Contents

[Context 2](#_Toc143358984)

[Approach to the problem 2](#_Toc143358985)

[Why Modular ? 2](#_Toc143358986)

[Key highlights for Modules 3](#_Toc143358987)

[Loading and Merging data 3](#_Toc143358988)

[Preprocessing/ Cleaning 3](#_Toc143358989)

[Data Selection 3](#_Toc143358990)

[Vectorization 5](#_Toc143358991)

[Modeling 5](#_Toc143358992)

[RNN Models attempted 5](#_Toc143358993)

[LSTM Models attempted 5](#_Toc143358994)

[Code and key functions 5](#_Toc143358995)

[Highlights of some best practices followed 7](#_Toc143358996)

[Some tables/ plots for showing performance of models created 7](#_Toc143358997)

[Results of 3 word German to English translations 7](#_Toc143358998)

[Results of 3 word English to German translations 8](#_Toc143358999)

[Results of 5 word German to English translations 8](#_Toc143359000)

[Results of 5 word English to German translations 8](#_Toc143359001)

[Results of 7 word German to English translations 8](#_Toc143359002)

[Results of 7 word English to German translations 8](#_Toc143359003)

[Results of 10 word German to English translations 8](#_Toc143359004)

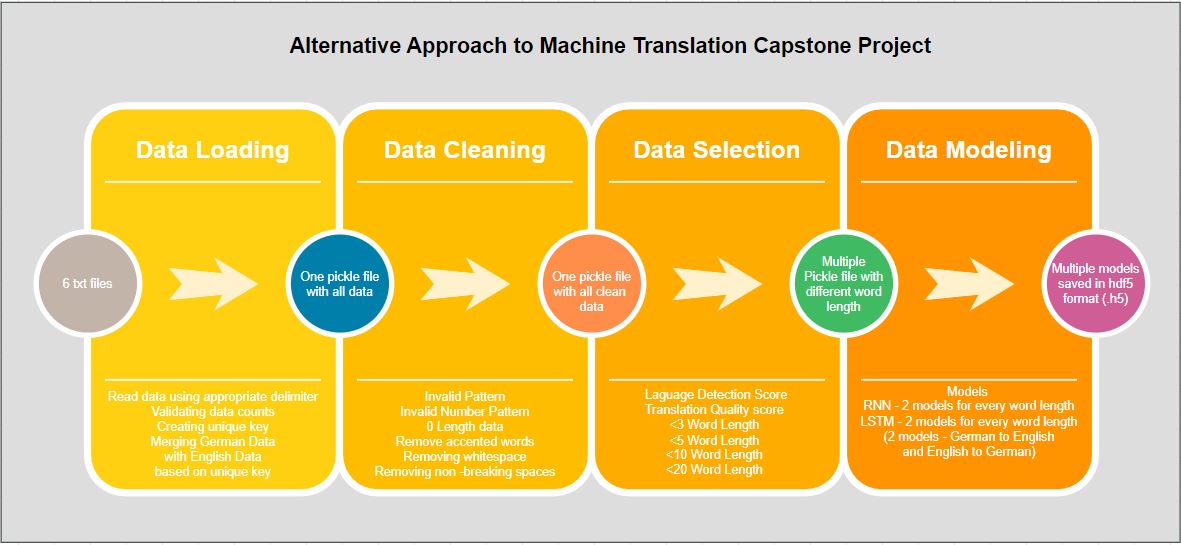
[Results of 10 word English to German translations 8](#_Toc143359005)

[Wishlist 8](#_Toc143359006)

# Context

This is alternative approach to the capstone MLT problem. It has separate modules for all the NLP tasks done to model out the solution. Currently being tried on word length 10 or less. However we can scale it.

# Approach to the problem



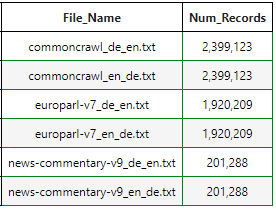
# Why Modular ?

