Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Machine Translation

June 30, 2020

Ganesamanian Kolappan, Proneet Sharma

Machine Translation(MT)

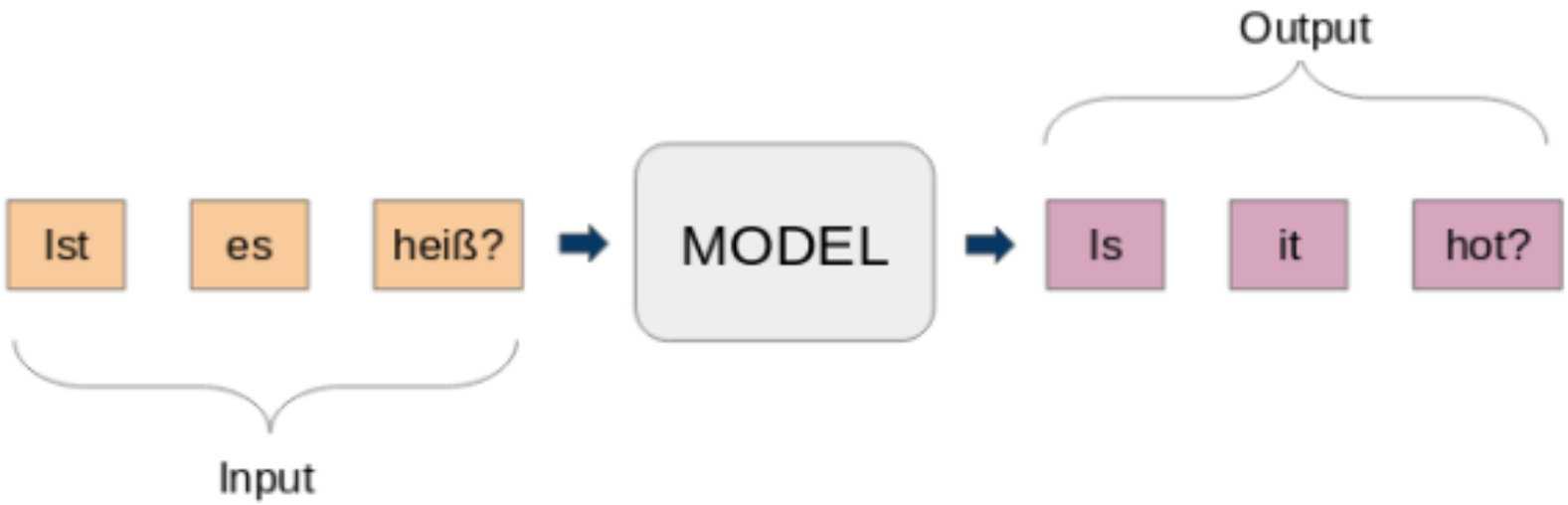


Figure 1: Basic machine translator

How NMT works?

