generate the most likely character alignment given the audio signal. DeepSpeech2 uses a hybrid CNN-RNN encoder, in which the CNN layers allow the model to extract spatial features from the audio signal, and the downstream RNN layers act as sequence encoders to a latent vector representation. Jasper exclusively uses CNN layers in its encoder, and leverages 1D convolution operations for fast and distributed sequence learning. LAS is different from DeepSpeech2 and Jasper in that it exclusively uses RNN layers for both the encoder and decoder modules and decodes to a standard softmax probability distribution rather than a CTC distribution.

The system uses two SRL models to infer semantic annotations: SLING and BSAT, a biaffine syntax-agnostic tagger. SLING is a state-of-the-art semantic parser which uses an LSTM encoder and a TBRU decoder. With its TBRU layers, SLING is able to dynamically update an internal model state to make efficient parsing decisions. Uniquely, SLING outputs a sequence of transition commands to generate semantic frame parses. Alternatively, BSAT is unique in the way it represents predicate-argument data: its LSTM encoder generates two distinct vector representations for both predicates and arguments, and the decoder uses a biaffine scoring function to predict the most likely predicate-argument pairs.

The experimentation conducted in this thesis is driven by an extension to the OpenSeq2Seq toolkit: the `PostProcessingBlock` configuration. The new configuration block is supported in either evaluation or inference modes for OpenSeq2Seq and allows users to design custom logic for sequence processing downstream from the decoder. To support the full transformation pipeline, two types of postprocessors are defined in this thesis: the first is an LM processor which performs either n-best transcription rescoring or shallow fusion decoding, and the second is an SRL processor to generate predicate-argument labels for a transcription. The SRL processor has out-of-the-box support for SLING or BSAT.

Using these extensions, I describe two experiments. In the first experiment, I create and run an end-to-end transformation pipeline using combinations of the previously described ASR and SRL models. Outputs from the ASR component are rescored using a Transformer-XL LM and then given as input to the corresponding SRL model. The second experiment is a similar end-to-end transformation flow, except ASR outputs are generated using shallow fusion decoding with Transformer-XL. In the fusion decoder, log probabilities for the ASR model and LM are linearly interpolated to generate the best transcription sequence. Both experiments are jointly evaluated in terms of WER and F1 metrics.

The end-to-end results are indicative of a promising research direction. Further endeavours could explore more sophisticated system compositions, such as deep LM fusion or a joint multi-task model architecture for ASR and SRL. Additionally, various optimizations can be applied to the system to improve the end-to-end results, including more a robust LAS

training procedure and shallow fusion computation. There are implications for other NLP tasks as well: OpenSeq2Seq supports a number of sequence-to-sequence models, including machine translation and speech synthesis, and the `PostProcessingBlock` could be a conduit for research into system design for joint sequence-to-sequence problems in general.

Appendix A

DEEPSPEECH2 CONFIGURATION

This section describes a complete DeepSpeech2 configuration JSON along with the `PostProcessingBlock` definition for LM rescoring and SRL. The configuration is largely a derivative of the open-sourced pretrained DeepSpeech2 configuration from OpenSeq2Seq[1], but contains several distinct modifications.

The configuration is discussed in three parts: (i) the base hyperparameters, which include the encoder and decoder architecture definitions and training parameters, (ii) the training procedure recipe, and (iii) the evaluation procedure recipe.

A.1 Base Parameters

```
base_params = {
  "random_seed": 0,
  "use_horovod": True,                #auto-enable GPU-distributed training
  "num_gpus": 4,
  "batch_size_per_gpu": 32,
  "max_steps": 50000,                 #default 50000 steps, but train until convergence
  "save_summaries_steps": 100,
  "print_loss_steps": 100,
  "print_samples_steps": 1000,
  "eval_steps": 5000,
  "save_checkpoint_steps": 10000,
  "logdir": "/raid/exp/ds2",
```

