INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DHARWAD, KARNATAKA 580009

MINI PROJECT

A Project Report On

"TRIBAL MACHINE TRANSLATION SYSTEM"

Submitted by

M. Yashwanth (19BEC025)
N. Sujith Joseph Ratnam (19BEC029)
A. S. S. Karthik(19BCS008)
R. Mounika (19BCS124)

Under the Guidance of

Dr. Deepak K T
Asst. Professor
Electronics & Communication Engineering
Indian Institute of Information Technology, Dharwad

Acknowledgment

We would like to express our sincere gratitude to Dr. K T Deepak, Asst. Professor, Department of ECE-IIIT Dharwad for his guidance and constant support throughout the course of this minor project. We would also like to thank all the faculty and administration of the institute who ensured the needs were fulfilled for the completion of this project.

Date : December 2022

Place: Hubballi

M. Yashwanth (19BEC025)

N. Sujith Joseph Ratnam (19BEC029)

A. S. S. Karthik (19BCS008)

Rayudu Mounika (19BCS124)

Abstract

India is known for its rich diversity of culture, tradition, religion as well as languages. Over 700 languages are spoken in India including 22 scheduled languages and almost all language pairs lack significant resources for training machine translation models. There has been increasing interest in research addressing the challenge of producing useful translation models when very little translated training data is available.Lambani is under resource oral language which does not have its own script. It is spoken by a Banjara Tribe spread across different parts of India i.e.Karnataka, Andhra Pradesh, Telangana State, Maharashtra and Rajasthan.Most of banjara people can not access content on the Web because most of the content is not readily available in their language. Machine translation (MT) systems have the potential to change this.In this project we are implementing Machine Translation (MT) system which translates sentences in English, a high-resource universal language to a Lambani, a low-resource Indian tribal language.We are using two approaches: statistical machine translation and neural machine translation .

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

India has 22 major languages today. These languages belong to two classes: Indo-European languages and Dravidian languages. The languages stemming from the Dravidian branch are: Kannada, Malayalam, Telugu and Tamil. These languages are highly inflexion based languages. Lambani is spoken predominantly in Karnataka, and it is widely spoken by the Banjaras tribe.

Natural Language Processing (NLP) allows machines to understand and deduce human languages based on their interpretation, and it serves as a link between human language and data science. The task of translating text or speech from one natural language to another with as little human effort as feasible is known as machine translation (MT). This can be of several types:

1. Rule based MT: tries to model machine translation by mapping source and target language sentences using necessary grammatical rules, and follow- ing a dictionary based approach.

2. Statistical MT: statistical models that learn to translate text from a source language to a target language provided a large corpus of examples is present.

3. Neural MT: uses a neural network which directly models the conditional probability of translating a given source sentence to a target sentence.

In the 1990s, a notable advancement in Machine Translation occurred when businesses such as IBM began to use statistical models to increase translation quality. Statistical Machine Translation engines were a cutting- edge technology at the time. To translate enormous piles of content, these engines depended on powerful statistical approaches and vast volumes of data from the Internet. Later, Google would strive to make all human knowledge searchable by deploying the technique on a large scale.

Statistical Machine Translation engines were better than rule-based en- gines in the beginning, but they still made a lot of mistakes. As a result, researchers began to experiment with Deep learning based MT and Hybrid Machine Translation engines, which incorporated both statistical and neural machine translation. Machine Translation technology became more popular as a result of these improvements, which aided its global adoption.

Since Lambani is a Tribal Language,it is hard to find the original transcript. To enrich the Tribal knowledge, we make an attempt to build a two way communication between the tribe and the society.Lambani is underrepresented in the field of MT as compared to other Indian languages.

1.1 PROBLEM STATEMENT

Implementation of a Machine Translation (MT) system, which translates sentences in English, a high-resource universal language, to Lambani, a low-resource Indian tribal language with limited available data.

1.2 OBJECTIVES

- Study and analyze current literature on under resource machine translation
- Build a Statistical machine translation model for English to Lambani translation.
- Exploring and adopting Neural machine translation for under resource language.

CHAPTER 2

LITERATURE REVIEW

This study proposes and experiments with different techniques for machine translation. Therefore the review of relevant works focuses on MT using var- ious approaches. Machine Translation (MT) can be described as the task of translating text or speech from one natural language to another, with as little human effort as possible. MT aims to achieve quality translations which are semantically equivalent to the source sentence and syntactically correct in the target language. MT performs simple substitution of words on a ground level, but that alone is not enough, as recognition of whole phrases and their closest counterparts in the target language are necessary.

English to Indian language translation poses the challenge of morphological and structural divergence. For instance, (i) the number of parallel corpora and (ii) differences between languages, mainly the morphological richness and variation in word order due to syntactical divergence.

Rule based approach of MT requires linguistic knowledge of both source and target language and it generates a generic model for translation from source to target. Statistical approach is based on probability of words, phrases and sentence score of a given corpus. Statistical approach will not generate a generic model but translates the phrases or sentences with training and tuning algorithms which are based on sequence libraries of noisy channel model problems.

There has been research on statistical machine translation previously. The reliability of these studies depend on the use case, as most of these have used small datasets ranging from 1000 to 20000 sentences, yielding results from 14 to 20 BLEU scores. Sata Anuvadak [Kunchukuttan et al., 2014], a compilation of 110 independently trained translation systems which used Statistical Machine Translation (SMT) analyzes a multilingual setup through SMT based approaches.

