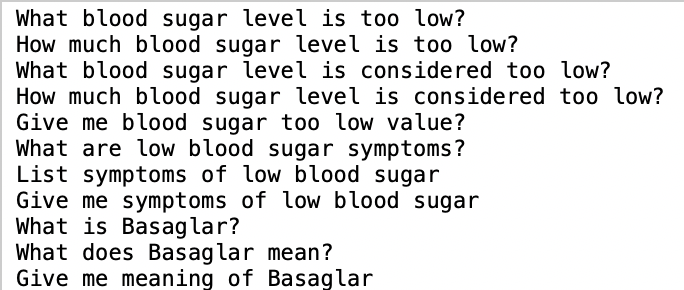
* Model selection
  + Multiple models had to be chosen from, based on different parameters such as:-
    - Model size
    - Complexity of architecture
    - Ease of pre-processing on device
    - Layer compatibility
    - Model conversion
  + The reason of choosing custom built model is:
    - Smaller in size
    - Customizable
    - Faster
    - Simpler pre-processing
    - Use of lesser dependencies
  + There are 3 kinds of ML models involved:-
    - Auto-correct model
    - Intent-classification model
    - NER-tagging model
* Development
  + General requirements for each model:
    - Data collection
    - Data pre-processing
    - Defining neural architecture
    - Training the model on processed data
    - Testing the model
    - Converting into
      * Coreml for IOS
      * TFlite for Android
  + Data collection
    - The collected data needs to be in a pre-defined structured format so that the data pre-processing doesn’t falter.
    - Each model has a different requirement of the data input feed.
    - Auto-correction model:-
      * Sample set:

|  |  |  |
| --- | --- | --- |
| emgality | olumiant | trulicity |
| m gallery | my event | truly city |
| in gallatin | all lament | the city |
| m garrity | my element | truly sweetie |
|  |  |  |

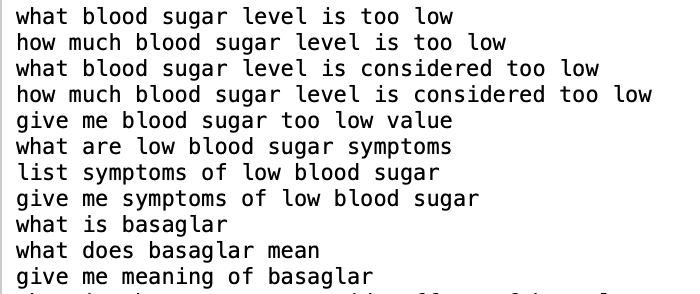
* + - * The highlighted line demarcates the actual tags for the drug names.
      * The corresponding rows in each columns showcase the mis-spells for the same drug name for voice input.
    - Intent-classification model
      * Sample set:-

|  |  |  |
| --- | --- | --- |
| log\_medication | content\_search\_details | content\_search\_pricing |
| Add Olumiant for 15th November | What blood sugar level is too low? | What is the price of Emgality? |
| Add one dosage for Olumiant for day after tomorrow | How much blood sugar level is too low? | Show the price of Emgality |
| Can you add my Olumiant medication for 10 days | What blood sugar level is considered too low? | What is the cost of Emgality? |

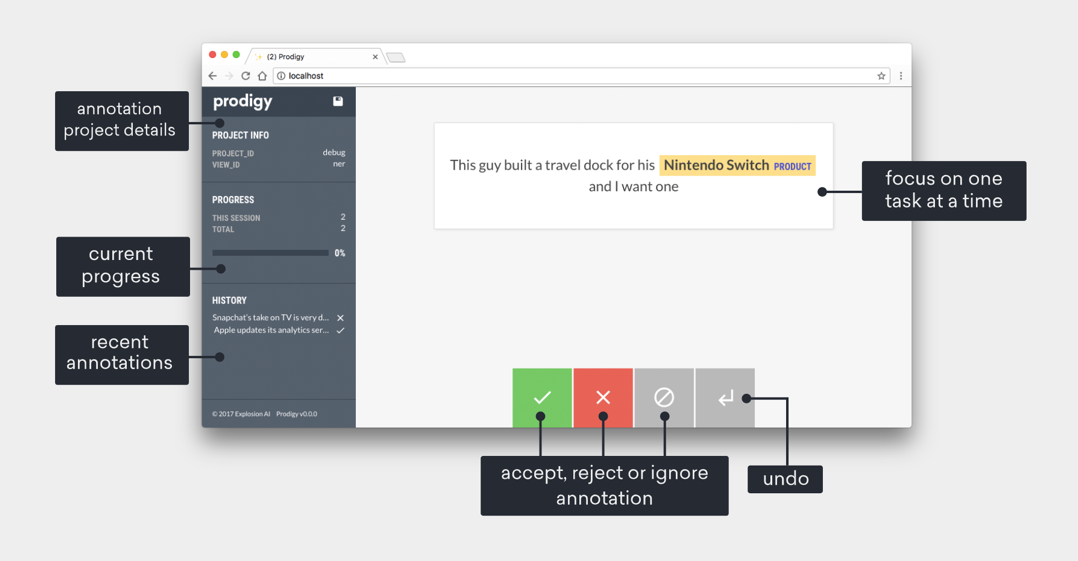
* + - * The first highlighted row represents the labels for the intent of the user.
      * N-columns showcase n-intents.
      * Each row in their corresponding columns represents the sample sentences for the labels mentioned.
    - NER-tagging
      * The process of data preparation for NER tagging is sub divided into several stages:-
        + Raw data
        + Pre-processing raw data
        + Passing into Prodigy (NER-tagging tool)
        + Extracting tags from JSON-L file
        + Passing into model
        + Train and test
      * Raw data sample set:



