Literature Survey : Spoken Language Translation

Sanket Gandhare
Preethi Jyothi
Pushpak Bhattacharyya
IIT Bombay, India
sanketmonu@cse.iitb.ac.in

Abstract

Spoken language translation system has received lot of attention in last decade or so. It enables in translation of speech signals in a source language A to text in target language B. This problem mainly deals with Machine translation (MT), Automatic Speech Recognition (ASR) and Machine Learning (ML). The spoken utterances are first recognized and converted to text and later this source language text is translated to target language. In this paper, we start with looking into the whole flow of speech translation by going via Automatic Speech Recognition and its techniques and neural machine translation. We study the coupling of speech recognition system and the machine translation system by doing the hypothesis selection and features for it by taking the output of the ASR system and fetching it to the MT system. We also look into the lattice and Confusion Network Decoding.

1 Introduction

Spoken language translation is the process by which conversational spoken phrases are converted to second language. This enables the speakers of different languages to communicate. The speech translation system integrates two technologies : **Automatic Speech Recognition, Machine Translation**. The speaker of language A speaks and the speech recognizer recognizes the utterance. The input is then converted into a string of words, using dictionary and grammar of language A, by using the massive corpus of text of language A. The machine translation part takes care of translating the text into another language. In this paper, we will first look into the ASR approaches, NMT approaches and the coupling of the systems.

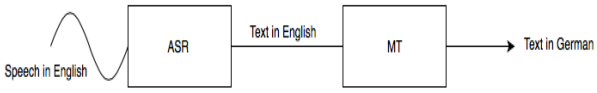

Figure 1: Spoken Language Translation

2 Automatic Speech Recognition

Speech recognition is an inter-disciplinary field of computational linguistics that develops method for recognition and translation of spoken language to text. Speech recognition applications include voice user interfaces such as voice dialing, call routing, simple data entry, preparation of structured documents, speech-to-text processing. Some Sr systems use ”training” where individual speaker reads text or isolated vocabulary into the system. The system analyzes the person’s specific voice and uses it to fine-tune the recognition of that person’s speech, resulting in increased accuracy. Such SR systems that use the training are called **speaker dependent** else it is called **speaker independent** systems. The term *speaker identification* refers to identifying the speaker, rather than what they are saying. The voice of any person can be translated and stored as data on which we can train person’s voice and it will be useful for speaker identification, helpful for security purposes

2.1 Approaches

ASR uses mainly two models **acoustic model** and **language model** to recognize the speech. Acoustic model gives a relationship between the phonemes and the audio signals. Language model gives idea of identifying correct words from the data looking at the context as well. There are few methods like HMM, Neural networks etc that are used in speech recognition. Let us look into them in brief :

2.1.1 HMM

HMMs are mostly used in speaker recognition today. We get the output sequence of symbols from these models. HMMs are used in speech recognition because the audio signal can be considered piece wise stationary signal.

Modern general-purpose speech recognition systems are based on Hidden Markov Models. These are statistical models that output a sequence of symbols or quantities. HMMs are used in speech recognition because a speech signal can be viewed as a piece wise stationary signal or a short-time stationary signal. In a short timescale (e.g., 10 milliseconds), speech can be approximated as a stationary process. Speech can be thought of as a Markov model for many stochastic purposes. HMMs can also be trained automatically and are simple computationally. In ASR, the HMM outputs real-valued vectors that consists of cepstral coefficients that are obtained by taking the fourier transform of speech and taking the most important coefficients. Each word, or more particularly each phoneme has different probability distribution and we combine lot of such HMMs for different words to get the phoneme.

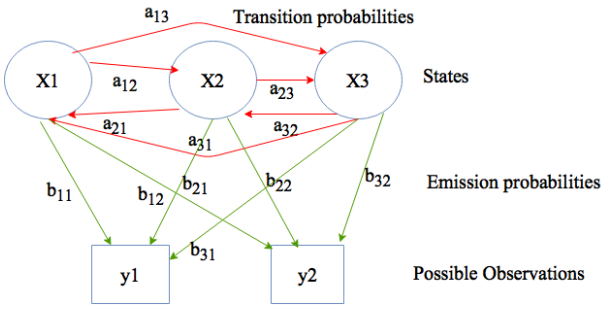

Figure 2: Example of parameters of HMM

The **Context model** is used to convert the tri-phones into the mono-phones. The **Pronunciation model** is used to combine the phones to get the words. The words are combined using n-gram **language models** to get the sentences. **Decoding** of the speech is the terms used to get the most likely word sequence when the system is presented with a new utterance. This uses **Viterbi algorithm** to find the best path and it uses both acoustic and language model information in the form of Finite State transducers. Efficient algorithms have been developed that gives good decoding results.

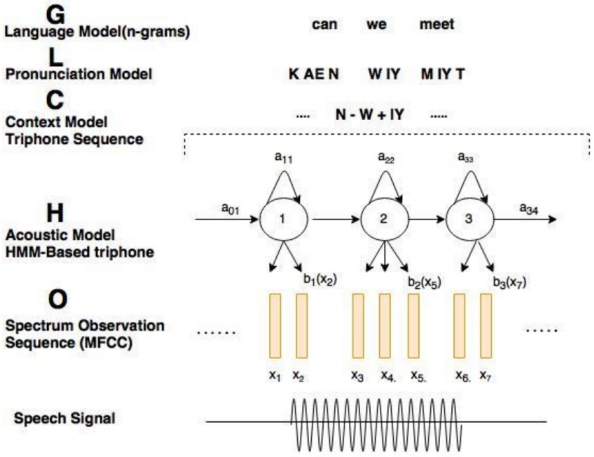

Figure 3: The full ASR steps

2.1.2 Neural Networks

Neural Networks has come up as nice approach for acoustic modelling in ASR since 1980s. In contract to HMMs, neural neftworks does not make any assumptions regarding the statistical properties and have several qualities that make them great model for speech recognition. Neural networks allow discriminative training in an efficient manner. The input features for the neural networks needs to be assumed. However, neural networks give good results on short time units for individual phones and isolated words, they are not useful in continuous recognition tasks because of their lack of ability to model temporal dependencies.

Recently, **LSTM RNN (Recurrent Neural Networks)** have been developed which use the temporal dependencies and use this information in speech recognition. Due to the inability of feed forward Neural Networks to model temporal dependencies, another approach is to use neural networks as a pre - processor using feature transformation.