CHAPTER 3

NATURAL LANGUAGE PROCESSING

NLP is a branch of Computer Science and Artificial Intelligence(AI) that studies how computers and human (natural) languages interact. It is used to process speech and text using machine learning techniques.

Models for speech recognition, document summarization, machine translation, spam detection, named entity recognition, question answering, autocomplete, predictive typing, and other tasks can all be developed using NLP.

Today, the majority of us own smartphones featuring speech recognition. NLP is used by these smartphones to interpret spoken language. Additionally, a lot of individuals use laptops with speech recognition embedded into the operating system.

Some Examples are: Cortana which identifies natural voice through which we can command systems.

Alexa is a Amazon's product that recognizes commands given by humans.

Gmail, which is the most widely used email service, uses NLP to process the text in the mails and filters out the spam mails.

3.1 nltk LIBRARY

NLTK (Natural Language Toolkit) is a popular toolkit for creating Python applications that interact with human language data. It offers simple user interfaces for several lexical and corpora resources. A collection of text processing libraries for categorization, tokenization, stemming, tagging, parsing, and semantic reasoning are also included.

We'll demonstrate the fundamentals of the field of natural language processing using the NLTK toolkit. We import the NLTK toolkit using "`import nltk`".

The of NLP for text data processing are:

1. Sentence Tokenization
2. Word Tokenization
3. Text Lemmatization and Stemming
4. Stop Word
5. Regex
6. Bag-of-Words
7. TF-IDF

3.1.1 Sentence Tokenization

The issue of breaking down a string of written language into its individual phrases is known as sentence tokenization, also referred to as sentence segmentation. This concept seems extremely basic. When we encounter a punctuation mark, we can break apart the phrases in English and certain other languages.

NLTK's "`nltk.sent_tokenize`" can be used to apply a sentence tokenization.

3.1.2 Word Tokenization

The challenge of breaking down a string of written language into its individual words is known as word tokenization (also known as word segmentation).

We use NLTK's "`nltk.word_tokenize`" function.

3.1.3 Text Lemmatization and Stemming

Stemming is a term that often describes a primitive optimization technique that removes prefixes and suffixes from words to get the root word within no time.

Lemmatization often refers to carrying out tasks correctly using a vocabulary and semantic analysis of words, usually with the goal of removing only inflectional endings and returning the lemma, or dictionary form, of a word.

The distinction is that a stemmer functions without context awareness, it is unable to distinguish between words having various meanings depending on their portion of speech. However, the stemmers also have certain benefits, like being quicker and simpler to use. Furthermore, the decreased "accuracy" might not be relevant for some applications.

We achieve stemming and lemmatization by:

```
"from nltk.stem import PorterStemmer, WordNetLemmatizer
 from nltk.corpus import wordnet"
```

3.1.4 Stop words

Stop words are words that are removed from text either during or after processing. These words can significantly increase noise when machine learning is applied to text. We wish to get rid of these words since they are unnecessary.

Stop words typically relate to the most prominent terms in a language, such as "and," "the," and "a," however there is no comprehensive list of stopwords. Depending on the program, the list of stop words may change.

There is a predefined list of stopwords for the most popular words in the NLTK tool. The stop words must be downloaded using the following code if we are using it for the very first time: `"nltk.download("stopwords")"`. Once the download is complete, we can use the stopwords package to load the stop words by loading it from the `"nltk.corpus"`.

3.1.5 Regex

A regular expression, regex, or regexp is a sequence of characters that define a search pattern. Regex is a powerful tool that we can utilize to further filter our content. For instance, we may eliminate any character that is not a word. The punctuation marks are frequently unnecessary, and regex makes it simple to get rid of them.

The “re” module in Python offers regular expression matching operations that are comparable to those in Perl. To replace the matches for a pattern with a replacement string, use the `"re.sub"` method.

3.1.6 Bag-of-Words

Since raw text cannot be processed directly by machine learning algorithms, it must be transformed into vectors of numbers. The term for this is feature extraction.

When working with text, a well-liked and straightforward feature extraction method is the bag-of-words model. It explains where each word appears in a document.

To apply this paradigm, we must:

1. Create a dictionary of well-known words (also called tokens)
2. Select an indicator of the existence of well-known words.

The sequence and structure of the words are completely ignored. It is termed as a bag of words for this reason. This model tries to determine whether a known word is in a document but does not know the word's location.

3.1.7 TF-IDF

A statistical method for assessing a word's significance to a document in a collection or corpus is called term frequency-inverse document frequency, or TF-IDF.

The TF-IDF scoring value raises accordingly to the number of times a word occurs in the document, but it diminishes by the number of documents in the corpus that contain the word.

1. **Term Frequency (TF)** is a measurement of a word's frequency in the current document.

$$TF(\boldsymbol{term}) = \frac{Number\ of\ times\ \boldsymbol{term}\ appears\ in\ a\ document}{Total\ number\ of\ items\ in\ the\ document}$$

Figure: Formula for TF

2. **Inverse Term Frequency (ITF)** is a scoring system for the rarity of a term across documents.

$$IDF(\boldsymbol{term}) = \log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ \boldsymbol{term}\ in\ it}\right)$$

Figure: Formula for IDF

3. Finally, we may determine the **TF-IDF** score using the previously mentioned formulae.

$$TFIDF(\boldsymbol{term}) = TF(\boldsymbol{term}) * \mathrm{IDF}(\mathbf{term})$$

Figure: Formula for TF-IDF

CHAPTER 4

DATASET DESCRIPTION

Parallel text is required for building high quality machine translation (MT) systems, as well as for other multilingual NLP applications. As far as Indic languages are concerned they are often termed as resource scarce in terms of the research done on them and the availability of related resources, during our course project we used broadly two types of datasets:

1. Already Available
2. Synthesized dataset

The datasets were then used accordingly for the below discussed models:

4.1 AVAILABLE DATASETS

4.1.1 Available Datasets For Lambani

Dataset used to build a model consists of 9000 English-Lambani parallel sentences.