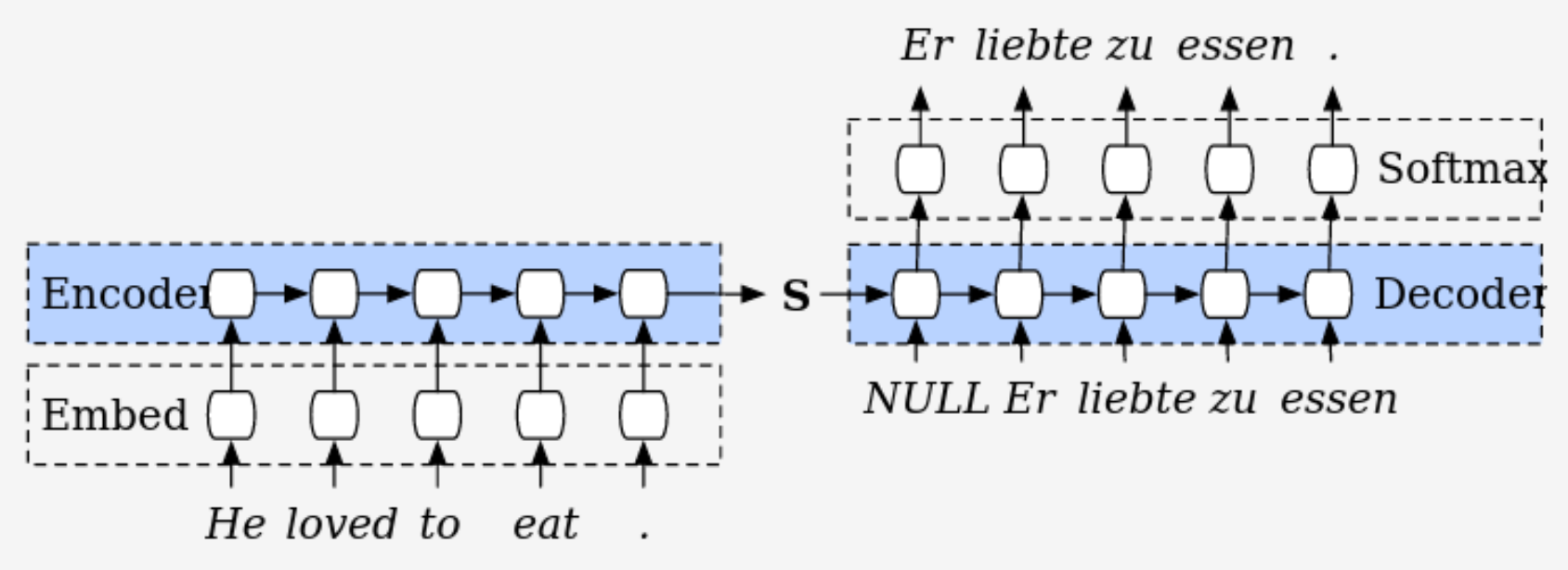


Figure 2: Basic neural machine translator

About the data

```
#Loading of data from txt file
data = pd.read_csv("deu-eng/deu.txt", sep='\t', names=['en','de', 'extra information'])
data.tail(10)
```

	en	de	extra information
208476	I recommend contributing sentences in your own...	Ich empfehle, muttersprachliche Sätze beizutra...	CC-BY 2.0 (France) Attribution: tatoeba.org #6...
208477	A building with high ceilings and huge rooms m...	Ein Gebäude mit hohen Decken und riesigen Räum...	CC-BY 2.0 (France) Attribution: tatoeba.org #2...
208478	As a prank, some students let three goats loos...	Als Streich ließen einige Schüler drei Ziegen ...	CC-BY 2.0 (France) Attribution: tatoeba.org #9...
208479	In today's world, we have to equip all our kid...	In der heutigen Welt müssen wir all unsere Kin...	CC-BY 2.0 (France) Attribution: tatoeba.org #3...
208480	Death is something that we're often discourage...	Wir werden oft davon abgehalten, über den Tod ...	CC-BY 2.0 (France) Attribution: tatoeba.org #1...
208481	At a moment when our economy is growing, our b...	In einem Moment, in dem unsere Wirtschaft wäch...	CC-BY 2.0 (France) Attribution: tatoeba.org #3...
208482	Even if some sentences by non-native speakers ...	Auch wenn Sätze von Nichtmuttersprachlern mitu...	CC-BY 2.0 (France) Attribution: tatoeba.org #6...
208483	If someone who doesn't know your background sa...	Wenn jemand, der deine Herkunft nicht kennt, s...	CC-BY 2.0 (France) Attribution: tatoeba.org #9...
208484	If someone who doesn't know your background sa...	Wenn jemand Fremdes dir sagt, dass du dich wie...	CC-BY 2.0 (France) Attribution: tatoeba.org #9...
208485	Doubtless there exists in this world precisely...	Ohne Zweifel findet sich auf dieser Welt zu je...	CC-BY 2.0 (France) Attribution: tatoeba.org #7...

Figure 3: Sample data

The data can be downloaded here.