* + - * Pre-processed data sample set:



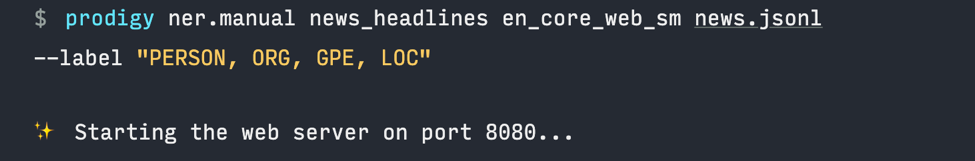
* + - * Prodigy sample of web application:



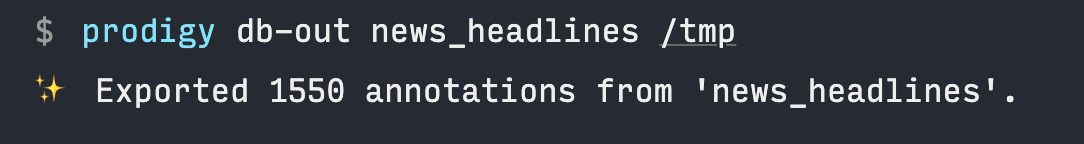
* + - * In order to access prodigy’s web application:
        + One is needed to procure the license to Prodigy
        + Installation of prodigy



* + - * + Running of custom tagging UI



* + - * + Exporting the tags as json-L



* + - * + Tags are ready to be passed into the model.
    - Now all the data is ready in the format needed to be passed in to the script for training and pre-processing.
  + Data pre-processing
    - It involves few basic steps:
      * Removal of stop words
      * Removal of single alphabet words
      * Removal of special characters
      * Converting all into lower case
      * Under some cases numbers are to be kept
      * Lemmatization of the raw data
      * Tokenizing the vocab
    - Before passing the data into model for training the input strings needs to be converted into arrays of numbers from the tokens generated.
    - The output labels in a classification type model needs to be encoded:
      * Hot encoded
      * Integer encoded
    - Now the data is ready for passing into the model as:

X -----🡪 Input array of tokens of words

Y -----🡪 Output array of lables of classes

* + Neural architecture
    - Refer to the Github repo with the jupyter notebook for:
      * Model architecture
        + Uni-directional LSTM
        + Bi-directional LSTM
      * Training
        + Passing the data as X and Y for training and validation.
      * Post-processing
        + It’s the same as pre-processing, the array of strings needs to be passed as array of tokens to the model for prediction.
* Conversion
  + IOS
    - For IOS conversion we use Coremltools.
    - It provides us with a package to directly convert keras models into .mlmodel that can be run in IOS apps.
    - Reason of choosing this over other available solutions such as:
      * TFlite
      * Pytorch
      * For both the cases layer dependencies is an issue that leads to compromising the model architecture and indirectly effecting the accuracy of the results.
  + Android
    - We use TFlite models for android.
    - Issues faced were:
      * Restructuring the neural model based on the compatibility of the layers.
      * Custom layers needs to be created in case of LSTM layers.
      * The converted models were relatively larger in size.
      * Direct conversion from keras to TFlite is difficult because of difference in backend design of the layers needed In neural models.
  + For pre and post processing of the data to be fed to the neural model, it needs to be written in the respective OS languages required by the processors.
  + The approach for the processing remains constant throughout both IOS and android.