A **deep neural network (DNN)** is an artificial neural network with multiple hidden layers of units between the input and output layers. DNN can use complex non-linear functions and the extra layers enable composition of features from lower layers increasing the learning capacity.

2.1.3 End-to-End ASR

This is the latest hot topic of research in ASR. Traditional phonetic-based like HMM based model approaches required separate components and training for the pronunciation, acoustic and language model. End-to-end models jointly learn all

the components of the speech recognizer. This eases in the deployment of the product at a go. Also the n-gram models take several gigabytes of memory while deploying. One of the example of end-to-end ASR system was **Connectionist Temporal Classification (CTC)** and it was first developed using RNN model. It takes care of both pronunciation and acoustic model together. An alternative approach to CTC-based models are **attention-based** models.

2.2 Time Delay Neural Network

Recurrent Neural Networks have efficiently modeled long time in comparison to feed forward neural networks due to it's sequential nature. Discussed is the TDNN architecture to achieve this task. It uses sub-sampling to reduce the training time.(Povey et al., 2011)

2.2.1 TDNN Architecture

It is a feed forward neural network, where the initial transform layer captures the dependencies in the narrow context and the deeper layers captures for the more wider context. The hyper parameters are to decide the input contexts for each layer. (Povey et al., 2011)

2.2.2 Sub-sampling

The computing of the hidden units at all the time stamps is considered as the expensive task. With nearby layers sharing same temporal context, computing all the activation units can be avoided and only a certain number can be calculated. In the figure below, the red units are calculated because of sub-sampling. Image taken from [1]

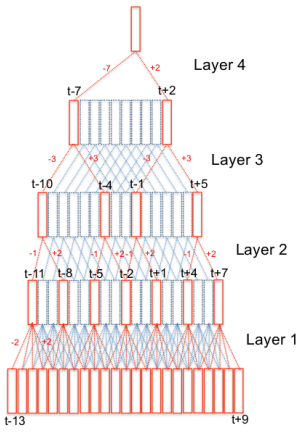

Figure 4: TDNN Architecture

3 Neural Machine Translation

Neural machine translation is end-to-end translation process for automated translation and is designed to remove all the weaknesses that was because of the phrase based machine translation(Chu et al., 2017). NMT is an asset as it has ability to learn directly, as end-to-end sequence and mapping the input sequence to the output sequence. NMT generally consists of two RNNs, with one RNN taking input text sequence and the other one giving the output sequence. NMT can be made efficient by making use of attention.

3.1 Encoder-Decoder Approach

This approach uses the Recurrent Neural Networks. RNNs can model the language as a sequential process. This model can predict the next word. Below figure will explain the flow(Chu et al., 2017) :

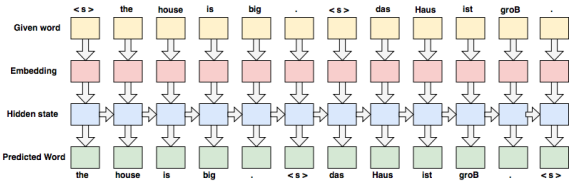

Figure 5: Encoder-Decoder Model

For training such a model, input and output sentences are concatenated which similar to the language model method. While decoding, input sentence is fed, and then the words are predict till the end of the sentence token is detected. Once end of the input sentence is processed, the hidden states encodes its meaning. The vector that holds the values of the nodes in its final hidden layer is called the **input sentence embedding**.(Chu et al., 2017) This is called as the **encoder** phase of the model. The hidden state is used to produce the respective translation in the **decoder** phase. Here, we notice that the hidden layer needs to do lot of work like storing the whole sentence from start to the end of the sentence. Also, during the decoding it not only has to find the next translated word in the sequence but also needs to take into account what part of the sentence has been translated and what needs to be covered yet. This model works well on smaller sentences but struggle on bigger sentences and hence we need to look into it in depth.(Chu et al., 2017)

[1] http://speak.clsp.jhu.edu/uploads/publications/papers/1048_pdf

3.1.1 Encoder

Encoder defines the representation of the input sentence. The input sentence which is the sequence of words, is the embedding matrix. These words are processed with recurrent neural network. This gives hidden states which encodes each word as its left context, with preceding words. For getting the right context we encode from right to left, or end of the sentence to the beginning. Running two neural networks in different directions is called a **bidirectional recurrent neural network**.(Chu et al., 2017) The encoder has the embedding that maps to each input word x_j, and this mapping is through two hidden states $\overleftarrow{h_j}$ $\overrightarrow{h_j}$

$$\overleftarrow{h_j} = f(\overleftarrow{h_{j+1}}, Ex_j) \quad (1)$$

$$\overrightarrow{h_j} = f(\overrightarrow{h_{j-1}}, Ex_j) \quad (2)$$

Image taken from : [2]

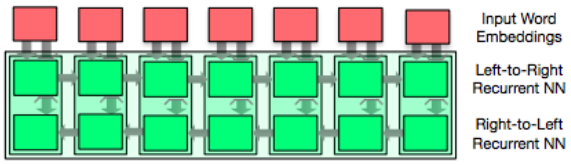

Figure 6: Input Encoder

3.1.2 Decoder

This is also an RNN. It takes some input context and the previous hidden state, output prediction and it generates new hidden and output state. Mathematically, this can be defined as(Chu et al., 2017) :

$$s_i = f(s_{i-1}, E_{y_i-1}, c_i) \quad (3)$$

where s_i is a sequence of hidden states, that are calculated from the previous hidden state $si-1$, the embedding of the previous output word E_{y_i-1} and input context c_i.

There are various choices for the function f that joins the inputs to get the next hidden state, linear function, LSTMs or GRUs. The choice for encoder and decoder is generally kept same. If encoder is LSTM, then the decoder should also be LSTM.

