To Automatically Annotate New Data

**Files:**

* Multiple Negation 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
* Multiple Negator Training Model Final.ipynb
  + Use this to run unannotated text files and label them with multiple negation
  + 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
* MN\_48\_152\_2 19 2024 .txt

**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: **Multiple Negator Training Model Final.ipynb** 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:
2. MN\_48\_152\_2 19 2024 .txt

To test:

1. Download the ' MN\_48\_152\_2 19 2024 .txt’.
2. Open and run the main file (**Multiple Negator Training Model Final.ipynb**) in Jupyter Notebook.
3. The annotated file will be save as multNeg.txt.
4. After running this file, compare the results of multNeg.txt with multNegResults.txt; these should be the same.

**Data:**

Each sentence was annotated by human annotators for use in developing the tagger. MN\_48\_152\_2 19 2024 .txt contains a combination of sentences with and without multiple negation, though predominantly without.

**Rules:**

A rule tagging a sentence means it fits the requirements of that rule, making it true. If any rules are true, the sentence is marked with 1 for the presence of multiple negation, otherwise it is marked with 0.

*Rule 1*

while ((i+num < len(parsed)) and (num < 9) and (parsed[i+num].pos\_ != "PUNCT" and parsed[i+num].pos\_ != "CCONJ" and parsed[i+num].pos\_ != "SCONJ" and parsed[i+num].text != "–")):

if next.text.lower() in negatives and (word.text.lower() != "no" or next.text.lower() != "no"):

*Rule 2*

while ((i-num >= 0) and (num < 9) and (parsed[i-num].pos\_ != "PUNCT") and parsed[i-num].pos\_ != "CCONJ" and parsed[i-num].pos\_ != "SCONJ" and parsed[i-num].text != "–"):

if prev.text.lower() in negatives and (word.text.lower() != "no" or prev.text.lower() != "no"):

For both of the above rules, the tagger would first identify an instance of negation. Then, it would search forward and backward, making sure it stays in the same clause (not interrupted by punctuations or conjunctions) to find another instance of negation. If it finds one, the rule will be set to ‘1’ for true. The exception to two negatives in the same clause being considered multiple negation is the case of two “no”. Each loop ends when it finds the end of a clause or sentence.

*Key/Definitions*

* 1: multiple negation, 0: not multiple negation
  + If there are multiple potential cases of multiple negation in a sentence, tags as “1” if any are found to be multiple negation, “0” if none are

**Results**

Below is the classification report of the Multiple Negation tagger.

A number of numbers in a row

AI-generated content may be incorrect.

As you can see above, the accuracy, recall, and precision are all very high if not 1.00. Observing the falsely predicted values, there is only one instance in this dataset, that being one sentence without multiple negation being tagged as multiple negation.