AI Quantification of Language Puzzle to Language Learning Generalization

•••

Harita Shroff

May 2020

Overview

- Introduction
- Background
- Model Architecture and Dataset
- Results
- Conclusions
- Q & A

Introduction

Language Learning

- The evolution of internet has left every aspect of our lives connected to it.
- Communication is also made easier with internet.
- Language was a barrier in earlier days but not any more.
- Online language learning platforms are here to HELP!

Popular Language Learning Platforms

verbling

Berlitz

Learn a language. Open your world.

Babbel

lingoda

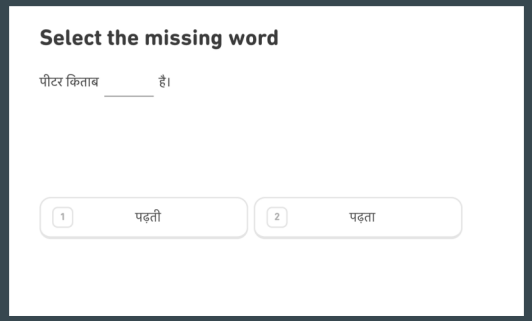
Busuu

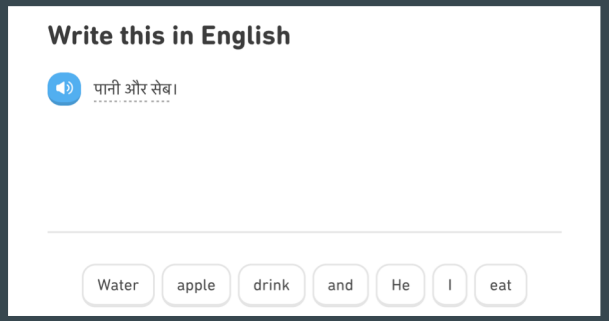
duolingo

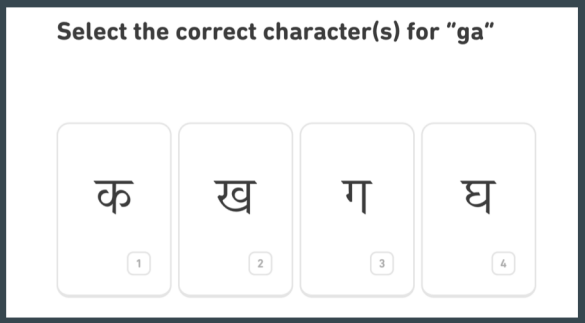
Rosetta Stone

memrise

Language Learning Activities





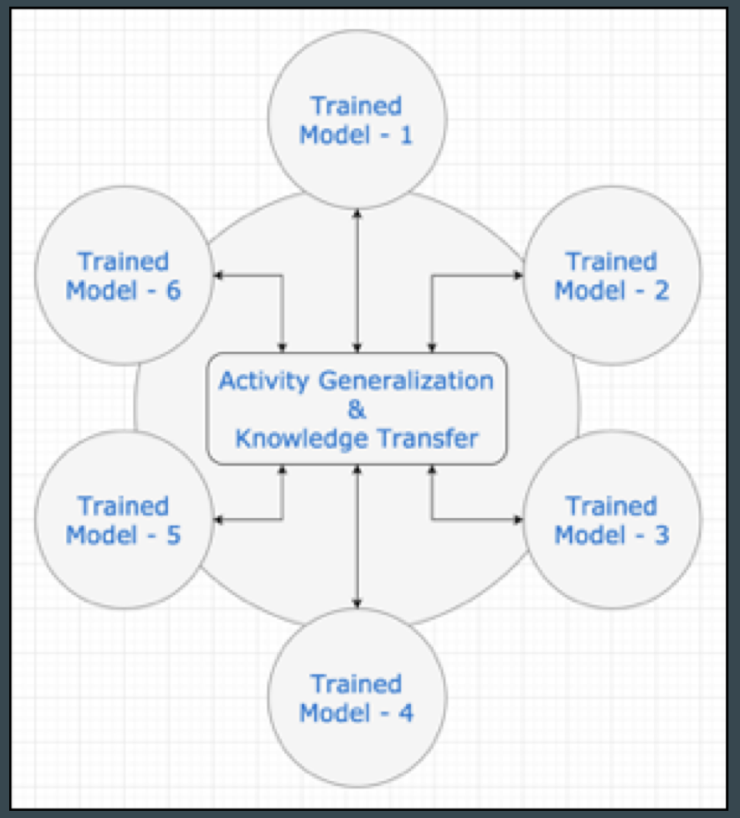


Problem Statement

- Are these platforms effective for a user who wants to learn a new language?
- Are different exercises going to help user learn a new language?
  - If so, how much?
- Can we generalize these activities to measure the transfer learning between them?

