Automated Language Proficiency Assessment

A Presentation by -

Shanmukha Sri Harsha Anivilla
Manikanta Sanjay Veera
Ashish Agarwal

What is Automated Language Proficiency Assessment (ALPA) and why do we need it?

- ALPA is an application, that scores the essay input based on certain parameters.
- The range of the scores is from 0 to 5.

WHY?

- It helps the new english learners get auto graded and improve their skills.
- Helps with professional write-ups.
- Reduces the effort involved in manual grading.

Basis for Scoring

- We scored the essays based on the following criteria:
  - Grammar
  - Cohesion
  - Syntax
  - Vocabulary
  - Phraseology
  - Conventions
- We find the scores for each of the 6 categories.

The Approach

- Such an application can be developed using Natural Language Processing.
- Various models can be used to train for this purpose.
- Models like Long Short-Term Memory (LSTM), Simple Dense Neural Networks, and transformers models like RoBERTa and DeBERTa can be used to automatically score essays and articles.

Libraries and Resources Used

- NLTK: From nltk we used a suit(tokenize, corpus, tag) of libraries for Natural Language processing.
- Transformers: This is used to take in the pre-trained model from huggingface, and re-train using our train dataset.
- Tensorflow.keras.preprocessing: For pad sequences and for Tokenizer functions.
- React with Node JS: for the frontend design
- Flask: for the APIs development
- NGrok: for server deployment
- Pandas: for dataframe manipulation and accessing.
- Numpy: for matrix and array operations.
- Matplotlib and Seaborn - for plots

Implementation Details

Training:

- A pre-trained model and the corresponding tokenizer is first pulled.
- The text input rows are tokenized using the model's existing tokenizer.
- Tensorflow keras is used to add layers - Dense layers and transformers model layer.
- Model is built.
- Model is saved.
- Model is predicted for test split.
- MCRMSE Losses are calculated for the test split - lower loss indicates better model.

(Continued)

API:

- Flask is used to expose endpoints for the following:
  - **Model Loading** - Loads the trained model saved during the testing phase.
  - **Model Testing** - calculates loss for test split
  - **Evaluating proficiency for sample essay** - takes in sample essay input and calculates proficient for 'cohesion', 'syntax', 'vocabulary', 'phraseology', 'grammar' and 'conventions'.

(Continued)

Frontend:

Buttons are created for the following:

- **Load Roberta** - Calls model loading API for Roberta.
- **Test Roberta** - Calls model testing API for Roberta to get the loss scores.
- **Load LSTM** - Calls model loading API for LSTM.
- **Test LSTM** - Calls model testing API for LSTM to get the loss scores.
- **Load DeBERTa** - Calls model loading API for DeBERTa.
- **Test DeBERTa** - Calls model testing API for DeBERTa to get the loss scores.

For proficiency calculation, the following is done:

- A textbox is provided wehere user can paste their sample essay.
- A button is provided that will call the API to get the proficiency scores for 'cohesion', 'syntax', 'vocabulary', 'phraseology', 'grammar' and 'conventions'.

For all the buttons provided above, results are shown in a popup.

The Models that we Used

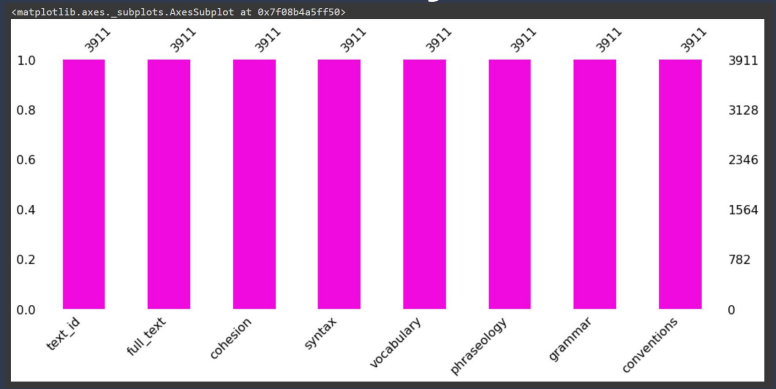
We have used 3 models to score the essays -