4.1.2 Available Datasets For Kannada

1. **PM-India Dataset**: The dataset is based on the news updates on the PMIndia website that are presented in an “infinite scroll” format. Exami- nation of the browser traffic showed that the requests for lists of URLs followed a fairly simple format, and so researchers were able to devise a custom scraper to obtain all the news feed articles in the archive, for each language. They developed the dataset using the following number of articles:
   The dataset consists of 29000 sentences for English-Kannada.

2. **AI4Bharat Samanantar Dataset**: Samanantar is the largest publicly available parallel corpora collection for Indic languages. The publicly released version is

randomly shuffled, untokenized, and deduplicated. The corpus has 49.6M sentence pairs between English to Indian Lan- guages. The dataset of interest is English to Kannada(En-kn) dataset, The En-Kn dataset contains around 4 million sentence pairs.

3. **Swadesh sentences:** 7000 parallel corpora sentences following Swadesh guidelines.

English (en)	5722	Odia (or)	3002
Hindi (hi)	5244	Malayalam (ml)	3407
Bengali (bn)	3937	Punjabi (pa)	2764
Gujarati (gu)	3872	Kannada (kn)	2679
Marathi (mr)	3762	Manipuri (mni)	1612
Telugu (te)	3631	Assamese (as)	1571
Tamil (ta)	3606	Urdu (ur)	1413

Figure 3.1: PM India articles

4.2 SYNTHESIZED DATASETS

Synthesized datasets throughout the duration of our project there was a need of datasets that could be used so as to avoid errors such as ‘Keyword Error’ which can be ensured by using a diverse dataset of parallel corpora which ensure that while training the model has exposure to the different words that exist in the corpus.

1. **180,000 sentences:** Tatoeba.org is a Multilingual Corpus of Sentence Equivalents, Tatoeba.org is a large database of example sentences trans- lated into many languages by its members who volunteer their time. Many of the sentences came from the Tanaka Corpus which was im- ported into the Tatoeba Corpus. The database contains tab delimited bilingual sentence pairs in the format, English + TAB + The Other Language + TAB + Attribution. 180k sentences of such format that followed the Anki guidelines were

translated to kannada language using google translate to synthesize a dataset. The sentences were progressive in length from 1 word to 50 words per sentence.

CHAPTER 5

STATISTICAL MACHINE TRANSLATION

5.1 LANGUAGE MODELS

Models that assign probabilities to sequences of words are called language models or LMs. Language modeling helps to find the most likely/probable/meaningful word sequences. So, basically, it predicts how likely it is for a different word to occur next given the sequence of the recent words. It helps to remove similar acoustics like “Here” and “hear” or “sea” vs “see”. A language model takes text input and outputs the next word or character. The input is often called a “context” or “history”, because it represents what has been written so far. Assigning probabilities to sequences of words is also essential in machine translation. Suppose we are translating a Kannada source sentence:

ಅವರು ವರ%&ಾರ() ಮುಖ, -ಷಯವನು1 ಪ(ಚ45ದರು

He to Reporters Introduced Main Content

As part of the process we might have built the following set of potential rough EnglishTranslations:

“He introduced reporters to the main contents of the statement.
He briefed reporters the main contents of the statement.”

5.1.1 N-Grams

An n-gram is a sequence of n words. 2 - gram (Bigram) is a two-word sequence of words and a 3-gram (Trigram) is a three-word sequence of words.

With a large enough corpus, such as the web, we can compute these counts and estimate

the probability. While this method of estimating probabilities directly from counts works fine in many cases, it turns out that even the web isn't big enough to give us good estimates in most cases. This is because language is creative; new sentences are created all the time, and we won't always be able to count entire sentences. One thing we can do is decompose this probability using the chain rule of probability:

P(w1:n) = P(w1)P(w2|w1)P(w3|w1:2)...P(wn|w1:n−1)

The chain rule shows the link between computing the joint probability of a sequence and computing the conditional probability of a word given previous words.N-gram model predicts the probability of the Nth word of the sentence given previous N-1 words of the sentence. The assumption that the probability of a word depends only on the previous word is called a Markov assumption.

5.1.2 Language modeling toolkits:

1. SRILM - The SRI Language Modeling (SRILM) toolkit offers tools for building statistical language models in speech recognition, statistical tagging and segmentation, and machine translation. It has a set of C++ which helps in implementing language models, supporting data structures and miscellaneous utility functions.

2. Kenlm - The toolkit everyone loves, really fast and can build really big language models. But no interpolation, only counts interpolation. No Witten-Bell too, no entropy pruning.

5.2 PHRASE TABLE.

Phrase-based models used to be state of the art for about a decade until neural methods came around. This is what powered google translate to become famous and they also have the benefit of being relatively simple conceptually. The word-based models translate words as atomic units while phrase-based models can say phrases and this has several advantages one is that many-to-many translations so you translate many words into many words can handle non-compositional phrases. So all kinds of idiomatic expressions like it's raining cats and dogs that if you pin it down into words are really hard to explain but as phrases, you just memorize them it also has the benefit of using local context in translation so a key feature in resolving the ambiguity of the word, that is to look at the surrounding words so actually translating a word with its surrounding words might actually help a lot in translating so here you see an example of where phrase-based models are kind of good solution for problems if this were word 'natuerlich 5598 which translates into 'of course' so you have one word moving into two words.