Activity Generalization Concept

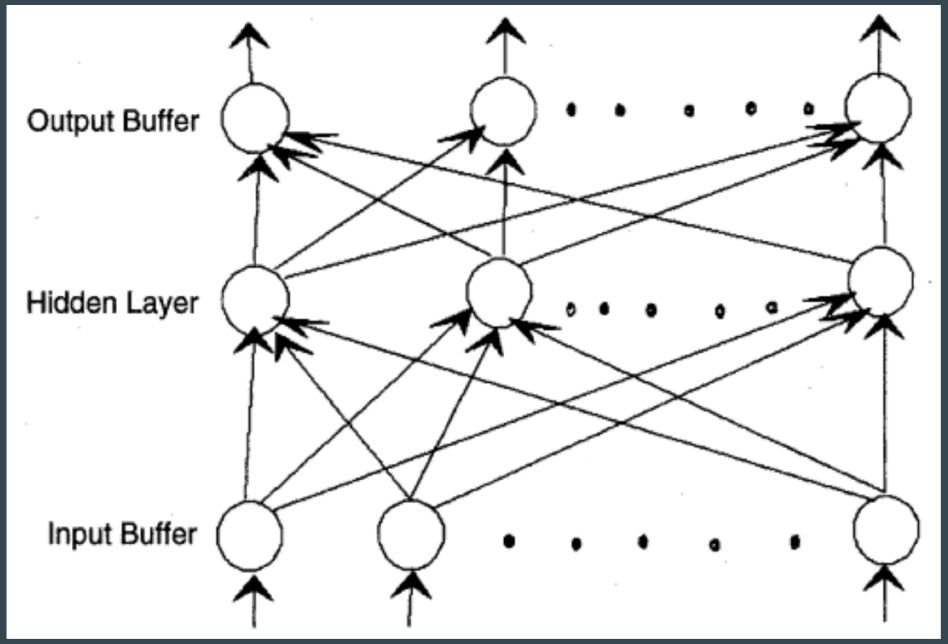
- Use of AI through neural network to create different models representing language learning exercises.
- Cross validation of all models on same data.
- Check generalization and knowledge transfer between activities with validation accuracy.



Background

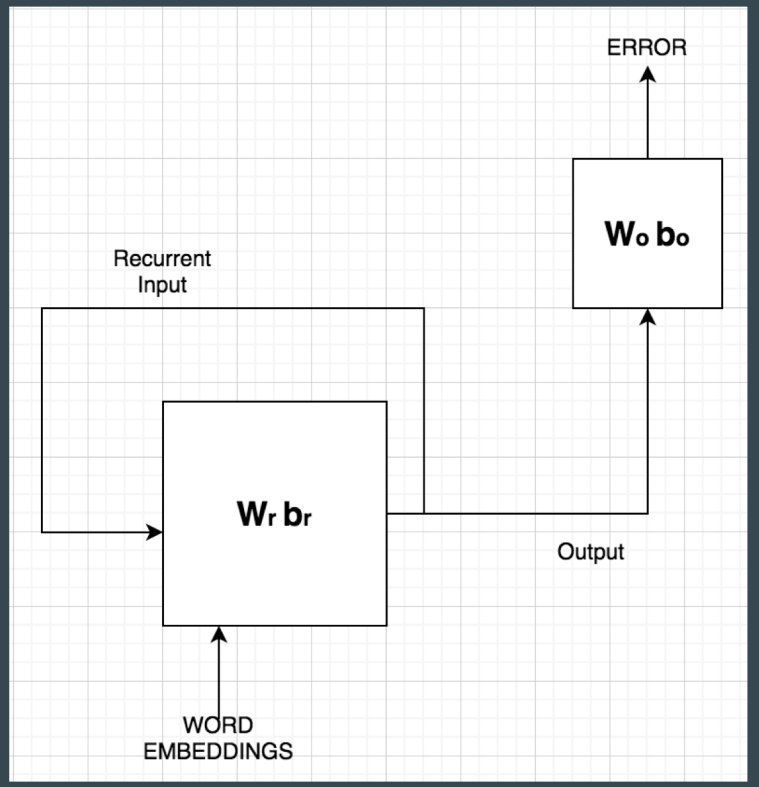
Neural Networks

- Type of Neural Networks:
  - Feed forward Neural Network
    - Multilayer perceptrons
  - Convolution Neural Network
  - Recurrent Neural Network
- Different layers in Neural Network:
  - Input Layer
  - Hidden Layer(s)
  - Output Layer
- Layers based Neural Networks:
  - Fully connected Neural Network.
  - Partially connected Neural Network.



Recurrent Neural Network (RNN)

- Type of Neural Network that allows its own output to be used as recurrent input.
  - Directed Cyclic Graph
- Opposite to Feed Forward Neural Network in some sense.
- Specially used when previous input of the model needs to have influence arbitrarily in future.
  - Language processing is the main application.



Back Propagation - Window size

- Current RNN approach is not practical for a large set of text.
  - I.e., if a prediction for the last word in the text is not the expected value then we will have to change all weights and biases for the last word in the corpus in a backward pass.
  - Brute force can be used to cut off backward calculations after several iterations.
- Iteration number after which calculations are cut off is called a **window size**.
- Window size is a system hyperparameter.
- Below window size is 3 and batch size is 2.