[1]https://nvidia.github.io/OpenSeq2Seq/html/speech-recognition/deepspeech2.html

```
"optimizer": "Adam",
"lr_policy": poly_decay,


"lr_policy_params": {
  "learning_rate": 0.0001,
  "power": 0.5
},
"larc_params": {
  "larc_eta": 0.001,
},
"initializer": tf.contrib.layers.xavier_initializer,
"regularizer": tf.contrib.layers.l2_regularizer,
"regularizer_params": {
  'scale': 0.0005
},
"summaries": ['learning_rate', 'variables', 'gradients', 'larc_summaries',
              'variable_norm', 'gradient_norm', 'global_gradient_norm'],
"dtype": "mixed",                 #mixed precision available for some GPU types

"encoder": DeepSpeech2Encoder,
"encoder_params": {
  "conv_layers": [                # 2 layers of 2D convolutions
    {
      "kernel_size": [11, 41], "stride": [2, 2],
      "num_channels": 32, "padding": "SAME"
    },
    {
```

```
      "kernel_size": [11, 21], "stride": [1, 2],
      "num_channels": 32, "padding": "SAME"
    },
  ],
  "num_rnn_layers": 5,            # 5 GRU layers with CUDA acceleration
  "rnn_cell_dim": 512,
  "use_cudnn_rnn": True,
  "rnn_type": "cudnn_gru",
  "rnn_unidirectional": True,
  "row_conv": True,               # row convolutions enabled as in original model
  "row_conv_width": 4,
  "n_hidden": 1024,
  "dropout_keep_prob": 0.5,
  "activation_fn": tf.nn.relu,
},

"decoder": FullyConnectedCTCDecoder,
"decoder_params": {
  "use_language_model": False,
  "alphabet_config_path": "/raid/exp/ds2/vocab.txt",
},
"loss": CTCLoss,
"loss_params": {},

"data_layer": Speech2TextDataLayer,       # taken from the Jasper recipe
  "data_layer_params": {
      "num_audio_features": 64,
      "input_type": "logfbank",
```

```
        "vocab_file": "/raid/exp/ds2/vocab.txt",
        "norm_per_feature": True,
        "window": "hanning",
        "precompute_mel_basis": True,
        "sample_freq": 16000,
        "pad_to": 16,
        "dither": 1e-5,
        "backend": "librosa"
    },
}
```

This implementation of OpenSeq2Seq uses the Adam optimizer [52], a type of first-order stochastic gradient descent (SGD) optimizer with momentum. In the reference implementation of DeepSpeech2, the authors use SGD with Nesterov momentum ([2]; [83]), but I find that Adam converges faster and more consistently than the standard Momentum optimizer provided in Tensorflow.

This DeepSpeech2 encoder is effectively structured the same as the reference implementation, except it uses fewer CNN and RNN layers. OpenSeq2Seq supports CUDA-enabled GRU cells for accelerated RNN training.

Librosa [60] is the audio processing toolkit used to extract fbank features from the input signal.

A.2 Training Recipe

```
train_params = {
    "data_layer": Speech2TextDataLayer,
    "data_layer_params": {
        "augmentation": {
            'n_freq_mask': 2,
```

```
            'n_time_mask': 2,
            'width_freq_mask': 4,
            'width_time_mask': 4,
        },
        "dataset_files": [
            "/raid/data/librispeech-train-full.csv",
        ],

        "max_duration": 16.7,
        "shuffle": True,
    },
```

The training parameter `Speech2TextDayaLayer` inherits the data layer configuration defined in the `base_params` block. This recipe uses 64 log-mel filterbank features with a sample frequency of 16000 Hz.

A.3 Evaluation Recipe

```
eval_params = {
    "data_layer": Speech2TextDataLayer,
    "data_layer_params": {
        "dataset_files": [
            "/raid/data/librispeech-eval-full.csv",
        ],
        "shuffle": False,
    },
    "postprocessing_block": {
        "enable_joint_outputs": True,
        "processors": [
```

```
            {"processor": TransformerXLProcessor, "data_source_idx" -1},
            {"processor": SlingProcessor, "data_source_idx": 0}
        ]
    }
}
```

In the evaluation procedure, Transformer-XL and Sling are used as postprocessors. By default, shallow fusion is disabled and the LM processor performs n-best rescoring. The joint postprocessor outputs are automatically logged to a file in the working directory.

BIBLIOGRAPHY

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

[2] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Vaino Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 173–182. JMLR.org, 2016.

[3] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042, 2016.

[4] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. *CoRR*, abs/1508.04395, 2015.

[5] Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur, Yi Li, Hairong Liu, Sanjeev Satheesh, David Seetapun, Anuroop Sriram, and Zhenyao Zhu.

Exploring neural transducers for end-to-end speech recognition. *CoRR*, abs/1707.07413, 2017.

[6] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[7] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[8] Arianna Bisazza, Marco Dinarelli, Silvia Quarteroni, Sara Tonelli, Alessandro Moschitti, and Giuseppe Riccardi. Semantic annotations for conversational speech: From speech transcriptions to predicate argument structures. pages 65 – 68, 01 2009.

[9] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *CoRR*, abs/1611.01576, 2016.

[10] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, Jun 1998.

[11] Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. A full end-to-end semantic role labeler, syntax-agnostic over syntax-aware? *CoRR*, abs/1808.03815, 2018.

[12] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. Listen, attend and spell. *CoRR*, abs/1508.01211, 2015.

[13] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems: Recent advances and new frontiers. *CoRR*, abs/1711.01731, 2017.

[14] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274, 2015.

[15] 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. *CoRR*, abs/1712.01769, 2017.

[16] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[17] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[18] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, KyungHyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *CoRR*, abs/1506.07503, 2015.