Preprocessing

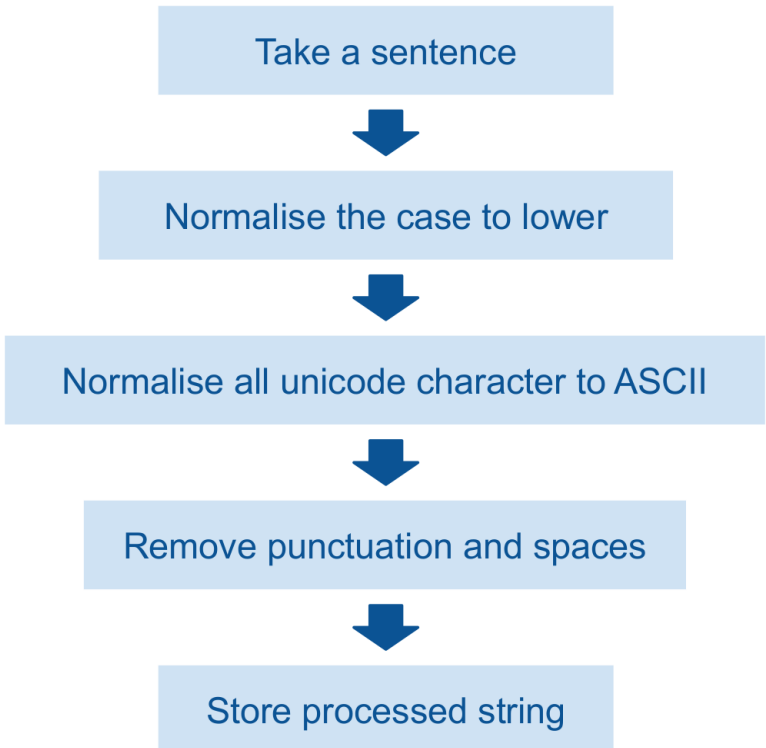


Figure 4: Steps involved in data preprocessing

```
def data_preprocessing(dataframe):
    doc = [re.split('\s+',dataframe.iloc[i]) for i in range(len(dataframe))]
    clean_doc = []
    for sent in doc:
        clean_sent = []
        clean_sent_str = ''
        for token in sent:
            # Normalise the case to lower
            token = token.casefold()
            # Normalise all unicode character to ASCII
            token = normalize('NFD',token).encode('ascii', 'ignore')
            token = token.decode('UTF-8')
            # Remove punctuations
            token = re.sub('[^\w|^\s]', '', token)
            if token != '':
                clean_sent.append(token)
        clean_sent_str = ' '.join(clean_sent)
        # Store processed string
        clean_doc.append(clean_sent_str)
    return clean_doc
```

Tokenization

- Words are mapped to integer which is the number of occurrence of the particular word in the whole corpus.
- Separate tokenizer is used for English and German sequences.
- The function below will return a tokenizer in the dictionary format.
- Find the length of longest sequence in a list a sentences.