STOP	It	is	a	small	world
but	I	like	it	that	way

STOP	It	is
but	I	like

a	small	world
it	that	way

Vanishing Gradient Problem

- In neural networks, gradient descent is a mechanism to pass error in a backpropagation phase to update input weight and biases for model to improve.
  - If for a reason, the gradient becomes very small which results in poor or no learning, then this scenario is identified as **Vanishing Gradient Problem**.
- Often seen in Recurrent Neural Network.
  - Information from previous time points is used as input for future time points.
  - Through backpropagation weights are updated on each neuron responsible for the specific output.
  - During this backpropagation gradient becomes small and small with each iteration resulting in Vanishing Gradient Problem.

Gated Recurrent Unit (GRU)

- RNN suffers from short term memory issue.
  - If a long sentence is fed into RNN, there is a chance that earlier part of sentence will not contribute to any learning.
- GRU or LSTM are used to mitigate that issue using gates.
  - Update gate controls the information that flows into memory.
  - Reset gate controls the information that flows out of memory.
- GRU helps by saving information which holds more influence over the result.
  - This way the information or words which has more influence are stored and not forgotten.
- It helps solve the vanishing gradient problem.

GRU VS. LSTM

- Compared to LSTM, GRU has less number of inputs.
  - Less number of input attributes to less memory usage.
  - Execute and train faster.
- Dataset with smaller sentences should use GRU for quick processing.
- LSTM is more accurate on the dataset with longer sentences.

Language model

- A language model is the probability distribution over all strings in the corpus.
  - Language translation programs need to identify differences between sentences in two different languages.
  - Language model helps with this idea.
- Sentences can be broken into words and probabilities of each word following the previous can be counted.
  - Each word's probability is calculated considering that all previous words are present in the sentence.
  - Not a practical approach as sentences could be long.

Word Embeddings

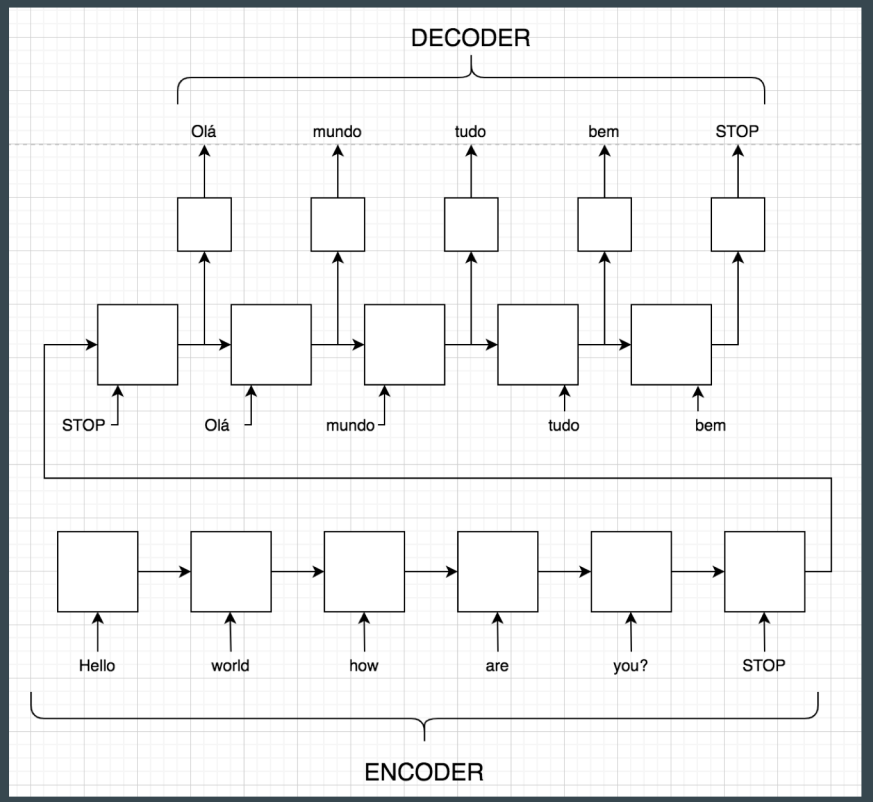
- Given a word in the corpus, a probability distribution of all other words following the previous one can be created in a table.
- Using a deep network, a reasonable probability distribution can be calculated over possible next word for the word W_i.
- Deep networks work with floating numbers, each word is mapped to a vector of float, that is called **Word Embeddings**.
  - Each embedding is initialized as a vector of e floats.
  - For V words in corpus, word embedding array E can be of size |V| x e.

Sequence-to-sequence Learning

- It is a technique to map a sequence of symbol to the other when mapping symbols individually is not possible.
  - Machine translation is one of the key application.
- It consists of two RNN units.
- Popular RNN cell choices are GRU or LSTM.
- Total two passes:
  - Encoding: First phase of sequence-to-sequence model. Goal is to summarize the sentence.
  - Decoding: Second phase of sequence-to-sequence model. Goal is to predict the sequence of symbols.

Sequence-to-sequence Paradigm

- Backpropagation through time.
  - All square boxes in encoder are same RNN unit, but at successive time.
  - Same applies to decoder.
- Encoder creates a sentence summary and passes it down to decoder.
  - Multiple ways to create sentence summary.
    - Summation of all encoder states
    - Average value of all encoder states



Levenshtein Distance

- It is a measure of distance between two strings. If the distance is measured between source string (S) and target string (T),
  - The distance between S and T will be the number of substitutions, insertions, and/or deletions required to transform S to T.
- The more different the strings are, the greater the Levenshtein distance will be.
- It is also known as edit distance.
- Examples:
  - Levenshtein distance between “test” and “tent” is 1.
  - Levenshtein distance between “honda” and “hyundai” is 3.