[19] Jan Chorowski and Navdeep Jaitly. Towards better decoding and language model integration in sequence to sequence models. *CoRR*, abs/1612.02695, 2016.

[20] Stephane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina. On the use of BERT for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 108–117, Hong Kong, November 2019. Association for Computational Linguistics.

[21] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.

[22] G. E. Dahl, Dong Yu, Li Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Trans. Audio, Speech and Lang. Proc.*, 20(1):30–42, January 2012.

[23] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019.

[24] Renato De Mori. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, pages 9 – 40. 03 2011.

[25] Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734, 2016.

[26] Jesse Dunietz, Jaime Carbonell, and Lori Levin. DeepCx: A transition-based approach for shallow semantic parsing with complex constructional triggers. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1691–1701, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[27] Nadir Durrani, Fahim Dalvi, Hassan Sajjad, Yonatan Belinkov, and Preslav Nakov. One size does not fit all: Comparing NMT representations of different granularities. In *Proceedings of the 2019 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 1504–1516, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[28] Andrew Errity and John McKenna. An investigation of manifold learning for speech analysis. In *INTERSPEECH*, 2006.

[29] Farheen Fauziya and Geeta Nijhawan. A comparative study of phoneme recognition using gmm-hmm and ann based acoustic modeling. 2014.

[30] Benoît Favre, Bernd Bohnet, and Dilek Z. Hakkani-Tür. Evaluation of semantic role labeling and dependency parsing of automatic speech recognition output. *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5342–5345, 2010.

[31] Charles J. Fillmore. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1):20–32, 1976.

[32] John Foley, Sheikh Muhammad Sarwar, and James Allan. Named entity recognition with extremely limited data. *CoRR*, abs/1806.04411, 2018.

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

[34] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Comput. Linguist.*, 28(3):245–288, September 2002.

[35] Rahul Goel, Shachi Paul, Tagyoung Chung, Jérémie Lecomte, Arindam Mandal, and Dilek Hakkani-Tür. Flexible and scalable state tracking framework for goal-oriented dialogue systems. *CoRR*, abs/1811.12891, 2018.

[36] Alex Graves. *Connectionist Temporal Classification*, pages 61–93. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[37] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14. JMLR.org, 2014.

[38] Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *CoRR*, abs/1503.03535, 2015.

[39] Aria Haghighi, Kristina Toutanova, and Christopher D. Manning. A joint model for semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 173–176, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[40] Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[41] Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[42] Shexia He, Zuchao Li, Hai Zhao, and Hongxiao Bai. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2061–2071, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[43] Kenneth Heafield. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.

[44] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[45] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: The 90In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 57–60, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[46] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[47] Hirofumi Inaguma, Jaejin Cho, Murali Karthick Baskar, Tatsuya Kawahara, and Shinji Watanabe. Transfer learning of language-independent end-to-end ASR with language model fusion. *CoRR*, abs/1811.02134, 2018.

[48] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[49] Rukmini Iyer and Mari Ostendorf. Relevance weighting for combining multi-domain data for n-gram language modeling. *Computer Speech Language*, 13:267–282, 1999.

[50] Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara Sainath, Zhifeng Chen, and Rohit Prabhavalkar. An analysis of incorporating an external language model into a sequence-to-sequence model. 12 2017.

[51] Jungo Kasai, Dan Friedman, Robert Frank, Dragomir R. Radev, and Owen Rambow. Syntax-aware neural semantic role labeling with supertags. *CoRR*, abs/1903.05260, 2019.

[52] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[53] Lingpeng Kong, Chris Alberti, Daniel Andor, Ivan Bogatyy, and David Weiss. DRAGNN: A transition-based framework for dynamically connected neural networks. *CoRR*, abs/1703.04474, 2017.

[54] Oleksii Kuchaiev, Boris Ginsburg, Igor Gitman, Vitaly Lavrukhin, Carl Case, and Paulius Micikevicius. OpenSeq2Seq: Extensible toolkit for distributed and mixed precision training of sequence-to-sequence models. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 41–46, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[55] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360, 2016.

[56] Tung D. Le, Haruki Imai, Yasushi Negishi, and Kiyokuni Kawachiya. TFLMS: large model support in tensorflow by graph rewriting. *CoRR*, abs/1807.02037, 2018.

[57] Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M. Cohen, Huyen Nguyen, and Ravi Teja Gadde. Jasper: An end-to-end convolutional neural acoustic model, 2019.

[58] Longfei Li, Yong Zhao, Dongmei Jiang, Yanning Zhang, Fengna Wang, Isabel Gonzalez, Enescu Valentin, and Hichem Sahli. Hybrid deep neural network - hidden markov model (dnn-hmm) based speech emotion recognition. pages 312–317, 09 2013.

[59] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.