* Team can distribute the modules and work in parallel.
* For a generic NLP model steps includes, loading, preprocessing, embeddings, modelling, fine tuning,
* In Modular approach output at every step can be evaluated and improved for inefficiencies without breaking the process.
* This also increases the overall model tenacity when every step has improved inputs and outputs
* Different combination of parameters and model flavors can be tried out better and can execute in parallel.
* The amount of effort done to achieve this surpasses the benefits in our current scope of milestone-1. As we move into milestone-2, the modularity achieved will allow us to look at different parts of data faster and produce a much more efficient and well co-ordinated effort from all team members.
* Also allows to model in case limited hardware resources are available. Let’s you break the steps in MT problem into small tasks done by separate modules. That was a big reason for the motivation in my case.

# Key highlights for Modules

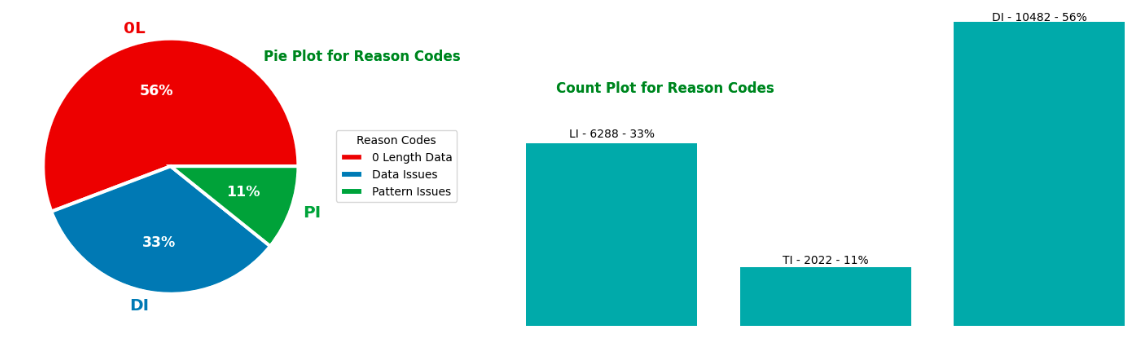
## Loading and Merging data

The data used is what was uploaded by GL and not from statmt.org



## Preprocessing/ Cleaning

* Data merged on key, so any amount of shuffling will still let's us back track to the origin of the row(source and location both)
* Data dropped is also strored
* Also clean data has been stored with remarks to know what has been cleaned, like nbsp if removed is stored as remarks for that row that it has been fixed



## Data Selection

**The observation over data were**

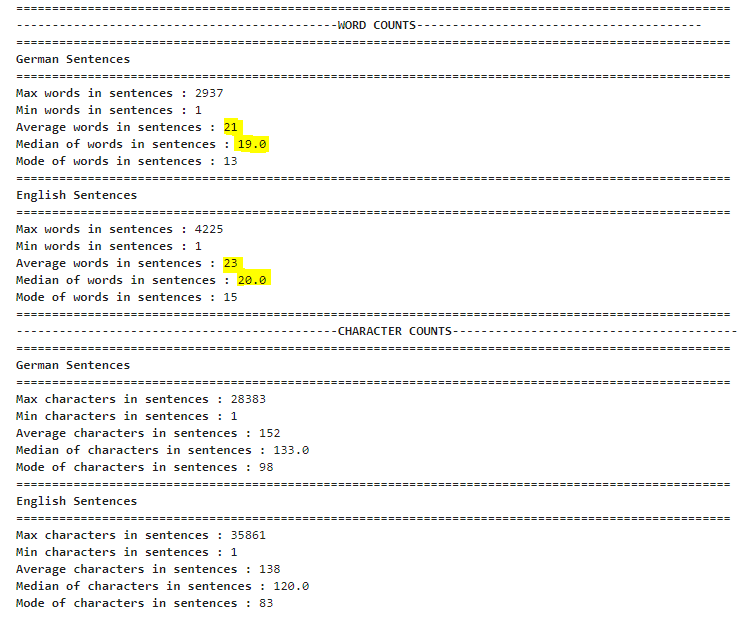
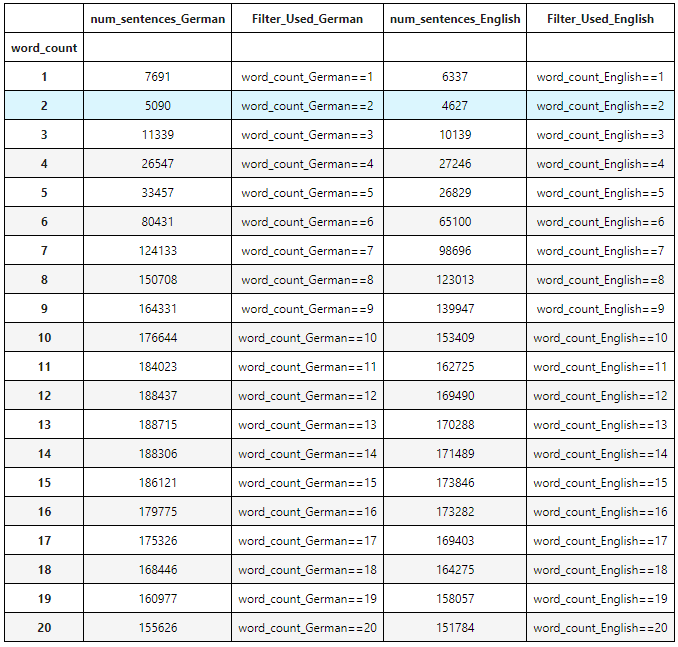
Word Counts - For both German and English most of the word length in sentences is around 20