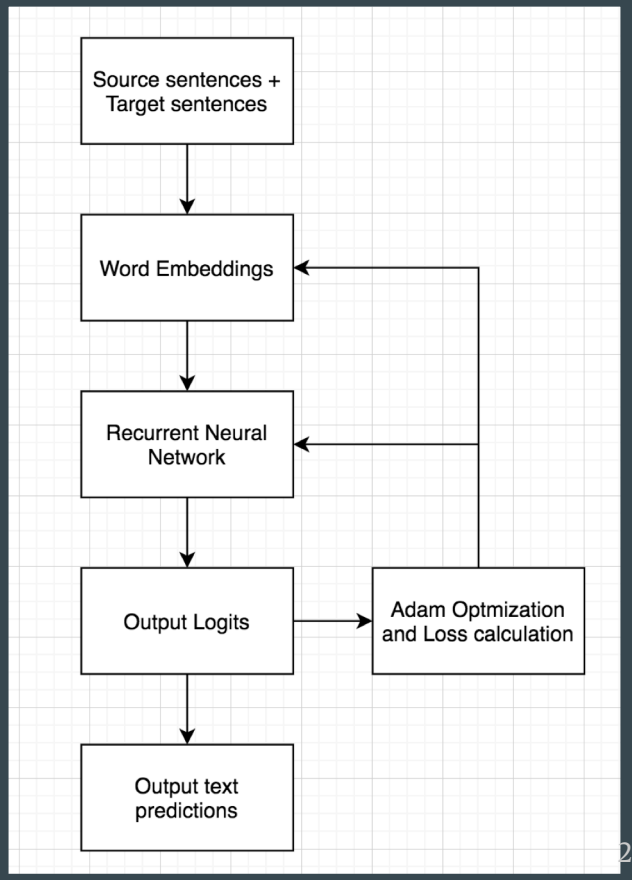
Levenshtein Distance Application

- Well known Levenshtein distance applications:
  - String matching
  - Spelling check
  - Speech recognition
  - DNA analysis
  - Plagiarism detection
- Equation used to calculate accuracy:
  - 1 - (Levenshtein distance / Target string length)
  - Dividing Levenshtein distance by length provides the part of the string which needed substitution, deletion and/or addition.
  - Subtracting this from 1, provides the amount of string which was predicted correctly.

Model Architecture & Dataset

Model Architecture

- Source and target sentences are processed and vectorized for the input into sequence-to-sequence model.
- Word Embeddings will be generated from the vectorized dataset.
- Recurrent Neural Network with GRU cells will be used to processes the data.
- Adam optimizer and loss will be calculated during the training of the model.
- Translated/Predicted sentence will be the output from the model.



Dataset

- Two datasets are used for this project.
  - Duolingo's Simultaneous Translation And Paraphrase for Language Education (STAPLE).
    - This dataset contains 4000 small English sentences followed by multiple Portuguese translations for the same sentence.
    - Each Portuguese translation is assigned a number indicating the weight based on user response rates.
  - Tatoeba dataset
    - It contains large amount of sentences and their translations in many different languages.
    - English to Portuguese translation has a total of 122,443 sentences and their translations.

Dataset statistics

- Total English sentences and their Portuguese translation: 126,443
- Unique English words in dataset: 24,696
- Unique Portuguese words in dataset: 36,963
- Maximum English sentence length: 35 words
- Maximum Portuguese sentence length: 33 words
- Average English and Portuguese sentence length: 6 words
- 95% of dataset contains sentences with fewer than 10 words.

Data Processing Flow

- A total of 126,443 sentences were partitioned in 2 sets.
  - Training dataset (85%, 107,443 sentences)
  - Testing dataset (15%, 19,000 sentences)
- Create word to integer map for sentence vectorizing.
  - Same file is used to store English and Portuguese words.
  - Each word is given an index.
  - Special words are also initialized to indicate:
    - End of Sentence (EOS)
    - Unknown Word (UNK)
    - Padding (PAD)

Data Processing Flow (cont'd)

- Word to integer map for entire dataset will be stored in a serialized python object format.
- For each model, training and testing sentences remain same.
- Based on the model definition, combination of English and Portuguese sentences will be generated and stored in an NumPy format file.
  - These files contain vectorized sentences.
  - Encoder and Decoder inputs for training are stored in one file while testing data is stored in the other.