[60] Brian McFee, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Dana Lee, Oriol Nieto, Eric Battenberg, Dan Ellis, Ryuichi Yamamoto, Josh Moore, Rachel Bittner, Keunwoo Choi, Pius Friesch, Fabian-Robert Stöter, Vincent Lostanlen, Siddhartha Kumar, Simon Waloschek, Seth Kranzler, Rimvydas Naktinis, Douglas Repetto, Curtis "Fjord" Hawthorne, CJ Carr, Waldir Pimenta, Petr Viktorin, Paul Brossier, Jo ao Felipe Santos, Jackie Wu, Erik Peterson, and Adrian Holovaty. librosa/librosa: 0.6.1, May 2018.

[61] Gertjan Van Noord and Jean-Claude Junqua. *Robustness in Language and Speech Technology (Text, Speech and Language Technology)*. Springer-Verlag, Berlin, Heidelberg, 2001.

[62] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

[63] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An asr corpus based on public domain audio books. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

[64] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin Dogus Cubuk, and Quoc V. Le. Specaugment: A simple augmentation method for automatic speech recognition. In *INTERSPEECH*, 2019.

[65] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[66] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[67] Douglas B. Paul and Janet M. Baker. The design for the wall street journal-based csr corpus. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, pages 357–362, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[68] D. Povey, A. Ghoshal, et al. The Kaldi Speech Recognition Toolkit. In *Proc. ASRU*, 2011.

[69] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[70] Sameer S. Pradhan, Wayne H. Ward, Kadri Hacioglu, James H. Martin, and Dan Jurafsky. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 233–240, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[71] Golan Pundak, Tara N. Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao. Deep context: End-to-end contextual speech recognition. *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 418–425, 2018.

[72] Pushpendre Rastogi, Arpit Gupta, Tongfei Chen, and Lambert Mathias. Scaling multi-domain dialogue state tracking via query reformulation. *CoRR*, abs/1903.05164, 2019.

[73] Michael Ringgaard, Rahul Gupta, and Fernando C. N. Pereira. SLING: A framework for frame semantic parsing. *CoRR*, abs/1710.07032, 2017.

[74] George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, Bergul Roomi, and Phil Hall. English conversational telephone speech recognition by humans and machines. *CoRR*, abs/1703.02136, 2017.

[75] Hiroshi Seki, Takaaki Hori, Shinji Watanabe, Jonathan Le Roux, and John R. Hershey. A purely end-to-end system for multi-speaker speech recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2620–2630, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[76] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[77] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *CoRR*, abs/1802.05799, 2018.

[78] Pamela Shapiro and Kevin Duh. BPE and charcnns for translation of morphology: A cross-lingual comparison and analysis. *CoRR*, abs/1809.01301, 2018.

[79] Niraj Shrestha, Ivan Vulić, and Marie-Francine Moens. Semantic role labeling of speech transcripts. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 583–595, Cham, 2015. Springer International Publishing.

[80] Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. Cold fusion: Training seq2seq models together with language models. *CoRR*, abs/1708.06426, 2017.

[81] Felix Stahlberg, James Cross, and Veselin Stoyanov. Simple fusion: Return of the language model. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 204–211, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[82] Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[83] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1139–III–1147. JMLR.org, 2013.

[84] Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. Deep semantic role labeling with self-attention. *CoRR*, abs/1712.01586, 2017.

[85] Shubham Toshniwal, Anjuli Kannan, Chung-Cheng Chiu, Yonghui Wu, Tara Sainath, and Karen Livescu. A comparison of techniques for language model integration in encoder-decoder speech recognition. 07 2018.

[86] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[87] Alexander Waibel, Toshiyuki Hanazawa, Geofrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Readings in speech recognition. chapter Phoneme Recognition Using Time-delay Neural Networks, pages 393–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[88] Yufei Wang, Mark Johnson, Stephen Wan, Yifang Sun, and Wei Wang. How to best use syntax in semantic role labelling. *CoRR*, abs/1906.00266, 2019.

[89] Puyang Xu and Qi Hu. An end-to-end approach for handling unknown slot values in dialogue state tracking. *CoRR*, abs/1805.01555, 2018.

[90] Nianwen Xue and Martha Palmer. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[91] Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. Neural generative question answering. *CoRR*, abs/1512.01337, 2015.

[92] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer Publishing Company, Incorporated, 2014.

[93] Kai Yu, Lu Chen, Kai Sun, Qizhe Xie, and Su Zhu. Evolvable dialogue state tracking for statistical dialogue management. *Frontiers of Computer Science*, 10(2):201–215, Apr 2016.

[94] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. A simple but hard-to-beat baseline for session-based recommendations. *CoRR*, abs/1808.05163, 2018.

[95] Albert Zeyer, Kazuki Irie, Ralf Schlüter, and Hermann Ney. Improved training of end-to-end attention models for speech recognition. *CoRR*, abs/1805.03294, 2018.