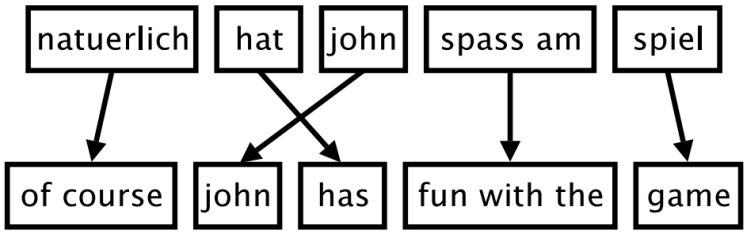

Figure: Mapping of Words

Here, of course, also the translation of artillery, both of which are not really correct course usually means something else and off is incredibly ambiguous. When we talk about phrases we don't mean linguistic phrases in a sense of what a syntax tree would call noun phrases, web phrases and so on. There are just any sequences of words okay so the key thing that has to be learned here is a phrase translation table like this one down below.

Translation	Probability $\phi(\bar{e}\|f)$
of course	0.5
naturally	0.3
of course ,	0.15
, of course ,	0.05

Figure: Phrase Probability Table

5.3 DECODING

Finding the translation with the highest score based on these formulae is the job of decoding in machine translation. When a specific sentence is given as input, there are an exponentially large number of possible solutions, making this a challenging task. It has even been demonstrated that the decoding problem for the machine translation models that have been described as NP-complete. In other words, for an input text even of small length, it is computationally too expensive to thoroughly examine all alternative translations, score them, and choose the best.

The search for the best translation may be conducted effectively, and can be achieved by various techniques. They are known as heuristic search strategies. The finest translation may not always be found, but we do try to find it frequently enough or at least one that 's similar to it.

There are two different kinds of errors that could stop this.

1. A search mistake is when the greatest probability translation, or the optimal translation according to the model, cannot be found.
2. And when the best translation is the bad translation.

CHAPTER 6

NEURAL MACHINE TRANSLATION

6.1 TRANSFORMERS

Transformer is a model architecture that completely ignores recurrence in favor of drawing global dependencies between input and output. Significantly greater parallelization is possible with the Transformer. It is the first transduction model to calculate representations of its input and output only via self-attention, without the use of convolution or sequence-aligned RNNs.

The Transformer model uses a self-attention mechanism to extract features for each word and determine the importance of each word relative to the other words in the sentence. Furthermore, since these features are simply weighted sums and activations rather than recurrent units, they may be efficiently and concurrently computed.

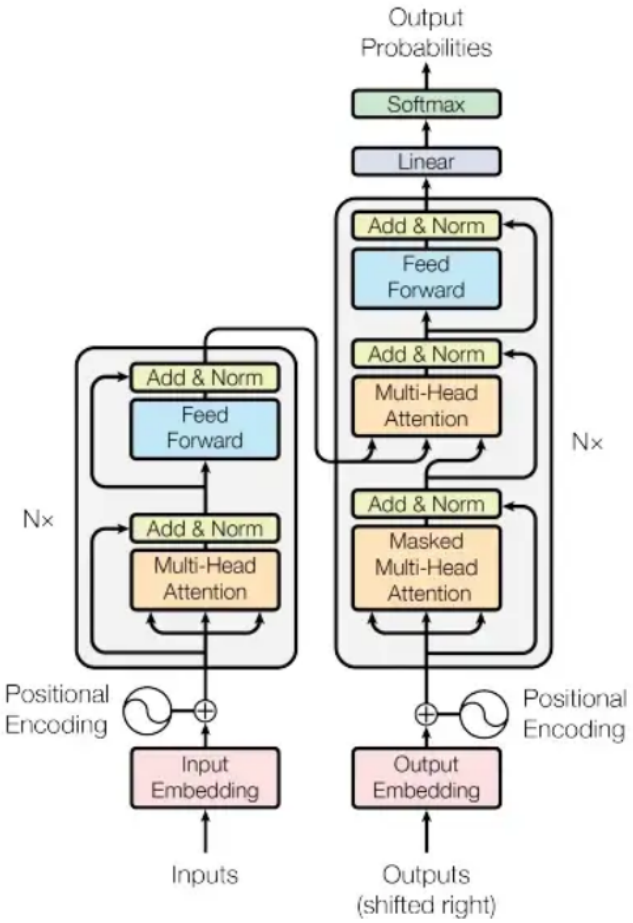

Figure: Transformers

6.1.1 Self-Attention

Self-attention is a sequence-to-sequence operation in which a sequence of vectors enters and leaves. Let's refer to the input vectors as x_1, x_2,..., and x_t, and the corresponding output vectors as y_1, y_2,..., and y_t. Each vector has a dimension of k. The dot product is the simplest option, and the self attention operation clearly takes a weighted average across all the input vectors to produce output vector y_i.

Three components must be added to our model's self-attention mechanism: Keys, Values, and Queries.

The Query, the Key, and the Value are three different ways that each input vector is employed in the self-attention process. Once the weights have been determined, it is compared to the other vectors in each role to obtain its own output, y_i(Query), the j-th output, y_j(Key), and to compute each output vector (Value).