Different Models for Generalization

MODEL NAME	Description	Corresponding activity
no-random-words	English sentence + shuffled Portuguese translation	Mark the correct meaning, Match the pair
5-fixed-random-words	English sentence + shuffled Portuguese translation + 5 random words	Select the character
10%-random-words	English sentence + shuffled Portuguese translation + 10% random words	Match the pair, Select the character
fill-in-the-blank	English sentence + shuffled Portuguese translation - 1 Portuguese word	Fill-in-the-blanks, Select the missing word
20%-random-words	English sentence + shuffled Portuguese translation + 20% random words	Match the pair, Select the character
no-portuguese-translation	English sentence	Write a sentence in foreign language

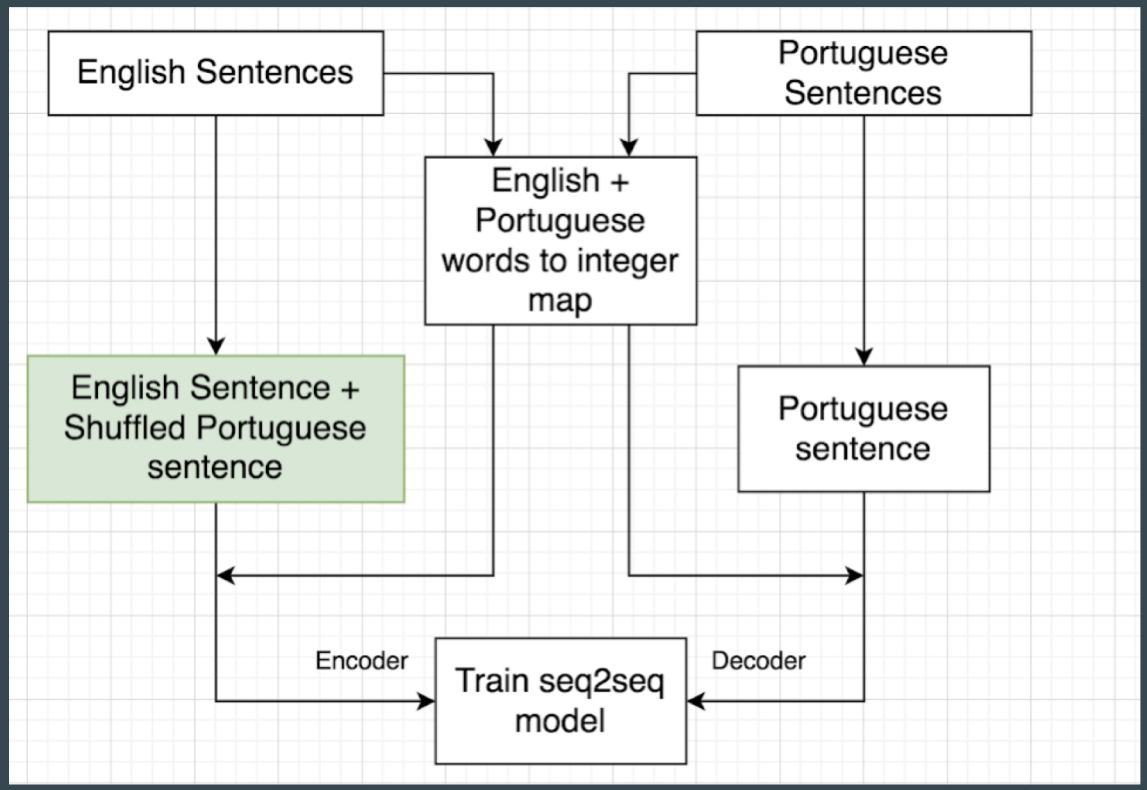
Data Processing for Model “no-random-words”

Example:

English Sentence: “can i help you”

Portuguese Translation: “posso ajudála”

Encoder Input: “can i help you ajudála posso”



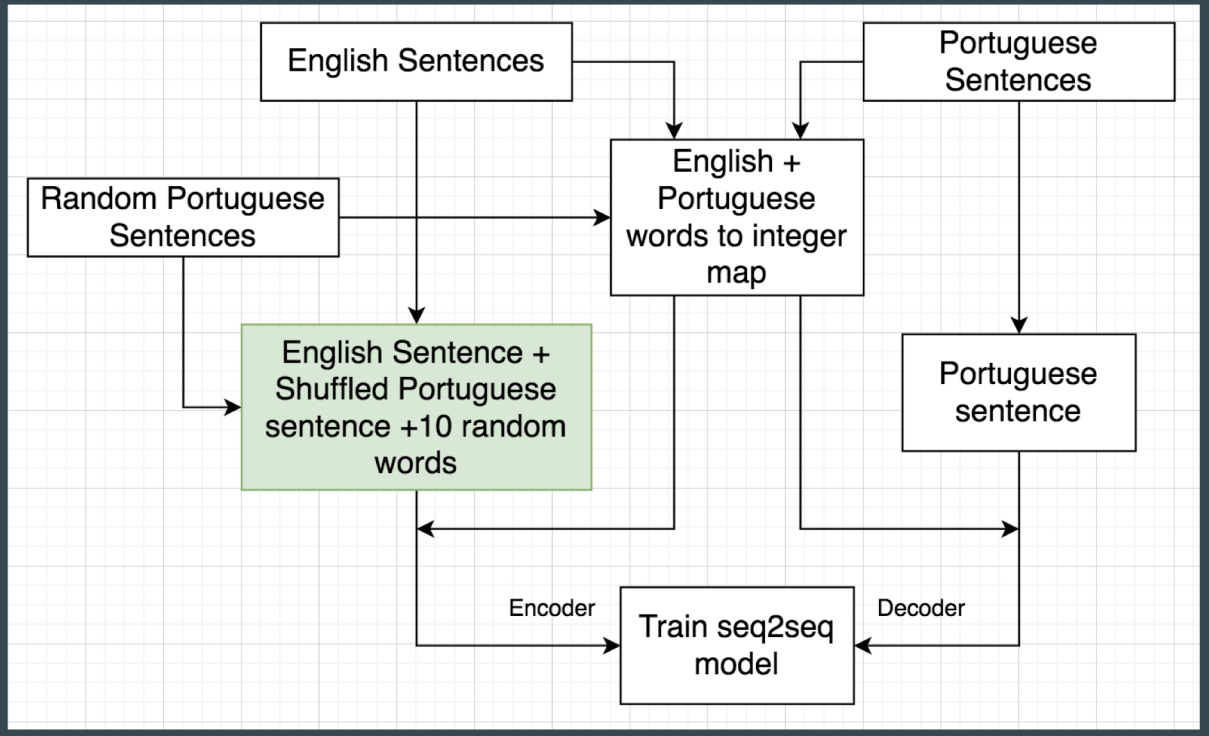
Data Processing for Model “5-fixed-random-words”