Image taken from : [3]

[2]https://arxiv.org/pdf/1709.07809.pdf

[3]https://arxiv.org/pdf/1709.07809.pdf

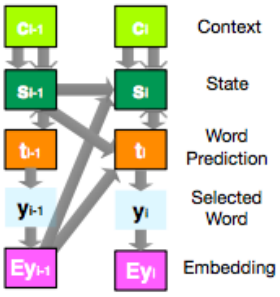

Figure 7: Output Decoder

The prediction vector t_i is conditioned on decoder hidden state s_{i-1}, embedding of previous output word E_{y_i-1} and input context c_i(Chu et al., 2017).

$$t_i = softmax(W(Us_{i-1} + VE_{y_i-1} + Cc_i)) \quad (4)$$

The softmax function is used to convert the raw vector in the probability distribution, that sums all values to 1. The highest value among the vectors gives us the output word. During the training the, we know the correct output word y_i so that the training can continue with that output word. Our objective of the training is to give as much probability mass to the correct word as possible. The cost function hence is the negative log of the probability given to the correct word(Chu et al., 2017):

$$cost = -logt_i[y_i] \quad (5)$$

In the ideal scenario, we want the probability to the correct word to get 1 and hence the negative log likelihood of 0 but its close to 0. The cost function is tied to the individual words, for the sentence the function is calculated as the sum for the individual words.

3.1.3 Attention Mechanism

The need of attention mechanism is to find a way to associate the decoder state and the every input word. Based on this result we can find the input words that how important are these to get the output words or we can set the weights accordingly. This is calculated as(Chu et al., 2017):

$$a(s_{i-1}, h_j) = w^{aT} s_{i-1} + u^{aT} h_j + b^a \quad (6)$$

where w_a and u_a are weights and b^a is bias value. The output is a value which shows the importance of input word j to get the output word i. This attention value is normalized by using softmax so that all these values sum to 1.

$$\alpha_{ij} = \frac{exp(a(s_{i-1}), h_j)}{\sum_k exp(a(s_{i-1}), h_k)} \quad (7)$$

This normalized attention value is used to get the context vector c_i by getting the input and multiplying it to the attention :

$$\alpha_{ij} = \frac{exp(a(s_{i-1}), h_j)}{\sum_k exp(a(s_{i-1}), h_k)} \quad (8)$$

Image taken from : [4]

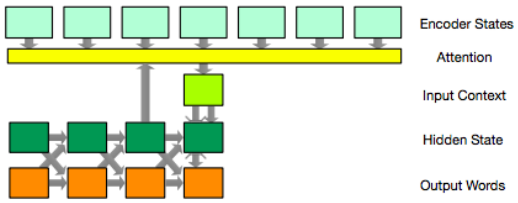

Figure 8: Attention Mechanism

3.1.4 Training

The challenge with training is the number of steps in the encoder and in the decoder varies as the training set changes. Also, the sentence length for every pair of the sentences change for each training example, so we calculate the computation graph for each training example. This is called **unrolling** technique the recurrent neural network(Chu et al., 2017). The error that we are computing is for the entire sentence which is calculated as the sum of all the words. For predicting the next word, correct word is used as context for decoder hidden state and the word prediction. The training objective depends on the probability mass given to the correct word, given that the context is perfect. For training of the NMT model requires GPUs, which gives high level of parallelism to the deep models. The speed can also be increased by giving lot of sentence pairs at a go (say 50). The GPUs ensure that the attention is used in a proper way as the attention mechanism is passed to the GPU with a matrix of encoder states s_{i-1} and input encodings h_j, resulting in values for the attention. Since the weights for each of these factors are reused, the power of GPUs in parallism comes into picture. Overall, training consists of, shuffle the training corpus, break up corpus into mini batches, process each of these mini-batches and gather the gradients, apply gradients to update the parameters. Generally, training takes 5-15 epochs i.e. it passes through the whole training corpus. The stopping criteria for the training is to verify the progress on the validation set and it it stops increasing we can stop training.
Image taken from : [5]

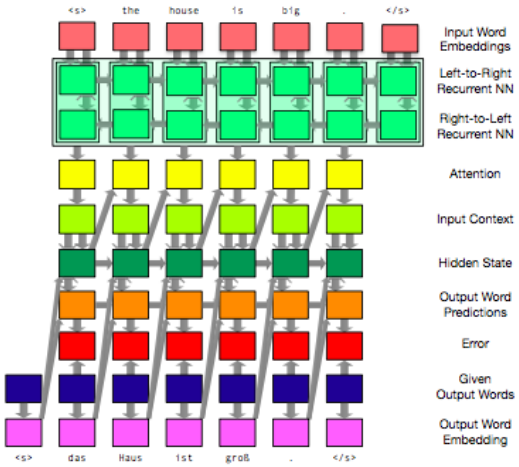

Figure 9: Unrolling RNN for training

3.1.5 Beam Search

The translation via a neural translation models happens with one word at a time. Every step the output word is calculated as probability distribution is calculated over all words. The word which is most likely is picked and the next word is selected based on the context. When we predict the first word of the output sentence, we keep track of the n-best words which are the likely choices. This is called the **beam search**.(Chu et al., 2017) All the words that are selected in the beam can be used to get the context for the next word. Hence we get different word predictions for each word. The score for the partial translation is multiplied with the probabilities of the predictions of the word. The score for the words which is maximum is selected for the next word prediction. We get the score for every such hypotheses. When we get the final end of the token we consider that the translation is done for the sentence. When no hypothesis is present in the beam we consider that the search is terminated. For any sentence, search starts with the start symbol <s>and ends with the <\s>. The hypothesis for which the score is the highest and which ends appropriately is considered as the actual output. The difference in the decoding of the

[4] https://arxiv.org/pdf/1709.07809.pdf

[5] https://arxiv.org/pdf/1709.07809.pdf

neural translation as compared to that of statistical translation is that we combine the hypothesis if they share the context and hence this will be used for predicting future translations. In the case of the RNNs, we condition on the overall word sequence from the start.(Chu et al., 2017)

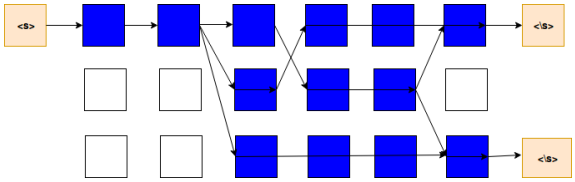

Figure 10: Simple search graph for a beam search decoding in the neural models.

3.2 Improvements

Till now we have seen the basic neural machine translation model. Though this performs fairly well for most of the language pairs the scores can be improved by few techniques.