Three weight matrices of dimensions kxk are required to obtain these roles, and three linear transformations must be computed for each x_i.

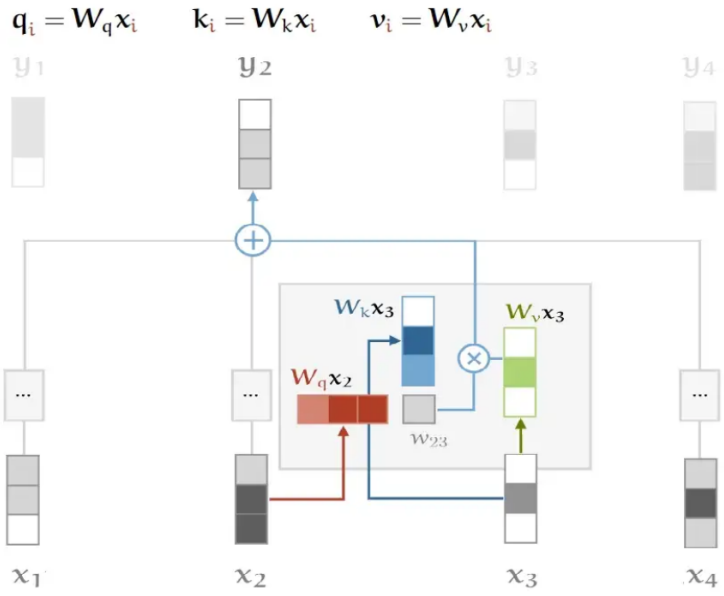

Figure: Self-Attention with Keys, Queries and Values

These three matrices—commonly referred to as K, Q, and V—apply three learnable weight layers to the same encoded input. As a result, we can use the attention mechanism of the input vector with itself, or a "self-attention," as each of these three matrices originates from the same input.

Next, we determine the attention scores using the Q, K, and V matrices. The results indicate how much attention should be paid to different positions or words in the input sequence in relation to a word at a particular position. That is, the dot product of the key vector for the particular term we are scoring and the query vector. As a result, we determine the dot product (.) of q1 and k1 for position 1, then q1 and k2, q1 and k3, and so forth.

The "scaled" factor is then used to create gradients that are more stable. Large values make the softmax function ineffective, which causes the gradients to evaporate and slows learning. We multiply by the Value matrix after applying "softmax" to maintain the values of the words we want to concentrate on while reducing or eliminating the values for the unnecessary terms.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Figure: Attention(Q, K, V)

6.1.2 Multi-Head Attention

The attention scores in the above description are concentrated on the entire sentence at once; this would result in the same results even if two sentences had the same words in a different order. Instead, we would prefer to focus on various word fragments. We can give the self attention greater power of discrimination, by combining several self attention heads, dividing the word vectors into a fixed number (h, number of heads), and then applying self-attention on the corresponding chunks, using Q, K, and V sub-matrices. This results in h separate score output matrices.

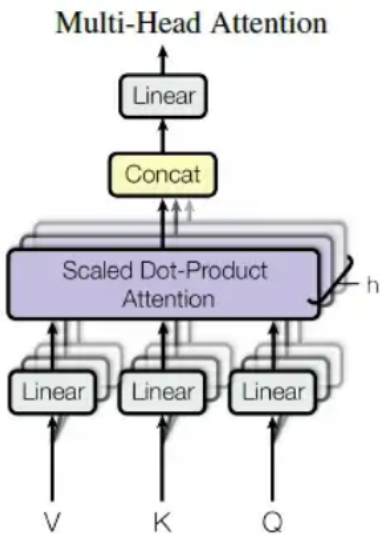

Figure: Multi-Head Attention

However, the following layer (the Feed-Forward layer) only needs one matrix, a vector for each word, so "we concatenate the output matrices and multiply them by an additional weights matrix Wo, after computing the dot product of each head. The information from each attention head is collected in the final matrix.

6.1.3 Positional Encoding

Because the network and the self-attention mechanism are permutation invariant, we briefly indicated that the order of the words in the sentence is a problem to be solved in this model. The solutions remain the same even if we rearrange the words in the input sentence. The word embedding has to have a representation of the word's place in the sentence added to it.

We briefly said that the order of the words in the phrase is an issue to be handled in this model because the network and the self-attention mechanism are permutation invariant. Even if we change the order of the words in the input sentence, the solutions remain the same. A visual depiction of the words position in the phrase must be included in the word embedding.

Therefore, we use a function to convert the sentence location to a real valued vector. The network will figure out how to use this data. Another strategy would be to

code every known location with a vector using a position embedding technique, which is similar to word embedding. Although positional encoding allows the model to extrapolate to sequence lengths longer than those observed during training, it would require phrases of all allowed positions within the training loop.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Figure: Sinusoidal Function for Positional Encoding

6.1.4 Encoder

We can now introduce the encoder components after describing all the major model components.

1. Encode the position and include it in the input embedding (our input words are transformed to embedding vectors). The pre-softmax linear transformation and the two embedding layers (encoder and decoder) share the same weight matrix. We multiply those weights by the square root of the model dimension in the embedding layers.

2. Two sub-layers, a multi-head self-attention mechanism and a fully connected feed-forward network, are contained within N=6 identical layers (two linear transformations with a ReLU activation). However, it does it positionally, which implies that the same neural network is applied to each and every "token" vector that is a part of the sentence sequence.

3. Each sub-layer (attention and FC network) has a residual connection that adds up the output and input of the layer before normalizing the layer.