Example:

English Sentence: “can i help you”

Portuguese Translation: “posso ajudála”

Encoder Input: “can i help you temperatura roque pegaste escovando ajudála colocou posso”



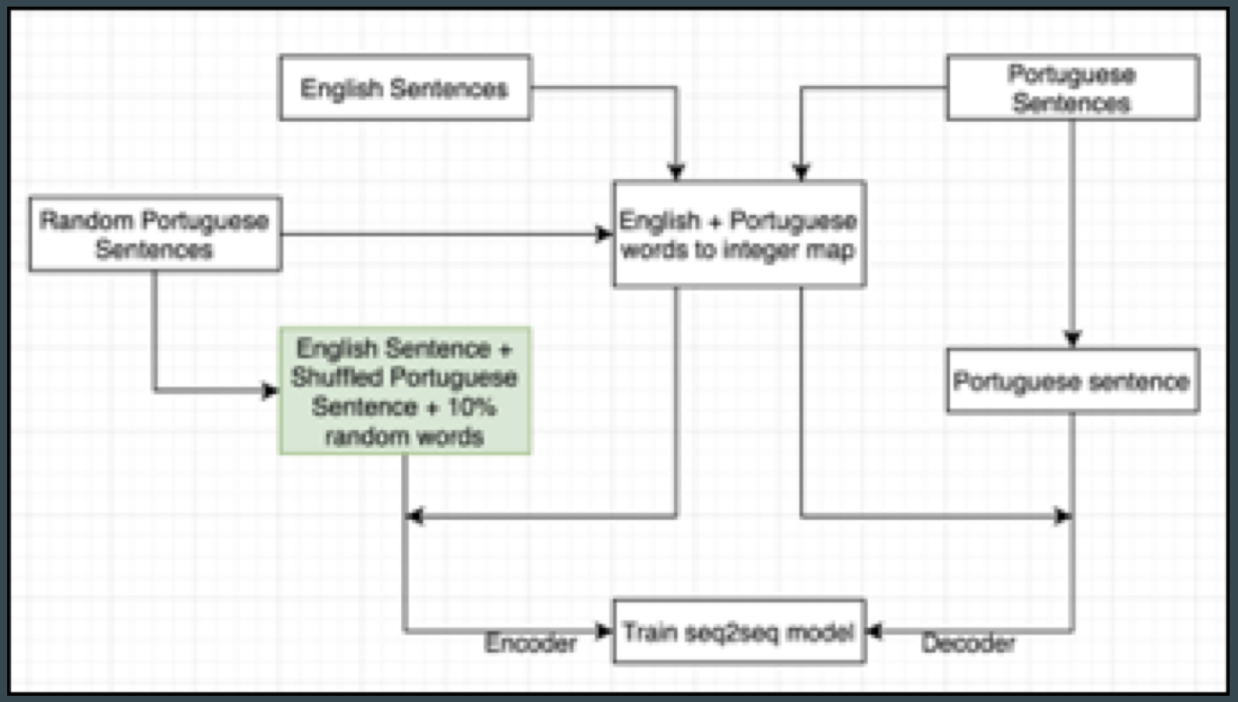
Data Processing for Model “10%-random-words”

Example:

English Sentence: “can i help you”

Portuguese Translation: “posso ajudála”

Encoder Input: “can i help you ajudála prossiga posso”