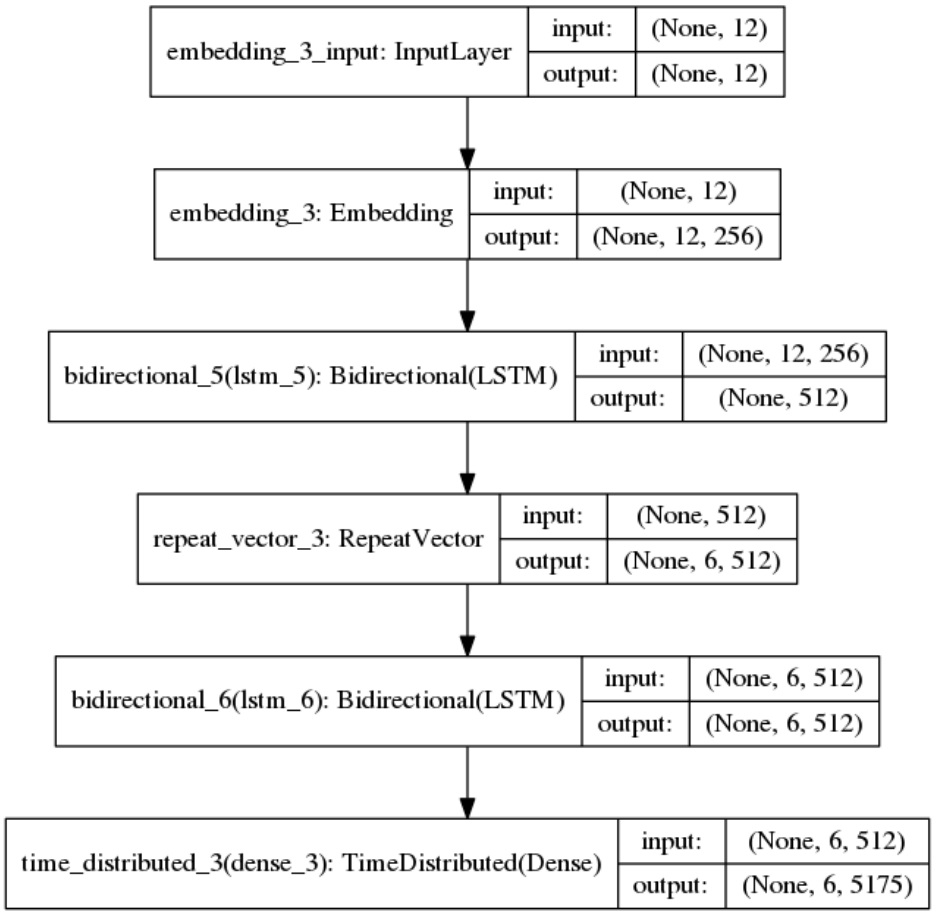
```
# fit a tokenizer
def create_tokenizer(data):
    tokenizer = Tokenizer(num_words=5000)
    tokenizer.fit_on_texts(data)
    return tokenizer

#Tokenizing the vocab
def get_vocab_size(data):
    tokenizer = create_tokenizer(data)
    vocab_size = len(tokenizer.word_index)+1
    max_length = max(len(sent.split()) for sent in data)
    return vocab_size, max_length, tokenizer
```

One hot encoding

- Input and output sequence is encoded to integer and padded to the maximum sentence length.
- This is done to use word embedding for the input sequences and one-hot encoding for the output sequences.
- One hot encoding is needed for the model to predict the probability of each word in the vocabulary.

```
#Preparing the training data
def encode_sequences(tokenizer, length, data):
    X = tokenizer.texts_to_sequences(data)
    X = pad_sequences(X, maxlen=length, padding='post')
    return X
# one hot encode target sequence
def encode_output(sequences, vocab_size):
    ylist = list()
    for seq in sequences:
        encode = to_categorical(seq, num_classes=vocab_size)
        ylist.append(encode)
    y = np.array(ylist, dtype=np.single)
    y = y.reshape(sequences.shape[0], sequences.shape[1], vocab_size)
    return y
```

Model in keras



- **Sequential()** - This class function helps to stack the upcoming layer linearly, so that each layer has one input and one output
- **Embedding()** - This is helpful to form the word vector, looks like lookup table, it takes up three arguments first is the vocab size, dimension along x, dimension along y, mask zero makes the 0 index invalid. The word is mapped to integer and this integer is used as a index to access its corresponding vector.
- **LSTM()** - Builds the LSTM layer with the specified dimension.
- **Repeat vector()** - Makes the context vector available at each time step of the decoder until the length of target sequence.
- **TimeDistributed()** - This is wrapper that is apply as a layer to every temporal slice of an input. Therefore, it outputs one time step from the sequence for each time step in the input since all the input time step is processed at a same instance.

Figure 5: Model configuration

Model Summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 17, 256)           3638528
_________________________________________________________________
lstm (LSTM)                  (None, 256)               525312
_________________________________________________________________
repeat_vector (RepeatVector) (None, 8, 256)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 8, 256)            525312
_________________________________________________________________
time_distributed (TimeDistri (None, 8, 8241)           2117937
=================================================================
Total params: 6,807,089
Trainable params: 6,807,089
Non-trainable params: 0
_________________________________________________________________
```

Evaluation

BLEU

- **B**ilingual **E**valuation **U**nderstudy
- This was originally developed to measure machine translation
- The idea: the closer a machine translation is to a professional human translation, the better it is.
- Therefore BLEU looks as computing N-gram matches
- Doesn't check for the meaning of the sentences which is the major drawback of this metric

$$p_n = \frac{\sum_{n\text{-gram} \in hyp} count_{clip}(n\text{-gram})}{\sum_{n\text{-gram} \in hyp} count(n\text{-gram})}$$

BLEU

Requirement

It takes up three inputs arguments

- Translated Sentence/hypothesis
- Reference sentence one or more in a list format,
- Weights (specify what should be the n)