4. Every residual connection has a regularization applied before it: Before adding it to the sub-layer input and normalizing it, we apply dropout to each sub-output.

Additionally, we apply dropout with a dropout rate of 0.1 to the sums of the embeddings and positional encodings in both the encoder and decoder stacks.

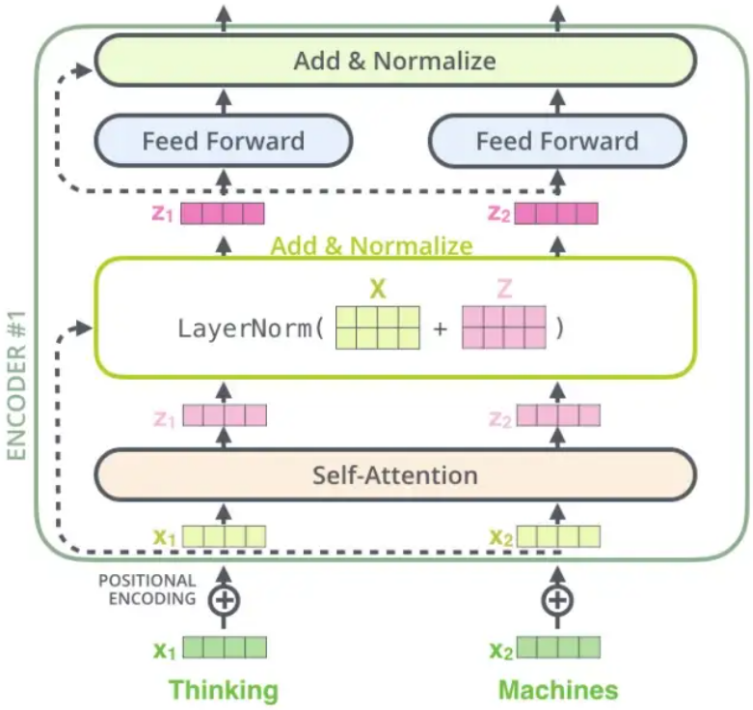

Figure: Encoder Block

Keep in mind that only the vector from the last layer (6-th) is sent to the decoder.

6.1.5 Decoder

Despite sharing some components with the encoder, the decoder uses them differently to accommodate for encoder output.

1. Positional encoding: Similar to the one in the encoder.

2. N = 6 identical layers with 3 sublayers each. To stop positions from attending to succeeding positions, first use masked multi-head attention or masked causal attention. The predictions for location I can only depend on the known outputs at

positions less than I because of this masking and the fact that the output embeddings are offset by one position. The values corresponding to the banned states are set to in the dot-product attention modules' softmax layer. The second part, known as "encoder-decoder attention," performs multi-head attention over the decoder's output. The decoder's output provides the Key and Value vectors, but the earlier decoder layer provides the queries. This enables every decoder location to monitor every input sequence point. And finally the fully-connected network.

3. Similar to the encoder, the residual connection and layer normalization are centered on each sub-layer.

4. Likewise, carry through the encoder's residual dropout once more.

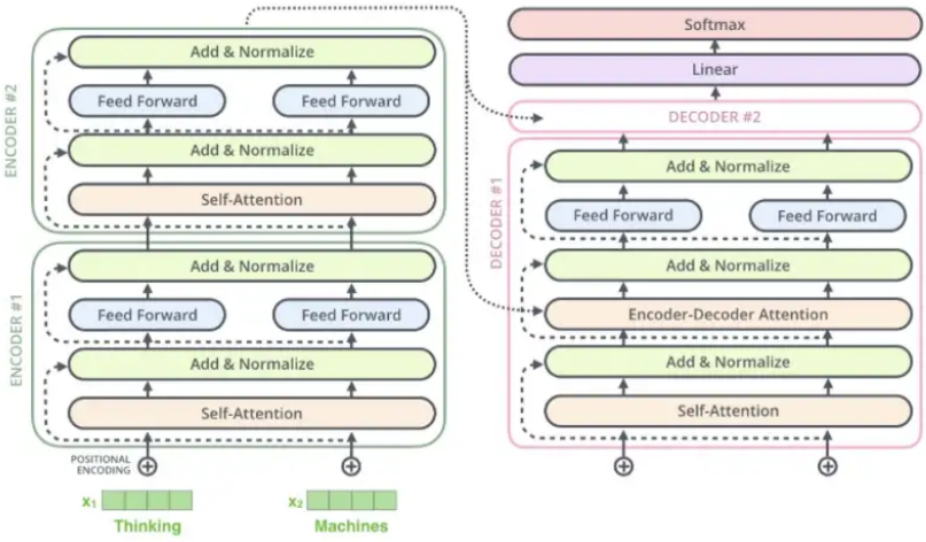

Figure: Decoder Block

The linear layer, a fully linked network, transforms the stacked outputs into the logits, a considerably larger vector at the conclusion of the N stacked decoders. Then, the softmax

layer converts those logit scores into probabilities (all positive, all add up to 1.0). The word linked with the cell with the highest probability is produced as the output for this time step.