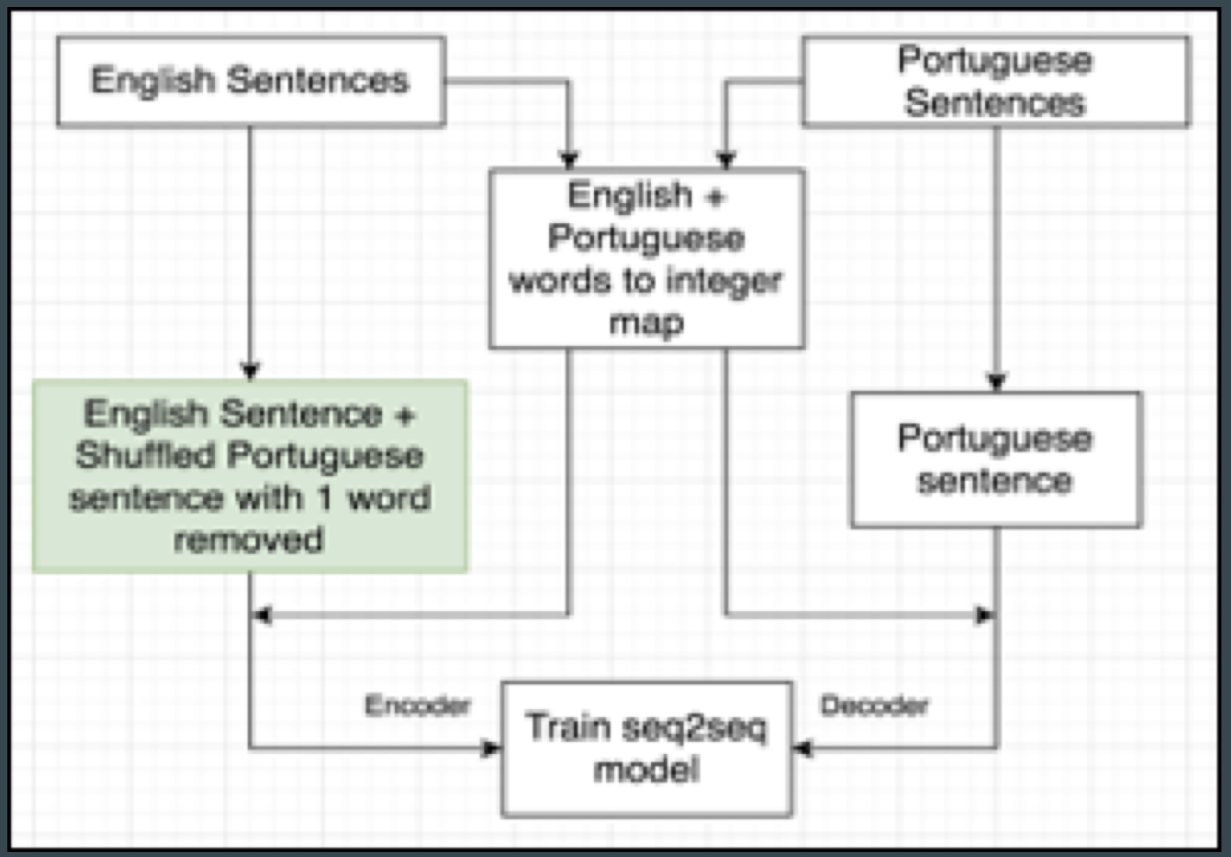
Data Processing for Model “fill-in-the-blanks”

Example:

English Sentence: “can i help you”

Portuguese Translation: “posso ajudála”

Encoder Input: “can i help you ajudála”



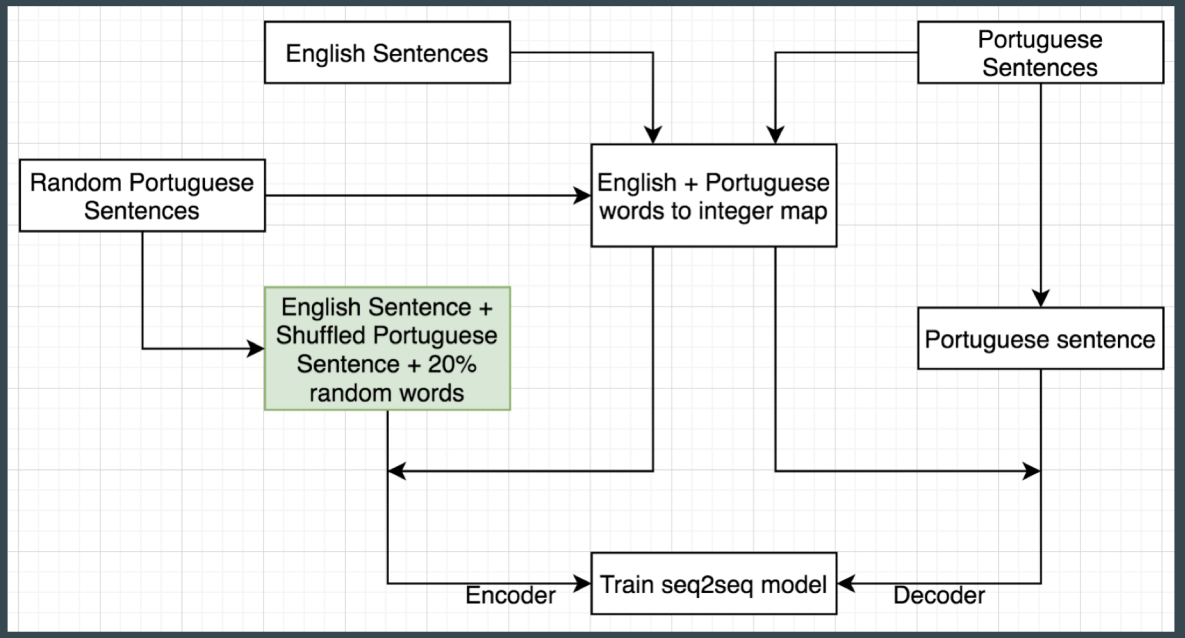
Data Processing for Model “20%-random-words”