3.2.1 Ensemble Decoding

This is the technique where we train lots of systems and combine them. This is called **ensemble systems**(Chu et al., 2017) This is very common technique and has resulted in great results. Few general ways can have different initialization points or different stopping points. The advantage of this technique is in a way that two systems can do same mistake but when combined not necessary that is repeated. There are two ways to do ensemble encoding - by generating alternate systems and combining outputs.

Generating Alternate System - Generally NMT is trained on a data until we reach a stopping criteria i.e where there is no improvement in the validation data accuracy. While the training is in progress we have models after few iterations. We take the best few models from this. This is called as **checkpoint ensembling**(Chu et al., 2017) since models are selected at different checkpoints while training is taking place.

Combining Outputs - Here we try to combine various system outputs. Any model first predicts predicts the output probability distribution over the output words and then these are averaged out to get the output word.

3.2.2 Large Vocabularies

With the vocabulary increasing day by day newer words come frequently and neural translation model is not able to find these words in its dictionary. To remove this words are converted to continuous vector space. But still it is not able to solve the problem. Two approaches are used for these :

1. The most common approach is to divide the words into **sub-word** units. This is done to handle the compound words (*website* = *web* + *site*) and morphology (*doing* = *do* + *ing*)(Chu et al., 2017).

2. Another popular method is **Byte Pair Encoding**. This works on parallel sentences. All the words are first divided into characters. After this the most frequent characters are joined. This is continued and hence the vocabulary size increases. After this we will have few word tokens and few un-merged words.

3.2.3 Adaptation

Most of the data that we find is from different domains. For example the texts that are used in the chat rooms are different than those used in the parliaments. The problem in the Neural machine translation can be that it is trained on the data that is not at relevant to the user and hence not getting correct translations. This problem is called as **Domain Adaptation**.(Chu et al., 2017) We have one set of the data that is called as the **in-domain** data that is relevant to the usecase and also data that is called as the **out-domain** data that is the general data and is not relevant to the usecase. The training is such cases is simple and is done in two steps. First we train the model on the all the data we have till the results on the validation data converges. Then we will run few more iterations on the in-domain data till the validation set converges. This is how the final model takes the benefit of both the in-domain data as well as the whole data.(Chu et al., 2017)

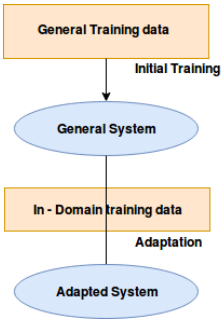

Figure 11: Domain Adaptation method

3.2.4 Adding Linguistic Annotation

The advantages of the neural network is that it can consider the rich context. In the neural networks the present output word is predicted based on the previously generated outputs and entire input sentence. Even if we find that the input or the partial output sequence is not observed the neural network is able to generalize the training data to get the output for the unseen word. So, we can add more information to the conditioning context in the neural network like part of speech tags, morphological words and lemmas.

3.3 Challenges

While neural machine translation systems give good results after the refinements, it still has few challenges. How does it perform in the out of domain data and if the language resource is less are two prominent challenges. Also, neural translation models are less interpretable. Lot of things happens in numbers that we wont be able to understand what is happening. Let us look into few such challenges in detail.

3.3.1 Domain Mismatch

One of the known challenges in the neural models is that of domain mismatch that is if the model is trained on one data and we are testing on data of different domain, neural models can give output that is not at all related to the actual output. SMT in such cases doesn't translate such words. Here is the example of the sentence from the subtitles corpus trained on various corparas. Image[6] clarifies it. It can be clearly seen in the image that neural translation model results are completely random and don't make any sense.

Source	*Schaue um dich herum.*
Reference	*Look around you.*
All	NMT: *Look around you.* SMT: *Look around you.*
Law	NMT: *Sughum gravecorn.* SMT: *In order to implement dich Schaue .*
Medical	NMT: *EMEA / MB / 049 / 01-EN-Final Work progamme for 2002* SMT: *Schaue by dich around .*
IT	NMT: *Switches to paused.* SMT: *To Schaue by itself . \t \t*
Koran	NMT: *Take heed of your own souls.* SMT: *And you see.*
Subtitles	NMT: *Look around you.* SMT: *Look around you .*

Figure 12: Output of subtitle data example when models trained on different corpara

[6]https://arxiv.org/pdf/1709.07809.pdf

3.3.2 Amount of Training Data

One of the problem that has been observed with the neural models is its dependence on the data. The BLEU scores rises significantly from the model that is trained on lesser words to the model trained on huge number of words. The curve is so steep that a particular point of time it also crosses the curve of the statistical machine translation model.

3.3.3 Noisy Data

Neural translation models results deteriorate when the data is noisy as compared to the statistical models. The translation can be affected in various ways - due to mis-aligned sentences, data in wrong language, incorrect sentences. Statistical machine translation models look into probability distributions from lots of occurrences of words and phrases, so the noise doesn't affect it much. The neural models perform poor because while predicting it has to find proper balance between language models and the input.(Chu et al., 2017) When it finds input sentences with lots of distractions in the meanings the probability distribution is set accordingly.

3.3.4 Beam Search

Decoding in neural models is done using beam search and it finds the full sentence with highest probability. In the neural models while predicting the next output word, we will not only concentrate on the highest scoring words but also the few best partial translations. We take the score of each partial translation and the subsequent word predictions and add these scores.(Chu et al., 2017) With every output word the the number of translations explode exponentially, hence we prune the beam size to the few best translations. Increasing the beam size doesn't necessarily increase the translational quality as it decreases after a certain value. So we need to find such optimal beam size for the language pair.

4 Coupling of ASR and MT

Speech translation is conventionally carried out by cascading an **Automatic Speech Recognition System** and **Machine Translation system**. Generally the factors that are optimized are the language models and the acoustic models alongwith the **word error rate** for the ASR system and the **BLEU** score for the MT system. The process of spoken translation is a three step pipeline. Step

one involves transcribing the speech to the text format using ASR system. Step two ensures that the output from the ASR is in the format in which the MT system expects the input to be in. The third step includes the MT part which translates the text from source language to the target language. The basic case will be to use the ASR 1-best output that can be used as an input by the MT system. The other output options from the ASR system that can be fed to the MT system are **N-best** or **lattices** and **confusion networks**. These can be useful for the tuning and decoding in the MT system, however, it increases the complexity because of the number of alternatives present are exponential(Kumar et al., 2015)

4.1 Coarse-To-Fine Speech Translation

In this section, we describe the featurized model for hypothesis selection that uses information from the ASR and MT systems. We are assuming that the ASR and MT systems are trained separately. We are trying to find out a pipeline between these two where we take the advantages from the local gains we get from each of the systems. We will try to introduce the formal machinery that can be used for this.
Let $\sum$ and τ be alphabets of words and phrases respectively in the source language(Kumar et al., 2015). We will use these to define these state machines:

1. **Word Lattice (L)** : It is a finite state acceptor which accepts word sequence in the source language. It can be represented as(Kumar et al., 2015) :

$$L : \sum^{*} \rightarrow \sum^{*} \tag{9}$$

This basically represents the ASR word lattice output for us.

