To Create a New Version of the Model

**Files:**

* Habitual Be Full and Final Documentation (*This document*)
  + Reference this for more information on the rules, process, data, reasoning, etc
  + This also contains the best results we have been able to generate so far
* Habitual Rule Tagger.ipynb
* Habituality Model Generator.ipynb
* test gold standard line+labels.csv
* test new predicted coraal\_analysis\_spreadsheet.csv

**Instructions:**

* Add in files/file names/data (here and upload to dropbox)
* Note: For our initial data, we only used sentences with one ‘be’. Any data augmented sentences with multiple ‘be’ were not used. The final version of the model generators and taggers have mostly been adjusted to accommodate multiple be’s, however there is one edit that needs to be made. They will need a small change depending on how the gold standard labels are entered for multiple be sentences (e.g. ‘1, -1’ or ‘1 -1’ or ‘1 ; -1’) to properly compare the true labels to their predicted values. For now, they are working on the assumption that the label on each line is only a singular value (e.g. ‘1’ vs ‘1, 1’)

To run the provided code successfully, you'll need to ensure that you have the required libraries and resources installed. Here's a step-by-step guide on what you need:

Python Environment: Make sure you have a Python environment set up. You can download and install Python from the official website: <https://www.python.org/downloads/>

1. Create a new Folder with a specific name:

Download all the above files to this folder

1. Before running the main file: **Habituality Model Generator.ipynb** and **Habituality Rule Tagger.ipynb,** download the required libraries.

Required Libraries: Install the necessary Python libraries using pip. Open your terminal or command prompt and run the following commands:

Command:

**pip install numpy pandas scikit-learn matplotlib seaborn spacy nltk gensim joblib**

1. Download Language Model for spaCy: You are using the spaCy library for natural language processing. You'll need to download the English language model. Run the following command in your terminal:

Command:

**python -m spacy download en\_core\_web\_sm**

1. Download NLTK Resources: NLTK requires additional resources such as stop words. Run the following Python code to download the required resources:

Command:

**import nltk**

**nltk.download('stopwords')**

To test:

1. Make sure you have Habitual Rule Tagger.ipynb, Habituality Model Generator.ipynb, and test gold standard line+labels.csv downloaded to the same folder
2. Run Habituality Model Rule Tagger.ipynb to generate a new spreadsheet that tags each sentences with our habituality rules
3. Run the Habituality Model Generator.ipynb to generate a new model from the rule tagged file that is produced by Habituality Model Tagger.ipynb
4. Compared the results of the new predicted coraal\_analysis\_spreadsheet.csv this generates and the test new predicted coraal\_analysis\_spreadsheet.csv to confirm they match
5. To use different data to create a new model, just sub in ‘test gold standard line+labels.csv’ with a file that has your lines in one column and their true habituality labels in the other; make sure the formatting is the same or you will likely run into errors
   1. As mentioned previously, the code will need an adjustment to properly read in and compared multiple be true habituality/gold standard labels

To Automatically Annotate New Data

**Files:**

* Habitual Be Full and Final Documentation (*This document*)
  + Reference this for more information on the rules, process, data, reasoning, etc
  + This also contains the best results we have been able to generate so far
* Use This to Predict Habituality of New Files.py
  + As the title suggests, use this to run unannotated text files and label them with habitual be
  + Put all files to be annotated in a folder, change the code to correspond to that file location, and run the .py file in terminal
  + There may be issues if the column labels do not match the code, e.g. ‘text’ instead of ‘sentence’; this is alright, just change either the name of the column or the column names in the code
* Merge\_Speaker\_and\_File.py
  + Speaker and text are separated for alignment purposes, but in order to properly tag the sentences, we use this code to merge them
* Habitual Final Code
  + This is the final code that we built the models from
  + Use this to generate new models, just change the name of all the data files
  + Make sure you have a file with the…Habituality labels
* cv.joblib
  + Current version of the countvectorizer model, used for n-grams
* n\_gram.joblib
  + Current version of the n-gram model, fed in as input for habitual model
* habituality\_model.joblib
  + Current version of the habitual model
* new\_texts\_for\_tagging
* speakers

**Instructions:**

To run the provided code successfully, you'll need to ensure that you have the required libraries and resources installed. Here's a step-by-step guide on what you need:

Python Environment: Make sure you have a Python environment set up. You can download and install Python from the official website: <https://www.python.org/downloads/>

1. Create a new Folder with a specific name:

Download all the above files to this folder

Before running the main file: **Use This to Predict Habituality of New Files.py** Download the required libraries.