Example:

English Sentence: “can i help you”

Portuguese Translation: “posso ajudála”

Encoder Input: “can i help you empobreceu posso utilizariam ajudála”



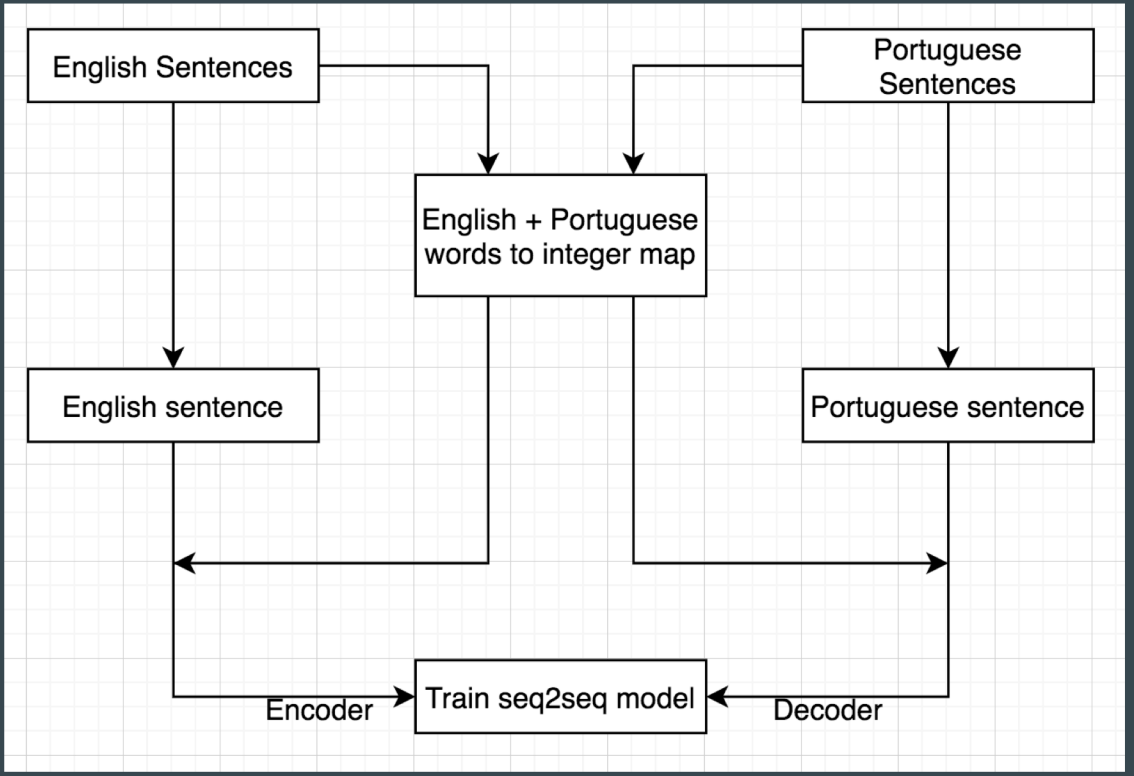
Data Processing for Model “no-portuguese-translation”

Example:

English Sentence: “can i help you”

Portuguese Translation: “posso ajudála”

Encoder Input: “can i help you”



System hyperparameter

- Total Epochs: 40
- Batch size: 256
- RNN size: 128
- Encoder & Decoder Embedding size: 200
- Learning rate: 0.001
- RNN Layers: 2

Results

Training Result

MODEL NAME	TRAINING ACCURACY	TRAINING LOSS
no-random-words	98%	0.01
5-fixed-random-words	99%	0.01
10%-random-words	99%	0.01
fill-in-the-blanks	95%	0.03
20%-random-words	98%	0.05
no-portuguese-translation	88%	0.16

Testing Results

		TEST MODELS					
		1	2	3	4	5	6
TRAINED MODELS	1	85%	61%	81%	80%	75%	39%
	2	77%	80%	78%	74%	79%	37%
	3	85%	66%	85%	80%	82%	38%
	4	81%	53%	74%	84%	68%	40%
	5	82%	70%	83%	77%	83%	39%
	6	42%	26%	37%	47%	34%	79%

LEGEND FOR THE TABLE

1. No random words
2. 5-fixed-random-words
3. 10%-random-words
4. fill-in-the-blanks
5. 20%-random-words
6. no-portuguese-words

- This table shows the accuracy based on Levenshtein Distance between expected and predicted strings.

Conclusion & Future work

Conclusion

- Average accuracy is ~75% for each model during cross validation.
  - Result shows that transfer learning happens between different exercises.
  - Activities does help a user who wants to learn a new language.
  - Activities can be generalized by using different neural network models.
- Last model with only English sentence had not so good performance with cross validation.
  - This points out that transfer learning does not happen efficiently between other exercises and this one.
  - This also supports the theory that conscious learning does not produce language competence.

Future work

- Use of attention mechanism can be added for sequence-to-sequence model.
- Audio exercises are not modeled in this project.
- Cross validation order is not taken into account for improving accuracy.

Questions!?