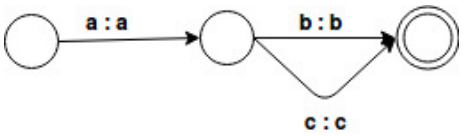

Figure 13: An unweighted Word Lattice

2. **Phrase segmentation Transducer (S)** : This finite state transducer that transduces sequence of words to phrases. It can be represented as(Kumar et al., 2015) :

$$S : \sum^{*} \rightarrow \tau^{*} \tag{10}$$

We traverse path over here by taking words and converting them into phrases.

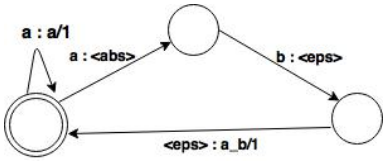

Figure 14: A phrase segmentation transducer which transduces words to phrases and has weights on its paths

3. **Weighted word lattice ($\hat{L}_{ASR}$)** : The weighted version of lattice and is represented as(Kumar et al., 2015) :

$$\hat{L}_{ASR} : \sum^{*} \rightarrow \sum^{*} / R^{+} \tag{11}$$

4. **Phrase Acceptor ($\hat{W}_{MT}$)** : It is a finite state acceptor that accepts the source phrases in the SMT system's phrase table. It can be represented as(Kumar et al., 2015) :

$$\hat{W}_{MT} : \tau^{*} \rightarrow \tau^{*} / R^{+} \tag{12}$$

5. **Phrase lattice (P)** : These are the composition of the word lattice with the phrase segmentation transducer. It represents phrase segmentation of all the ASR hypothesis in the word lattice.

$$P = det(min(LoS)) \tag{13}$$

The weighted version of P can be represented as $\hat{P}_{ASR/MT}$ where the ASR/MT parts denote the origin of the weights.

4.2 Maximum Spanning Phrases Model

In SMT phrase base translation, the translation is produced for each possible span of input sentence as allowed by the phrase table.(Kumar et al., 2015) If the **phrases** are **longer**, it gives **lesser options** and the translation is reliable as there may be sufficient occurrences of the phrases in the model. So, the longer the phrases, better the translation. We want the **ASR hypothesis** that requires least number of hypothesis to cover. We use the phrase lattice which is the composition of word lattice and

phrase segmentation transducer for achieving this. The phrase lattice use the weights from the phrase segmentation transducer and these weights are the number of phrases used to cover the path. The shortest path will give us the phrase we were looking for. Thus, this feature of SMT of source path length can be used for the phrase selection.

4.3 Featurized model for hypothesis selection

We will now try to build a model which does the hypothesis selection using the features from both the ASR and MT system. This gives us the hypothesis from the unpruned ASR lattice for which the MT system is supposed to give good translations. Giving importance to the weights of the ASR system also ensures that we are considering those hypothesis which the ASR system considers good. The phrase lattice for the ASR system only will be(Kumar et al., 2015) :

$$\hat{P}_{ASR} = det(min(\hat{L}_{ASR} o S)) \tag{14}$$

Next we will use the weighted phrase acceptor for the SMT for using its features :

$$\hat{P}_{ASR,MT} = det(min(\hat{P}_{ASR} o \hat{W}_{MT})) \tag{15}$$

The edge weights are considered to be in log semiring. Hence, these edge weights can be represented as(Kumar et al., 2015):

$$\omega(\rho^{(\hat{P}_{ASR,MT})}) = \sum_j \beta_j f_{j,ASR} + \sum_k \gamma_k f_{k,MT} \tag{16}$$

where the $\rho^{(\hat{P}_{ASR,MT})}$ represents the edge in the phrase lattice. β and γ are feature weights and f_{ASR} and f_{MT} are the feature weights for the ASR and the MT system respectively. We are considering the log linear space here. This model gives us idea which hypothesis should be used as 1-Best, N-Best and the lattice input for the SMT system from $\hat{P}_{ASR,MT}$.

4.4 Training and Feature selection

The training of the hypothesis selection is based on the standard methods of log linear model on the held-out set. For this we decode the **N-Best** derived from the held out set. Our main **objective** is to **maximize the translation quality** on the basis on some sentence level scores. Each time we get the translation, it can be compared to the N-Best and whenever the weights are updated it tells us how much importance needs to be given to the ASR and MT. The following features can be used for the model (Kumar et al., 2015):

1. **ASR Scores** : We combine the ASR acoustic model and the language model scores as the combined feature.

$$f_{ASR} = LM + \alpha AM \tag{17}$$

where the AM and the LM are the negative log probabilities and α is the acoustic scaling factor choosen to minimize the word error rate.

2. **Source Phrase Count** : This feature gives the intuition that using the fewer number of phrases for covering the input sentence will give better result.

3. **Length normalized phrase unigram probability** : We can use a phrase Language Model feature using the n-gram probabilities normalized by the length.

$$f_{uni}(f_j) = \left[\frac{count(f_j)}{\sum_k count(f_k)}\right]^{len(f+i)} \tag{18}$$

4. **Phrase Translation Entropy** : For each of the source phrase p_j there can be many translations e_i with different translation probabilities ($P(e_i/f_j)$). A simple metric to get the correct translation will be to use the entropy measure to get the confidence of which translation is the best for the SMT.

$$H_{tr}(E|p_j) = -\sum_i p_{tr}(e_i|f_j) log(p_{tr}(e_i|f_j)) \tag{19}$$

4.5 Decoding and related techniques

1. **Decoding** : If we replace the source side phrase acceptor to the target side acceptor we can get the output in the target language.(Kumar et al., 2015) The issue with this model is that it does not consider the reordering issue, the decoder is good enough to take decision on selecting the hypothesis that gives better result for the translation.