CHAPTER 7

STATISTICAL MACHINE TRANSLATION IMPLEMENTATION

7.1 INSTALLATION

sudo apt-get install [package name]

Packages:
g++
git
subversion
automake
libtool
zlib1g-dev
libicu-dev
libboost-all-dev
Libbz2-dev
liblzma-dev
python-dev
graphviz
imagemagick
make
cmake
libgoogle-perftools-dev (for tcmalloc)
autoconf
Doxygen

7.2 DOWNLOAD CMPH

(C Minimal Perfect Hashing Library) is needed for phrase table binarization
wget http://downloads.sourceforge.net/project/cmph/cmph/cmph-2.0.tar.gz
tar xf cmph-2.0.2.tar.gz
cd cmph-2.0.2

./configure --prefix=$PWD/build

make

make install

7.3 DOWNLOAD MosesDecoder

wget https://github.com/moses-smt/mosesdecoder/archive/master.zip

unzip master.zip

cd mosesdecoder-master

./bjam --with-cmph=/home/s2st/workspace/smt/cmph-2.0.2/build -j`nproc`

7.4 VERSION OF BOOST

cat /usr/include/boost/version.hpp | grep "BOOST_LIB_VERSION"

// BOOST_LIB_VERSION must be defined to be the same as BOOST_VERSION

#define BOOST_LIB_VERSION "1_71"

7.5 DOWNLOAD MGIZA

The steps are done as given in http://www2.statmt.org/moses/?n=Moses.ExternalTools#ntoc3

wget https://github.com/moses-smt/mgiza/archive/master.zip

unzip master.zip

cd mgiza-master/mgizapp

cmake .

Make

make install

Go to mosesdecoder-master folder

mkdir tools

cd tools

cp -r ../../mgiza-master/mgizapp/bin/* ./

cp ../../mgiza-master/mgizapp/scripts/merge_alignment.py ./

7.6 RUN MOSES FOR THE FIRST TIME

wget http://www.statmt.org/moses/download/sample-models.tgz

tar xzf sample-models.tgz

7.7 TOKENIZATION

Indic NLP - To tokenize Indian Languages

7.7.1 Installation

!pip install indic-nlp-library

from india nlp.tokenize import sentence_tokenize

indicnlp.tokenize.indic_detokenize.trivial_detokenize(text, lang='kn')

7.8 CLEANING

We have to clean the tokenized text file

~/mosesdecoder/scripts/training/clean-corpus-n.perl \

~/corpus/lambani.clean 1 80

7.9 LANGUAGE MODEL TRAINING

Go to the parent folder of mosesdecoder-master folder

mkdir lm

cd lm

_binary ~/workspace/smt/lm/exp1/kannada-lam.arpa.kn ~/workspace/smt/lm/exp1/train.blm.en

~/workspace/smt/mosesdecoder-master/bin/lmplz -o 3 <

~/workspace/smt/corpus/exp1/english-lam.en > ~/workspace/smt/lm/exp1/kannada-lam.arpa.kn

*Then you should binarise (for faster loading) the *.arpa.en file*

~/workspace/smt/mosesdecoder-master/bin/build

7.10 TRAINING THE TRANSLATION SYSTEM

Go to the parent folder of mosesdecoder-master folder

mkdir ~/working

cd ~/working

nohup nice ~/mosesdecoder/scripts/training/train-model.perl -root-dir train \ -corpus ~/corpus/lambani.clean \ -f lb -e en -alignment grow-diag-final-and -reordering msd-bidirectional-fe \ -lm 0:3:$HOME/lm/english-lam.en:8 \ -external-bin-dir ~/mosesdecoder/tools >& training.out &

7.11 BINARIZED MODEL

mkdir ~/workspace/smt/working/binarised-model

cd ~/workspace/smt/working

~/mosesdecoder/bin/processPhraseTableMin \ -in train/model/phrase-table.gz -nscores 4 \ -out binarized-model/phrase-table ~/mosesdecoder/bin/processLexicalTableMin \ -in train/model/reordering-table.wbe-msd-bidirectional-fe.gz \ -out binarised-model/reordering-table

7.12 TESTING WITHOUT TUNING

~/mosesdecoder/bin/moses -f ~/working/mert-work/moses.ini

7.13 BLEU SCORE

~/workspace/smt/mosesdecoder-master/scripts/generic/multi-bleu.perl -lc ~/workspace/smt/corpus/test.kn < ~/ashwini/workspace/test.translate

CHAPTER 8

NEURAL MACHINE TRANSLATION IMPLEMENTATION USING OpenNMT

8.1 INSTALLATION STEPS

8.1.1 Package Installation

!pip install OpenNMT-tf[tensorflow]

8.2 BUILD VOCABULARY

onmt-build-vocab --size 50000 --save_vocab src-vocab.txt src-train.txt

onmt-build-vocab --size 50000 --save_vocab tgt-vocab.txt tgt-train.txt

8.3 CREATE A data.yml FILE

data= """model_dir: /content/gdrive/MyDrive/MT

data:

  train_features_file: /content/gdrive/MyDrive/MT/vocab.src

  train_labels_file: /content/gdrive/MyDrive/MT/vocab.tgt

  eval_features_file: /content/gdrive/MyDrive/MT/src_val.txt

  eval_labels_file: /content/gdrive/MyDrive/MT/tgt_infer.txt

  source_vocabulary: /content/gdrive/MyDrive/MT/vocab.src

  target_vocabulary: /content/gdrive/MyDrive/MT/vocab.tgt

train:

  save_checkpoints_steps: 1000

  maximum_features_length: 50

```
  maximum_labels_length: 50
eval:
  external_evaluators: BLEU
params:
  optimizer: Adam
  optimizer_params:
    beta_1: 0.8
    beta_2: 0.998
  learning_rate: 3.0
  dropout: 0.3
infer:
  batch_size: 32
  """
```