1. Required Libraries: Install the necessary Python libraries using pip. Open your terminal or command prompt and run the following commands:

Command:

**pip install numpy pandas scikit-learn matplotlib seaborn spacy nltk gensim joblib**

1. Download Language Model for spaCy: You are using the spaCy library for natural language processing. You'll need to download the English language model. Run the following command in your terminal:

Command:

**python -m spacy download en\_core\_web\_sm**

1. Download NLTK Resources: NLTK requires additional resources such as stop words. Run the following Python code to download the required resources:

Command:

**import nltk**

**nltk.download('stopwords')**

1. Make sure to keep the below files in the same folder along with the Main file which are:

i. Use This to Predict Habituality of New Files.ipynb

ii. habituality\_model.joblib

iii. n\_gram.joblib

iv. cv.joblib

v. Merge\_Speaker\_and\_File.py

To test:

1. Download the 'new\_txts\_for\_tagging’ and corresponding ‘speaker’ folders and add it to the newly created folder. Note that after testing, either change your file names to those names or go within the code and replace the file names there with the names of your files. The former should include the text to be tagged and the latter should have the speaker labels. Note, for future texts, this step will only be necessary for where the speaker is not specified in each row, and/or the speaker is not in a separate column from the sentence text.

1. There should be three.txt file in there; “AAHP 020 Juanita Scott Williams 5-14-2010ufdc\_norm.txt”, “AAHP 029 Patricia Burnett 06-02-2010ufdc\_norm.txt” and “AAHP 030 Alvin Butler 06-02-2010ufdc\_norm.txt”

1. Open up a python terminal and type in “cd Downloads” to go to the Downloads folder, or “cd” + the name of the folder you are working from
2. Add the corresponding speakers to each line of the text by typing “python3 Merge\_Speaker\_and\_File.py”; this should produce the ‘new\_txts\_for\_tagging\_speaker’ file, which will be used in the following steps. The ‘new\_txts\_for\_tagging\_speaker’ file is also available in the GitHub for comparison.
3. Run the main file (**Use This to Predict Habituality of New Files\_Final.py**) by typing “python3 Use\_This\_to\_Predict\_Habituality\_of\_New\_Files\_Final.py” within the terminal
4. All the annotated file from the tagger will be saved as Annotated/Actual-filename in the Habitual Annotated folder

1. After running these files, open up the Test folder and compare the results with the txts saved in the Annotated folder. They should all be equivalent, e.g. “Speaker Version AAHP 020 Juanita Scott Williams 5-14-2010ufdc\_norm.txt” in the Annotated file would have the same contents as “Speaker Version AAHP 020 Juanita Scott Williams 5-14-2010ufdc\_norm.txt” in the Test file

**Data:**

The data set for these rules were all the sentences with “be” from the SPOHP oral interviews. Each sentence was annotated by human annotators in order to train the model. Later, additional, data augmented sentences were added to the data set in order to balance it. These sentences were based on habitual sentences in the original data set, and then run through a filter such that they were most likely habitual and thus labeled as such. Note that the test gold standard line+labels.csv contains only sentences with exactly one ‘be’, but Use This to Predict Habituality of New Files.py does have the framework to tag the habituality of each ‘be’ in each sentence for future datasets.

**Rules:**

A rule tagging a sentence means it fits the requirements of that rule, making it true. If it is true, it is marked as 1, if it is false, it is marked as 0.

The rules were parsed using SpaCy’s dependency parser.

1. *POS1*

The word immediately preceding “be” is a modal, an adjective or “to”, most commonly “to” preceding “be”, then modal preceding “be” and finally adjective preceding be

e.g. “Well, no, what you mean what as that like to born to be born in a…”

e.g. “But he didn't prepare properly so that it could be continued.”