2. **Lattice Decoding** : This method which is broadly discussed in the next chapter gives better idea about how lattice can used for the decoding by the SMT system.

5 Spoken Translation using Word Lattice Decoding

We know that the Statistical MT uses the noisy channel model and translates the source language to the target language. It takes the hypothesis f as input and gives e as the output as per :

$$\hat{e} = argmax_e Pr(e|f) \tag{20}$$

When we try **spoken translation** where the input from the ASR system is converted into text and given as input to the MT system it may be the case that the ASR system does not give much importance to the translation. Hence instead of looking at the single best transcription f for getting the translation it is advantageous to consider all possibilities of the source sentence and hence that gives rise to using the **lattice** for decoding.(Dyer et al., 2008) The advantage also goes beyond the spoken translation as few languages have various issues that need to be taken care. **Segmentation** in Chinese, **decompounding** in German, **morphological analysis** for Arabic and other ambiguities in the input source sentence gives rise to multiple possibilities for the source word sequence.(Dyer et al., 2008)

5.1 Word Lattice and its representation

A word lattice G can be defined as :

$$G = (V, E)$$

where V are the vertices and E are the edges in the lattice. The word lattice is the **directed acyclic graph** that can be formally weighted finite state automate (FSA). We have a start node as well as the end node. There is no outgoing edge from the end node. The lattice can also be defined as partially ordered set of **poset** that follow the properties of reflexivity, transitivity and anti-symmetric as we move in the forward direction in the lattice. The word lattices are useful because it gives us set of strings to be represented and allows sub strings to be represented using a common structure where multiple strings can have a common member. For translation we will try to encode the graph in **topological ordering** and hence the numbering is in increasing order. The representation is chart based and is a triple of 2-dimensional matrices (F, p, R) as(Dyer et al., 2008) :

- $F_{i,j}$: Word label of j^{th} transition from node i
- $p_{i,j}$: Transition cost or probability

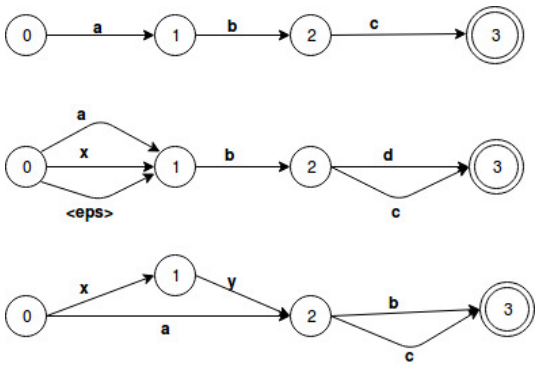

Figure 15: Three examples of lattice a) sentence b) Confusion network c) general word lattice

- $R_{i,j}$: The node number of the node on the right side of j^{th} transition from node i. $R_{i,j} > i$ for all i, j.

For the above 3 lattices, they can be represented as:

0	1	2
a 1 1	b 1 2	c 1 3
a 1/3 1 x 1/3 1 <eps> 1/3 1	b 1 2	c 1/2 3 d 1/2 3
a 1/2 2 x 1/2 1	y 1 2	c 1/2 3 b 1/2 3

Figure 16: Topological ordered chart representation for the three lattices

5.2 Decoding with phrase based models

We will look into the decoding of the word lattice using the **phrase based translation model**. Phrase based models translate a foreign sentence f into the target sentence e by breaking the sentence f into sequence of phrases $\hat{f}$ where each phases can contain one or more contiguous words which are translated to one or more words of the target language which form the phrases in e and hence combined into a sentence. Each phrase should be translated to one phrase. To generalize this model into the word lattice, we need to consider the paths through the lattice with the phrase and the partition of the sentence into sequence of phrases.(Dyer et al., 2008) Altough we find that the number of phrases in the word lattices are **exponential** in the number of nodes, all possible translations are

tractable in the the lattice. We look into the Moses decoder for decoding the word lattice. The unmodified decoder builds a **translation hypothesis** from left to right by selecting a range of untranslated words and adding translations of this phrase to the end of the hypothesis being extended. When all the words are translated we consider that the translation process is complete. The word lattice decoder works in the same way where along with taking into consideration the translation units, it also keeps track of the nodes covered, given topological ordering of the nodes. For example, taking into consideration the third lattice in the figure in the input, if the edge a is translated, this will cover two nodes [0,1], even though it is a single word. A translation hypothesis is completed when all the nodes in the input lattice are covered.

5.3 Problems

There are few problems with the lattice decoding. Few of them we will look into.

5.3.1 Unreachable nodes

In the normal sentence decoder, any span of untranslated words is an extension of the partial hypothesis. In the **non-linear lattice**, another constraint needs to be added that there is always a path from the starting node of the translation extension's source to the node representing the nearest right edge of the translated material as well as the path from the ended node to the future translated spans.(Dyer et al., 2008) Figure 18 illustrates this problem. if [0,1] is translated the decoder must not consider translating [2,3] nodes as the possible extension of the partial hypothesis as there is no path that exists between [1,2] and hence this span will never be covered. Hence there will be nodes that will never be reached and we will have some unreachable nodes. This problem becomes huge in big lattices where there can be lot of unreachable nodes.

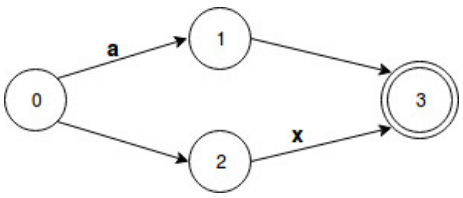

Figure 17: The span [0,3] has a inconsistent covering [0,1]+[2,3]

5.3.2 Distortion in non-linear lattice

In these **word lattices**, distortion is taken care of. These models include distortion penalty, as $|a_i - b_{i-1} - 1|$ where a_i is the starting position of the source phrase f_i and b_i is the ending position of the phrase f_{i-1}.(Dyer et al., 2008) The logic behind using this is most of the translation is monotonic the cost of skipping words while translating should be proportional to the number of words that are skipped. A limit on the distortion is also kept so that the search size does not grow out of bounds. In the **confusion networks** that we will look into the next chapter this problem of distortion is easy to resolve as we can easily define the **distortion penalty** and the **distortion limits**. This problem of distortion can be seen in the below case: If we are assuming left to right decoder as we had described above, if c is generated by the first target word and c comes after node 3, it is very much possible that it might have come via path with length 2 or length 3, depending upon which path has been chosen by the lattice while decoding. Though this problem looks small in this case, this is more of an issue in larger lattices. The cost of swapping of words in some case can be quite large at times due to large size of the lattices and is almost impossible to handle.