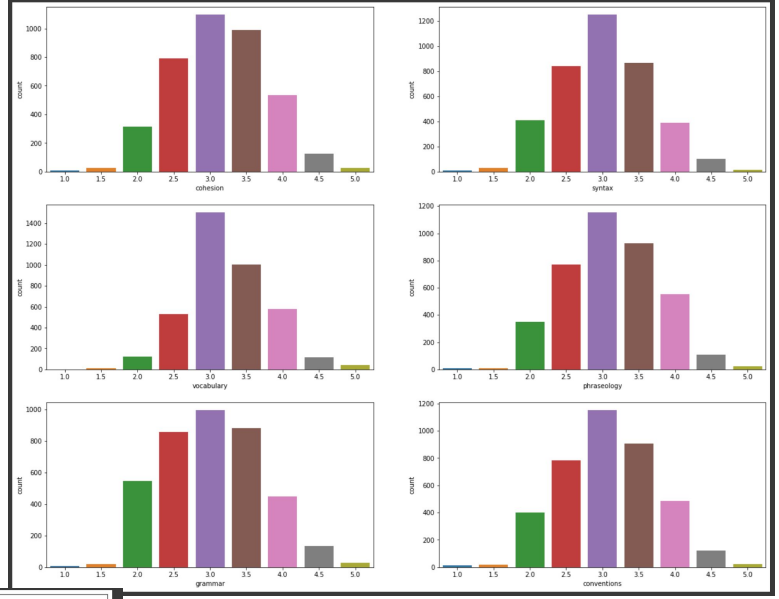
- Long Short-Term Memory
- DeBERTa V3 Small
- RoBERTa Large

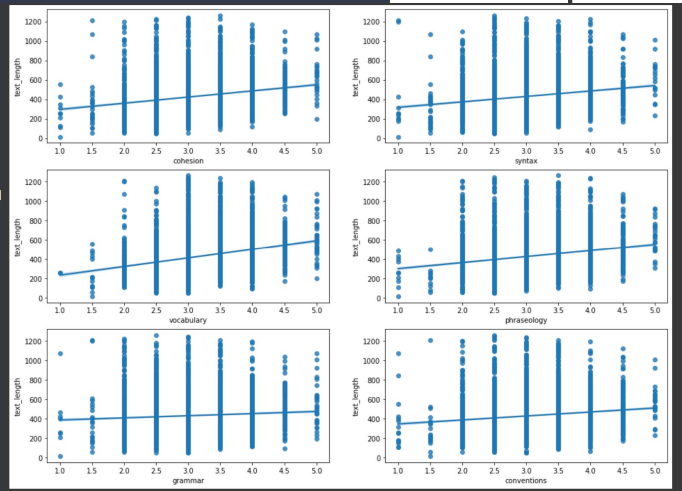
Dataset Description

- The training dataset consists of 3911 rows.
- Each row contains of a text, that is the essay and the score for each of the 6 categories.
- There are no missing or null values in the dataset.

Exploratory Data Analysis







Long Short-Term Memory (LSTM)

- It is an artificial neural network.
- It can process entire sequences of data such as speech or video.
- The LSTM architecture aims to provide a short-term memory for RNN that can last thousands of timesteps, thus "long short-term memory".
- An RNN using LSTM units can be trained in a supervised fashion on a set of training sequences.

How we implemented LSTM

- We have used the Tensorflow Tokenizer to preprocess and tokenize the text.
- First, We removed the extra spaces from the text.
- We calculated the word length and then subsequently the max word length of the training text.
- And converted the text to sequences using the text_to_sequences function of the tokenizer.

LSTM

- Finally, we performed padding on the text, to transform the sequences to a numpy array.
- This padded text is used to train the model.

DeBERTa

- DeBERTa - Decoding-enhanced BERT with Disentangled Attention.
- This is an advanced version of the BERT and RoBERTa models.
- In this, each word is represented using two vectors that encode its content and position, respectively.
- The attention weights among words are computed using disentangled matrices on their contents and relative positions.

DeBERTa

- An enhanced mask decoder is used to replace the output softmax layer to predict the masked tokens for model pretraining.
- We have used the DeBERTa v3 small model here.
- To preprocess and tokenize this text, we have used the transformers' AutoTokenizer.

Roberta - base

- Pretrained model on English language using a masked language modeling (MLM) objective.
- RoBERTa is a transformers model pretrained on a large corpus of English data in a self-supervised fashion. This means it was pretrained on the raw texts only, with no humans labelling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts.

How we implemented Roberta

- We have used the RobertaTokenizer for tokenization.
- We load the pretrained roberta-base model from transformers module.
- We then instantiate the model and compile it with adam optimisation.
- Later, we fit the model using our train data.
- We perform bert encoding and predict the model on user test data.