```
from nltk.translate.bleu_score import corpus_bleu
def evaluate_model(actual, predicted):
    print("BLEU-1 is {:.2f}".format(corpus_bleu(actual, predicted, weights=[1.0, 0.0, 0.0, 0.0])))
    print("BLEU-2 is {:.2f}".format(corpus_bleu(actual, predicted, weights=[0.5, 0.5, 0.0, 0.0])))
    print("BLEU-3 is {:.2f}".format(corpus_bleu(actual, predicted, weights=[0.3, 0.3, 0.3, 0.0])))
    print("BLEU-4 is {:.2f}".format(corpus_bleu(actual, predicted, weights=[0.25, 0.25, 0.25, 0.25])))


actual, predicted = predict(model, en_tokenizer, trainX, train_de_processed, train_en_processed)
evaluate_model(actual, predicted)
```

BLEU

How it works

Lets take an example to see how BLEU score works

Reference:- The cat is sitting on the mat

Machine translation:- On the mat is a cat

S.No	Unigram	Match	Bigram	Match	Trigram	Match	Four-gram	Match
1	on	1	on the	1	on the mat	1	on the mat is	0
2	the	1	the mat	1	the mat is	0	the mat is	0
3	mat	1	mat is	0	mat is a	0	mat is a cat	0
4	is	1	is a	0	is a cat	0		
5	a	0	a cat	0				
6	cat	1						
	P1	**0.83**	**P2**	**0.40**	**P3**	**0.25**	**P4**	**0.00**
	Weights	**0.25**	**Weights**	**0.25**	**Weights**	**0.25**	**Weights**	**0.25**

Figure 6: BLEU score for N-gram

$$BLEUScore = \sum_{i=1}^{N} W_i * P_i$$

Training Evaluation

```
source is [geh], target is [go], predicted is [go]
source is [hallo], target is [hi], predicted is [hi]
source is [gruss gott], target is [hi], predicted is [hi]
source is [lauf], target is [run], predicted is [run]
source is [lauf], target is [run], predicted is [run]
source is [potzdonner], target is [wow], predicted is [wow]
source is [donnerwetter], target is [wow], predicted is [wow]
source is [feuer], target is [fire], predicted is [fire]
source is [hilfe], target is [help], predicted is [help]
source is [zu hulf], target is [help], predicted is [help]
```

```
1 evaluate_model(actual, predicted)
```

```
BLEU-1 is 0.95
BLEU-2 is 0.93
BLEU-3 is 0.90
BLEU-4 is 0.76
```

Figure 7: Training result

Testing Result

Slide Type Slide

```
1 test_actual, test_predicted = predict(model, en_tokenizer, testX, test_de_processed, test_en_processed)
2 evaluate_model(test_actual, test_predicted)
```

```
source is [ich erzahle gern witze], target is [i like to tell jokes], predicted is [i like to]
source is [ich gewinne gerne preise], target is [i like to win prizes], predicted is [i like to start]
source is [ich schreibe hier gerne], target is [i like to write here], predicted is [i love to]
source is [ich gehe gern allein spazieren], target is [i like walking alone], predicted is [i like to walk]
source is [ich mag dich sehr], target is [i like you very much], predicted is [i really like you]
source is [mir gefallt deine einstellung], target is [i like your attitude], predicted is [i like your]
source is [mir gefallen deine ohrringe], target is [i like your earrings], predicted is [i like your]
source is [dein optimismus gefallt mir], target is [i like your optimism], predicted is [i like this juice]
source is [ich mag wie du denkst], target is [i like your thinking], predicted is [i like like you]
source is [dein kommentar hat mir gefallen], target is [i liked your comment], predicted is [i like your speech]
BLEU-1 is 0.51
BLEU-2 is 0.38
BLEU-3 is 0.31
BLEU-4 is 0.20
```

Figure 8: Testing result

Take home message

- Preprocessing the data
- Understood how neural machine translation works
- Implemetation and evalaution of model
- How keras makes things handy

Thank You !