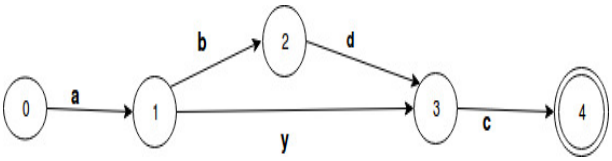

Figure 18: Distortion problem

6 Spoken Translation using Confusion Network Decoding

Spoken translation is basically identifying the words from the speech and later translating it into target language. Given a vector o representing the acoustic observations of the input utterance and let $F(o)$ be the set of transcription hypothesis which is computed by the ASR and represented by a word-graph.(Zhou, 2013) The **best translation** e^* is found out from all the strings which follows this property(Bertoldi and Federico, 2005) :

$$e^* = argmax_e \sum_{\tau(o)} Pr(e|f, o) \qquad (21)$$

where f is the hidden variable which gives the speech transcript. It is calculated by taking the

summation of all possible transcription hypothesis. The **Spoken Language Translation** assumes two things :

1. We can get the correct hypothesis in the word graph itself.

2. The quality of the translation is proportional to the transcription.

6.1 Confusion Network Decoding

Decoding can be done using Confusion Networks.

6.1.1 Confusion Network

A **confusion network (CN)** G is a type of a word lattice or a weighted directed graph that has a start node, a end node and word labels over its edges.(Bertoldi et al., 2008) The CN can have various words between two nodes and hence more than one edge. It covers all the other nodes while going from start node to end node. Hence we can represent a CN as a matrix which has words with each column having different number of words. Let us consider there are m columns and each column has d_j words. Each word can be represented as $w_{k,j}$ for all $j = 1....m$ and $k = 1....d_j$ and it is in j_{th} column and the k_{th} word in that column. Each word $w_{k,j}$ has the posterior probability of $p_{j,k}$ of having the word at the particular position in the row and column. The probability at a particular position is defined over all words in the column of G.(Bertoldi et al., 2008)
A string $f = f_i, ..., f_m$ is a **realization** of G where f_i is any word in the column $j = 1....m$. Any choice of word from the column represents the string. The set of all such realizations of G is $F(G)$. Any such realization $f = f_i, ..., f_m$ can be represented by probability $Pr(f|o)$ which can be factorized to represent as(Bertoldi et al., 2008) :

$$Pr(f|o) = \prod_{j=1}^{m} Pr(f_j|o, j) \tag{22}$$

The output of the ASR system can be converted to CN where few columns can have words as ϵ which means the empty words. Though these are similar to the other words we will consider these as different words as that may improve the **quality** of the **translation** and the **decoding efficiency**.

6.1.2 Generative Translation Process

Let us now look into the translation of the input $f = f_1....f_l$ which is composed of l phrases. The phrases of the input are basically the combination of words. the generative process gives idea about the process of how to build the specific translation of G in the **incremental** way in l steps.(Bertoldi et al., 2008)
At each step 1..l do the following :

1. A set τ_i of some yet not covered and contiguous columns are covered. The set τ_i is called the **tablet** and ϕ_i is called **fertility**.

2. One word from each column is chosen. the set of the positions of these phrases is represented as ω_i

3. A target phase $e = e_1...e_{k_i}$ is chosen of length k from the translation options of f_i and this gets added to the actual translation.

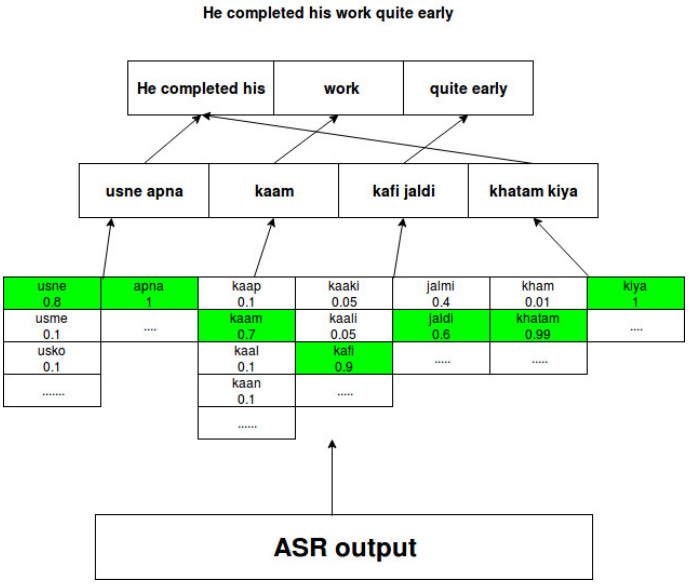

Figure 19: Generative translation process. The words alongwith their posterior probabilities are shown. The translation is produces using a phrase based decoder.

The **generative process** induces a **ordered segmentation** which is a permutation of the segmentation of the tablets and we identify a path from it ω_i which finally gives us a realization of f and finally gives the translation.The order in which the translation takes place can be in any order and hence it gives rise to local reordering.

6.1.3 CN Based log linear Model

If we use the **maximum entropy** as metric, the conditional probability can be a function of a real values feature function $h_r(e, f, o)$, and real parameters λ_r for r between $1...R$ and this expression

in parametric form can be represented as(Bertoldi et al., 2008):

$$Pr(e, f|o) = \frac{1}{Z(o)} \exp \sum_{r=1}^{R} \lambda_r h_r(e, f, o) \quad (23)$$

where $Z(o)$ is the normalization term.