Test Set Analysis

Model Roberta

MCRMSE Score: 0.5346775230571287

Individual MCRMSE Loss Scores

cohesion	syntax	vocabulary	phraseology	grammar	conventions
0.7416198487095663	0.4609772228646444	0.4472135954999579	0.5477225575051661	0.5361902647381804	0.4743416490252569

Model LSTM

MCRMSE Score: 0.6948373246967169

Individual MCRMSE Loss Scores

cohesion	syntax	vocabulary	phraseology	grammar	conventions
0.6224949798994366	0.689202437604511	0.6422616289332564	0.689202437604511	0.8366600265340756	0.689202437604511

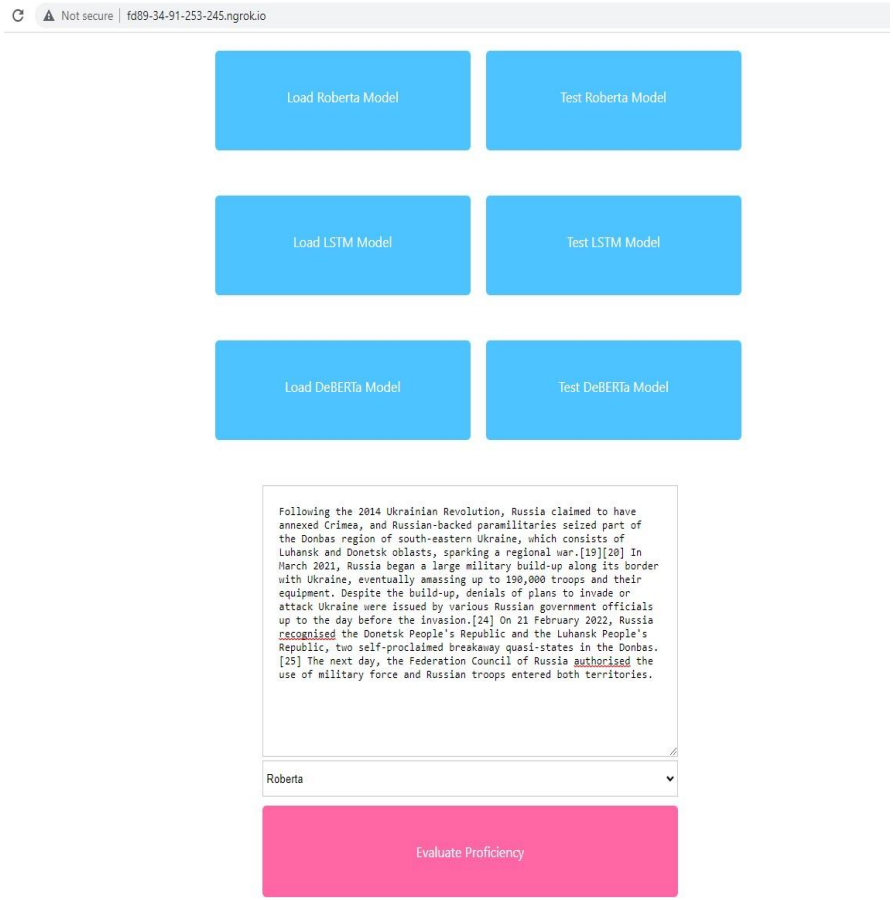
Model DeBERTa

MCRMSE Score: 0.33823532827860325

Individual MCRMSE Loss Scores

cohesion	syntax	vocabulary	phraseology	grammar	conventions
0.3711537444790451	0.31134992453861954	0.31339158526400435	0.3282995769794685	0.3580346009957754	0.3471825374147068

Frontend UI


Not secure | fd89-34-91-253-245.ngrok.io
Load Roberta Model
Test Roberta Model
Load LSTM Model
Test LSTM Model
Load DeBERTa Model
Test DeBERTa Model
Following the 2014 Ukrainian Revolution, Russia claimed to have annexed Crimea, and Russian-backed paramilitaries seized part of the Donbas region of south-eastern Ukraine, which consists of Luhansk and Donetsk oblasts, sparking a regional war.[19][20] In March 2021, Russia began a large military build-up along its border with Ukraine, eventually amassing up to 190,000 troops and their equipment. Despite the build-up, denials of plans to invade or attack Ukraine were issued by various Russian government officials up to the day before the invasion.[24] On 21 February 2022, Russia recognised the Donetsk People's Republic and the Luhansk People's Republic, two self-proclaimed breakaway quasi-states in the Donbas.[25] The next day, the Federation Council of Russia authorised the use of military force and Russian troops entered both territories.
Roberta
Evaluate Proficiency

Results, Conclusion and Future Work

- Based on the losses, DeBERTa performs the best.
- The frontend UI application is successful at auto grading the user input essays based on the 3 models.
- In the future, we wish to add feedback to user based essays.
- We can also give different results based on different audience in the future:
  E.g. Beginners will be scored differently compared to professionals.