8.4 TRAINING THE MODEL

```
!onmt-main --model_type Transformer--config data.yaml --auto_config train --with_eval
```

8.5 INFERENCING

```
!onmt-main --config ./data.yaml --auto_config infer --features_file src_val.txt --predictions_file
tgt_infer.txt
```

8.6 EXPORTING THE MODEL

```
!onmt-main --config data.yaml --auto_config export --output_dir /content/gdrive/MyDrive
```

8.7 BLEU SCORE

```
!pip install sacrebleu
result = bleu.corpus_score(hypo, lines)
```

CHAPTER 9

RESULTS

Language		Sentence	
Source	**Target**	**Train**	**Test**
English	Kannada	4L	2L
English	Lambani	8.1K	0.9K

Table: Illustrates the size of Train and Test Data Set

Languages	Model	BLEU Score
English-Lambani	Statistical Model	12
English-Kannada	Transformer Model	15.47

Table: Illustrates the BLEU Score

Source Text	Predicted	Ground Truth
Can you get your wages?	ಕರೆಚೆ ತಾರ ಪಗಾರ ಮಳ ಸಕಚ ಕ?	ತೋನ್ ತಾರ ಪಗಾರ ಮಳ ಸಕಚ ಕ?
Can you swim as fast as him?	ತೋನ್ ವೊರ್ ಅತ್ಯಾ ಜೋರ್ ಪೇರೇನ್ ಆವಚ ಕ?	ತೋನ್ ವೊರ್ ಅತ್ಯಾ ಜೋರ್ ಪೇರೇನ್ ಆವಚ ಕ?
Can you tell wheat from barley?	ಕರೆಚೆ ಬಾರ್ಲಿ ಅನ	ತು ಬಾರ್ಲಿ ಅನ ಘಂವು ಕು ರಚ ಕೆನ್ ಕೆ ಸಕೇಚೆ ಕ?

Table: Illustrates the Predictions using Statistical Model

Source Text	Predicted	Google Translation
Students should abide by the rules.	ವಿದ್ಯಾರ್ಥಿಗಳು ಕೂಡಾ ಶಿಸ್ತು ಪಾಲಿಸಬೇಕು	ವಿದ್ಯಾರ್ಥಿಗಳು ನಿಯಮಗಳನ್ನು ಪಾಲಿಸಬೇಕು.

Table: Illustrates the Predictions using Transformers

CHAPTER 10

CONCLUSION

In this project, we accomplished the machine translation of Lambani language, in the en-lb to be incorporated in the conventional speech-to-speech model. In this project, we first built a baseline model for English-to-Kannada translation. We then create a dataset of competitive size to compare the model results. We focused on creating an SMT for En-lb and an NMT to build a parent model of en-kn.This provides us with a platform to build on for machine translation for the Lambani language. We hope this model and the synthesized dataset can be used for future research.For the future scope of the project we can use Transfer Learning to build the child model of en-lb Neural Network model.

In this project,We have implemented a phrase-based statistical machine translation system which translates english sentences into lambani sentences.

We present a pruning algorithm for Machine Translation, where we assess whether the translation event encoded in a phrase pair can be decomposed into combinations of events encoded in other phrase pairs.

We have also build transformer based English to kannada machine translation model that can be used as parent model to do transfer learning for English to lambani Translations

In future these approaches can be used to build machine translation system for other tribal languages

ACKNOWLEDGMENTS

We would like to share our sincere gratitude to everyone who helped us achieve this project. During the implementation, we learnt new ideas and mechanisms. We did face challenges while learning and executing but we are grateful to our mentors for helping us get over our difficulties and compile our ideas into working code.

We would thank our project guide, **Dr. Deepak K T** for mentoring us and giving us this opportunity to apprise ourselves with these new concepts. We would also thank Mr.Prasanth for helping us with the imple- mentation. We would like to show our gratitude to them for helping us understand the minute details of the project work.

We are also thankful to our institute, classmates and families for keeping us motivated during the development of this project

M. Yashwanth (19BEC025)

N. Sujith Joseph Ratnam (19BEC029)

A. S. S. Karthik(19BCS008)

R. Mounika (19BCS124)

Seventh semester

Indian Institute of Information Technology Dharwad.

REFERENCES

philip koehn-http://www2.statmt.org/book/slides/07-language-models.pdf

philip koehn-http://www2.statmt.org/book/slides/06-decoding.pdf

philip koehn-http://www2.statmt.org/book/slides/05-phrase-based-models.pdf

philip-koehn-http://mt-class.org/jhu/slides/lecture-introduction.pdf

NLP-https://web.stanford.edu/~jurafsky/slp3/3.pdf

Attention Is All You Need-31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Neural Machine Translation for Low-Resource Languages: A Survey

A Baseline Neural Machine Translation System for Indian Languages Jerin Philip, Vinay P. Namboodiri, C. V. Jawahar - arxiv 2019

Neural Machine Translation for Low-Resourced Indian Languages Himanshu Choudhary, Shivansh Rao, Rajesh Rohilla - Conference on Language Resources and Evaluation LREC 2020.

Machine Translation for a very Low-Resource Language - Layer Freezing approach on Transfer Learning-Proceedings of the 29th International Conference on Computational Linguistics.

MosesDecoder Manual-http://www2.statmt.org/moses/manual/manual.pdf

Transformers Pytorch Documentation-https://huggingface.co/docs/transformers/index

Jay Alammar's Blog on Illustrated Transformer-https://jalammar.github.io/illustrated-transformer