1. *POS2*

The word immediately following “be” is an adjective and the word preceding be is not a personal pronoun or noun

e.g. “He told him whatever he did, if he didn't get to go to college, be good at whatever he did.”

1. *POS3*

The word immediately following “be” is a preposition or subordinating conjunction and the word immediately preceding “be” is a singular present verb

e.g. “When you say be mindful of what do you mean?”

1. *POS4*

The word preceding “be” is a noun and the word preceding that noun is an adjective.

e.g. “And I ask the lord that my last day be my best.”

1. *POS5*

The word preceding “be” is an adverb and the word following “be” is either a personal pronoun or determiner.

e.g. “I'm only here to teach, not be a baby-sitter.”

1. *POS6*

The word preceding “be” is an adverb and the word preceding the adverb is a verb or modal.

e.g. “It can't be no worse than it has been!”

1. *A1*

The 'be' is preceded by don't and the word before 'don't' is a noun or pronoun.

eg. She don't be working out.

Or

The 'be' is preceded by a word that is not a verb or auxiliary, which is preceded by a 'don't' and a noun or pronoun.

eg. They don't really be talking

1. *A2*

The 'be' is followed by certain parts of speech; tends to be Non-Habitual. These parts of speech include:

INTJ : Interjection

CCONJ: Conjunction

DET: Determinant

PROPN : Proper noun

PUNCT: Punctuation

eg. It wasn't as big as I thought it would be.

eg. It going to be an interesting time to wonder. ('an' is a Determinant)

1. *A3*

The 'be' is directly preceded by a pronoun or indirectly preceded by a pronoun and the words between the pronoun and 'be' are not auxiliaries , verbs or particles; tends to be Habitual.

eg.I just be liking the beat to a hip hop song.

Here, 'be' is indirectly preceded by a pronoun and 'just' here is an adverb.(which is not AUX or Verb or Particle)

eg.I be listening to the beats

1. *A4*

'Be' is followed by a verb ending in 'ing' and is not preceded by an auxiliary verb, ‘to’, or any of the words in phonetic variation: ‘gonna’, ‘gotta’, ‘wanna’, or ‘tryna’; tends to be Habitual.

eg. Elysa be showing me some work

1. *A5*

'Be' is preceded by a word ending in ‘n't’ which is not ‘don't’; it tends to be Non-Habitual.

eg. I mean, you can but you wouldn't be too successful with it

1. *SynPar1*

The “be” in the sentence has children with an aux dependency relation and an AUX upos tag

e.g. “All the things that children should do for parents we did, I did, and she says it will be a different day and further up on that road, you're gonna see what I am talking about.”

1. *SynPar2*

The “be” in the sentence has siblings with an aux dependency relation and an AUX upos tag

e.g. “Those barriers may not really be holding them back and they could possibly find a way to do something about the issues that exist.”

1. *SynPar3*

The “be” in the sentence has an aux dependency relation and an AUX upos tag and it’s head has a upos tag of VERB

e.g. “Like a person be lying.”

1. *SynPar4*

The “be” in the sentence is labeled has a VERB upos tag

e.g. “We had study hour, and from that, there was no reason for you be out and stuff.”

1. *R1*

If none of the pos rules are true, the ‘be’ tends to be Habitual.

**Rule Interactions:**

Rule interactions help make the rules stronger; i.e. lean more toward habitual or non-habitual. For example, a rule (rule 1) might somewhat indicate habituality, unless another rule (rule 2) is true, in which case the be is non-habitual. By setting rule 1 to 0 if rule 2 is true, rule 1 will be a better predictor of habituality.

e.g. if (Sentence.a4 == 1):

Sentence.a5 == 0

In the above, a4 indicates a habitual be, while a5 indicates a non-habitual be, however if both are true, the sentence is usually habitual. Thus, a5 is marked 0 to reflect this idea. In fact, observing the training data, every sentence still marked with a5 after applying this interaction is non-habitual.