Advantage - The biggest advantage of using the log linear model is that various kind of **feature functions** can be used, which can be considered as important for translation. The better performances can be found out if we use **phrases** instead of the words, where we search for the best phrase from the string of phrases form the vocabulary of phrases.(Bertoldi et al., 2008)

Feature Functions - The feature functions can be like :

1. h_{lm} : word based n-gram language model.
2. h_{dst} : negative exponential distortion model which takes into consideration the relative movement of the phrases.
3. h_{lex} : Lexicon models that assign probability to the source phrase given the target phrase by taking the count of the phrases and the words into consideration.

These functions can be decomposed using the previously used translation process. We give some weight to these feature functions and the cost of generating the translation is the **sum** of all the feature function steps. Few of these functions can be dependent on the previous term as is (h_{lm}, h_{dst}) and some of them are not dependent like h_{lex}.(Bertoldi et al., 2008)

6.1.4 Decoding Algorithm

the optimal solution can be found out though an iterative process by getting the scores for the **partial theories** and also **recombination** of these theories. The below algorithm gives idea about the decoding in the case of normal sentence as it can be also considered as the confusion network.(Bertoldi et al., 2008)

The algorithm can be described as follows :

1. Initially the theory th is empty and the possible translation theory $th^{'}$ is found out for it.
2. The **GetTablet, GetPath** and **GetTrans** loops are the three steps of the generative translation process. The GetTablet gives the proper ordering of the input phrases as it is free to use any phrase order.
3. **BuildTheory** generates the output phrase for the input phrase theory.
4. Each of the expanded theory has a a score to it added to it the untranslated theory translation score.
5. Theories which are indistinguishable are **recombined** with the initial theory.
6. The best translation is selected from the set of translations computed.

```
th = setEmptyTheory ()
Recombine (Theories[0],th)
for j = 0 to m-1
    while  (th = Theories[j])
        while  (t in getTablet (th))
            while  (o in getPath (t))
              while  (e in getTrans (t,o))
                th' = BuildTheory (th,t,o,e)
                Score (th,th')
                Recombine (Theories[j+|t|],th')
th* = getBestTrans (Theories[m])
```

The order in which the source phrases are translated is assumed to be absolutely free. This adds additional level of complexity as we need to consider lot of theories during the translation. Hence we try to set some **reordering constraint**.

6.2 Improvements in the Decoding

The CN decoding algorithm is good but it adds another level of complexity by adding the paths or the choice of words within the column. Hence, we need to find few ways to improve this.

6.2.1 Early Recombination

Analysis shows that their are pairs of tablets and paths that represent the same source and hence give the **same translated output phrase**. This may be because of the presence of the ϵ transitions.(Bertoldi et al., 2008) It is also possible that two different phrases of the same tablet have the same translation. If there are differences in these paths, it will effect only the expansion independent features. Hence, we can keep only those translations that are distinct in the tablet and apply **early recombination**. This improves the complexity in much better way.

6.2.2 Prefetching of Translation Options

The generation of the translation for a particular tablet can found out by enumerating the source phrases and **merging the translations**. The number of source phrases in a tablet can be exponential in the length of the tablet and hence this approach should be fine only for the small CN. The

better way to increase the efficiency will be by using the **prefix tree representation** and look only at the prefix in the phrase table while considering phrases **incrementally** in the tablet length. So while looking in the tablet $[j_1, j_2]$, the algorithm will look at only the its prefixes $[j_1, j_2 - 1]$. If the word sequence in the CN also occur in the phrase table, this approach enumerates exponential number of phrases. Hence, the worst case complexity will still be exponential.

6.2.3 Improved Decoding Algorithm

The algorithm defined above can be modified as follows(Bertoldi et al., 2008) :
At each step, $i = 1...m$,

1. a set of the tablet and the path yet which are not covered and the contiguous ones are covered.

2. a new phrase $e = e_1...e_{k_i}$ of length k_i is chosen and appended to the actual translation.

The improved algorithm can be defined as :

```
th = setEmptyTheory ()
Recombine (Theories[0],th)
for j = 0 to m-1
    while  (th = Theories[j])
        while  (t in getTablet (th))
          while  (e in getTrans (t))
            th' = BuildTheory (th,t,,e)
            Score (th,th')
            Recombine (Theories[j+|t|],th')
th* = getBestTrans (Theories[m])
```

Conclusion

In this paper, we presented the survey on the Automatic Speech Recognition approaches, Neural Machine Translation and its improvements and work on the integration of speech recognition and machine translation. We did it using N-Best, word lattice and confusion networks decodings. We also looked into the decoding algorithm for the confusion networks and also how it can be improved using various ways. We observed various advantages and the disadvantages of using these coupling techniques. Spoken Language Translation research has flourished significantly in the past few years, necessitating a look-back at the overall picture that these individual works have led to. Based on the survey, we find that lot of research has been done on the ASR side, NMT side and their coupling and hence these three can be combined to form a complete spoken translation system.

References

Nicola Bertoldi and Marcello Federico. 2005. A new decoder for spoken language translation based on confusion networks. In *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, pages 86–91. IEEE.

Nicola Bertoldi, Richard Zens, Marcello Federico, and Wade Shen. 2008. Efficient speech translation through confusion network decoding. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(8):1696–1705.

Pushpak Bhattacharyya. 2015. *Machine translation*. CRC Press.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint arXiv:1701.03214*.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. Technical report, MARYLAND UNIV COLLEGE PARK INST FOR ADVANCED COMPUTER STUDIES.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational linguistics*, 27(2):153–198.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Dan Jurafsky. 2000. *Speech & language processing*. Pearson Education India.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Philipp Koehn. 2017. Neural machine translation. *arXiv preprint arXiv:1709.07809*.

Gaurav Kumar, Graeme Blackwood, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. 2015. A coarse-grained model for optimal coupling of asr and smt systems for speech translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1902–1907.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*.

Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

Anthony Rousseau, Paul Deléglise, and Yannick Estève. 2014. Enhancing the ted-lium corpus with selected data for language modeling and more ted talks. In *LREC*, pages 3935–3939.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Hassan S Shavarani, Maryam Siahbani, Rantim M Seraj, and Anoop Sarkar. 2015. Learning segmentations that balance latency versus quality in spoken language translation. In *Proceedings of the Eleventh International Workshop on Spoken Language Translation (IWSLT 2015), Da Nang, Vietnam*.

Bowen Zhou. 2013. Statistical machine translation for speech: A perspective on structures, learning, and decoding. *Proceedings of the IEEE*, 101(5):1180–1202.