**Conclusion based on that**

- Create one model on all data may not give good results. Will also need a much powerful machine

- Therefore have created models on small data with limited word lengths.

- However have built flexibility to scaled to bigger sentences with more words.

- Once the data is extracted we can also further weed off low quality translations and Data with low language detection score for German and English.- 

## 

## Vectorization

- Includes creating vocabulary, tokenization, data padding

- Has separate functions for each.

- Data consumed by RNN models is different from LSTM. In RNN input and output sequences are mostly kept same. The functions are paramterized enough to account for such needs to create vectorized data.

## Modeling

### RNN Models attempted

* Simple RNN with
* Simple RNN with embed
* RNN updated(GRU)
* RNN updated(GRU) with embeddings

### LSTM Models attempted

* Simple LSTM
* Stacked LSTM

## Code and key functions

|  |  |
| --- | --- |
| ipynb File name | Functions |
| data\_merging.ipynb |  |
|  | merge\_files() – merges the txt files given for the problem |
|  | count\_records() – count number of records in the files |
| data\_pre\_processing.ipynb |  |
|  | isvalid() – regex pattern to find special characters |
|  | invalid\_pattern() – regex to find defind pattern |
|  | pre\_process\_text() – cleans text |
|  | initialize\_pattern() – initialize valid acceptable patterns for both German and English |
| data\_selection.ipynb |  |
|  | calculate\_LT\_scores() - for transaltion and language quality |
|  | selection\_criteria() - defines condition to select data |
|  | get\_data() - extracts data based on a filter |
| data\_modeling.ipynb |  |
|  | create\_vocab() - return vocabulary for the corpus |
|  | create\_vector\_seq() - returns vector sequences |
|  | get\_word() - gets back sentences/word from vector sequences/ logits |
|  | pad\_data() - pads data based on maxlen passed |
| MLT\_RNN\_Model.ipynb |  |
|  | RNN\_1\_Simple() - defines RNNsimple model |
|  | RNN\_2\_GRU() - defines RNN updated(GRU) |
|  | RNN\_4\_Simple\_Embed() - defines RNN simple with embeddings |
|  | RNN\_4\_GRU\_Embed() - defines RNN simple with embeddings |
|  | add\_score() - stores the results of best hyperparameters for the model |
|  | read\_score() - reads the score from json file |
|  | highlight\_max() - highlights max score/column |
|  | highlight\_min() - highlights min score/column |
|  | model\_configuration() - configures model |
| MLT\_LSTM\_Model.ipynb |  |
|  | LSTM\_1\_Simple() - defines LSTM simple model |
|  | LSTM\_2\_Stacked() - LSTM stacked model |
|  | add\_score() - stores the results of best hyperparameters for the model |
|  | read\_score() - reads the score from json file |
|  | highlight\_max() - highlights max score/column |
|  | highlight\_min() - highlights min score/column |
|  | model\_configuration() - configures model |
| model\_evaluation.ipynb |  |
|  | get\_word() - gets back sentences/word from vector sequences/ logits |
| functions\_used.ipynb |  |
|  | initialize\_thresholds() |
|  | initlialize\_patterns() |
|  | is\_valid() |
|  | read\_frm\_pickle() - read from a pickle file |
|  | write\_to\_pickle() - writes to a pickle file |
|  | write\_to\_txt() - read from txt file |
|  | read\_frm\_txt() - write to a txt file |
|  | reclaim\_memory() - frees up memory |
|  | langdetect() - detection language score using |
|  | translation\_score() - scores quality of translation using transformer pipeline |
|  | cosine\_similarity() - checks siilarity between two same language text |
|  | all\_punctuation\_marks() - returns all unicode characters used across languages |
|  | invalid\_pattern() - regex pattern to identify defined invalid pattern |
|  | pre\_process\_text() - preprocess text based on functaionality define is data\_preprocessing.ipynb file |
|  | move\_records() - move records from one dataframe to another with remarks |
|  |  |
| data\_visualizations.ipynb | For visualizations/ plots/ will carry this forward in milestone-2 |
|  |  |
| .pkl files for data (some important ones) |  |
|  | all\_data\_capstone.pkl - stores all data |
|  | all\_data\_cleaned.pkl - stores clean data |
|  | data\_not\_selected.pkl - stores data which has been dropped |
|  | English\_3\_word\_clean.pkl, German\_3\_word\_clean.pkl - small datasets like these |
| .h5 files for models |  |
|  | German\_English\_3\_LSTM.- Best German to English LSTM model with 3 words |
|  | English\_German\_3\_LSTM.h5 Best English to German LSTM model with 3 words |