*Full list of the rule interactions*

for Sentence in sen:

if (Sentence.r1 == 0):

Sentence.synPar3 = 0

if (Sentence.a4 == 1):

Sentence.synPar3 = 0

if (Sentence.a2 == 1 and Sentence.a3 == 1):

Sentence.pos5 = 0

if (Sentence.a4 == 1):

Sentence.a5 == 0

if (Sentence.a3 == 1):

Sentence.synPar2 = 0

if (Sentence.pos6 == 1 or Sentence.synPar1 == 1):

Sentence.a3 = 0

if (Sentence.synPar1 == 1):

Sentence.r1 = 0

**N-grams:**

First, the tagger is shown a dataset where it observes the four words before and after each ‘be’ and the habituality tag of each ‘be’. A MultinomialNB model is trained on this data and exported. Using the information it has learned, the model then predicts the habituality of new sentences from the four words before and after the ‘be’ in each sentence. Finally, this initial prediction is used as one of the variables to ultimately determine habituality.

**POS:**

Certain POS (part-of-speech) tags are taken into account by many of the habituality rules. However, in addition to this, there are numerous patterns that may not have been considered by the rules. Thus, the POS tags are also used as separate variables to help the tagger determine whether each ‘be’ is Habitual or Non-Habitual.

**Model:**

The habituality model was created with all the inputs above as the predictor (X) values used to predict the predicted (y) value. To train the model and assess its accuracy, we using a stratified k-fold ensemble model. The stratified k-fold component divides the data into 10 equal, balanced datasets and rotates which 9 are the train and 1 is the test. The ensemble model combines the results of the logistic regression (lr), multilayer perceptron (mlp), and support vector classification (svp) models.

The ensemble model is run on each of the train/test split combinations and the results are averaged to a percentage.

The higher the percentage for each sentence, the more confident the model that the ‘be’ is non-habitual. Based on this, the ‘be’ is classified as -1 or 1, -1 being non-habitual and 1 being habitual. The default threshold value is 0.5, that is, above the threshold means non-habitual, and below means habitual.

However, we adjusted the threshold to 0.84 to increase the recall of the Habitual class. The reasoning here is that with very high habitual recall, annotators only have to verify that the sentences tagged habitual are in fact habitual; the sentences tagged non-habitual will almost always be tagged correctly and thus can be accepted as is. In addition, there are many fewer habitual ‘be’ in most transcripts as compared to non-habitual ‘be’, so this even further reduces the amount of manual verification that needs to be done after automatic tagging

Regarding the data, at first, there were many fewer habitual sentences than non-habitual, making it more difficult for the model to accurately tag sentences as habitual. Thus, we ran the habitual sentences through data augmentation code written by Harrison Santiago. This resulted in us having a roughly equal number of habitual and non-habitual sentences for the model to be trained on.

In the case of multiple be’s, the model is run on each individual ‘be’, then the results of each ‘be’ are combined together, e.g. “1, -1, 1”, indicating the first and last be are habitual but the middle be is not.

*Key/Definitions*

* 1: habitual be, -1: non-habitual be, 0: no be
  + If multiple be’s in a sentence, tags each be, e.g. “-1, 1”
* Stratified k-fold – rotates train and test data while preserving the percentage of samples in each class (e.g. 40% non-habitual, 40% habitual), then averages results
* Ensemble – mix of results from lr, mlp, svc
* Pipeline – standardizes data
* Threshold – higher threshold = more likely to class ‘be’ as habitual, thus increasing the recall for the Habitual class

**Results**

Below is the classification report of the Habituality model. Initially, the threshold for classing ‘be’ as non-habitual was 0.94, that is, the tagger had to have 94% certainty or higher that the ‘be’ was non-habitual in order to class it as such. As a result, the recall for the Habitual class was very high (around 0.99), which seemed ideal for annotating. However, in practice, the 0.94 threshold resulted in excessive amounts of non-habitual sentences predicted as habitual without a clear benefit in regards to accurately predicting habitual sentences as habitual. Thus, we chose a lower threshold of 0.84 for better results.

A screenshot of a computer

Description automatically generated

As you can see above, the accuracy, recall, and precision are all quite high, particularly the recall for habitual sentences and precision for nonhabitual sentences.

We output the components of this model as cv.joblib, ngram.joblib, and habituality\_model.joblib. Then, we wrote a second script to analyze each sentence to generate the inputs like rule truth values, pos value, etc. Finally, we ran the model on each file and adjusted for the threshold of 0.84.