### 

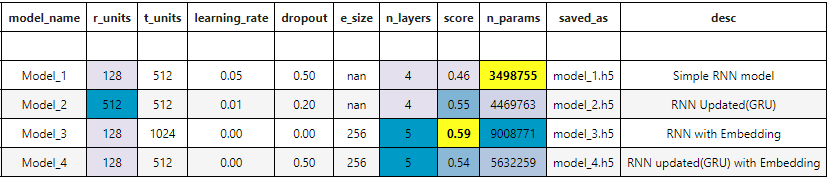
## Highlights of some best practices followed

* Hierachical formatting of topics, Code commenting
* Exception handling
* Common functions used are kept in a ipynb file which is referred across code base
* Pickle files for data and HDF5 file for models are used. Also allow us to execute any module at anytime since data is always available.
* Best model is chosen after hyper parameter tuning using keras\_tuner

# Some tables/ plots for showing performance of models created

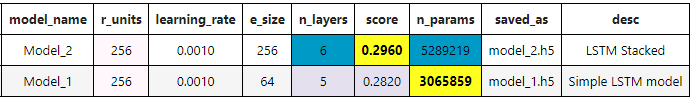
## Results of 3 word German to English translations

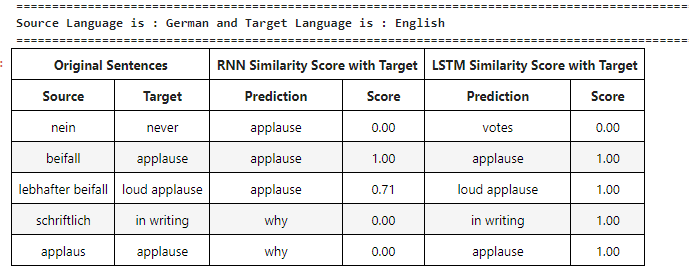
RNN Model Comparison

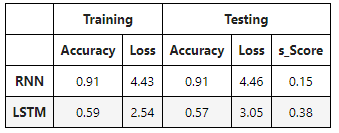


Model\_3 is performing best

LSTM Model Comparison

  
Model\_1 is performing better





\*s\_Score – Average Cosine similarity scores across all sentence predictions

# TODO <Have to add below scores. Will add before submission>

## Results of 3 word English to German translations

## Results of 5 word German to English translations

## Results of 5 word English to German translations

## Results of 7 word German to English translations

## Results of 7 word English to German translations

## Results of 10 word German to English translations

## Results of 10 word English to German translations

# Wishlist

* Simple character based model
* Use of more regex to find valid number patterns
* Explore aspects to understand how we can improve model learning for Grammar and punctuation during training
* Click based UI which can allow to plug and play different module parameters and understand the model learning capability
* Also want to try if we can have an executable out of python notebooks