To Automatically Annotate New Data

**Files:**

* Person Number Disagreement 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
* Person Number Agreement Wil. Final.ibynb
  + Use this to run unannotated text files and label them with person number disagreement
  + 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
* person\_num\_revised\_test.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: **Person Number Agreement Wil. Final.ibynb** 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. person\_num\_revised\_test.txt

To test:

1. Download person\_num\_revised\_test.txt.
2. Open and run the main file (**M Person Number Agreement Wi. Final.ibynb**) in Jupyter Notebook.
3. The annotated file will be saved as personNum.txt.
4. After running this file, compare the results of personNum.txt with personNumResults.txt; these should be the same. The classification report should match the report in the results section as well.

**Data:**

Each sentence was annotated by human annotators for use in developing the tagger. person\_num\_revised\_test.txt contains a combination of sentences with and without person number disagreement, though predominantly without. This dataset was removed and isolated from the larger dataset prior to analysis or training, such that none of the rules were influenced by this data and the results reflect the true performance of the tagger.

**Rules:**

A rule tagging a sentence means it fits the requirements of that rule, making it true. If it is true, pna is set to 1. If none of the rules apply, pna is set to 0, indicating there is no person number disagreement

The rules were parsed using SpaCy’s dependency parser.

First, the tagger identifies a potential subject, a potential verb, and the dependencies they have between them. If those dependencies meet a certain criteria, the sentence could potentially have person number disagreement. The subject in question is changed in one particular circumstance, where the word before the potential subject is a better indicator of subject tense than the original word. In that instance, the previous word is designated as the subject.

if word.tag\_ in ['DT', 'WDT', 'WP'] and word.dep\_ in ['nsubj', 'nsubjpass'] and prevword.pos\_ in ['NOUN', 'PRON', 'PROPN', 'NUM']:

word = prevword

Then, the rules are applied to the sentence determine whether they do or do not have person number disagreement. With the exception of rules 3 and 8, if any of these rules are true, the sentence is tagged as person number disagreement. Rules 3 and 8 may tag a sentence as agreement or disagreement depending on the circumstances.

1) subject is not plural and is a third party singular pronoun followed by non-third party singular verb

if (((word.text.lower() in thirdparty\_singular\_pronouns) and nextword.tag\_ =='VBP')):

Sentence1.p = 1

2) subject is singular and followed by a base or non-third party verb which is not 'was' and does not have a 'relcl' (relative clause) dependency

if ((word.tag\_ == 'NN' or word.tag\_ == 'NNP') and (word.text.lower() not in pluralquantifiers) and (nextword.tag\_ == 'VB' or nextword.tag\_ == 'VBP') and nextword.text.lower() != 'was' and nextword.dep\_ != 'relcl'):

Sentence1.p = 1

3) if either of rules 1-2 are true (e.g. singular noun followed by plural verb), but the pronoun/verb combination preceded by a 'question' word, e.g. 'did', then there is no disagreement. Otherwise, if those rules are not true but the noun is singular (e.g. the verb is singular as the plural condition is not met) and preceded by a question verb, there is disagreement

changed = 0

if (nextword.text not in ['was', 'were'] and nextword.tag\_ not in ['VBG', 'VBD'] and ((prevword and prevword.text.lower() in questionwords and Sentence1.text[tok\_l[i+count-2]['start']:tok\_l[i+count-2]['end']] == prevword.text and Sentence1.text[tok\_l[tok\_l[i+count-2]['head']]['start']:tok\_l[tok\_l[i+count-2]['head']]['end']] == nextword.text) or (prevprevword and prevprevword.text.lower() in questionwords and Sentence1.text[tok\_l[i+count-3]['start']:tok\_l[i+count-3]['end']] == prevprevword.text and Sentence1.text[tok\_l[tok\_l[i+count-3]['head']]['start']:tok\_l[tok\_l[i+count-3]['head']]['end']] == nextword.text) or (prevprevprevword and prevprevprevword.text.lower() in questionwords and Sentence1.text[tok\_l[i+count-4]['start']:tok\_l[i+count-4]['end']] == prevprevprevword.text and Sentence1.text[tok\_l[tok\_l[i+count-4]['head']]['start']:tok\_l[tok\_l[i+count-4]['head']]['end']] == nextword.text))):

if Sentence1.p == 1:

Sentence1.p = 0

changed = 1

elif ((changed == 0) and (Sentence1.p == 0) and (word.text.lower() in thirdparty\_singular\_pronouns) or ((word.tag\_ == 'NN') and (word.text.lower() not in pluralquantifiers))):

Sentence.p = 1

4) singular pronoun subject 'I' or 'you' followed by plural verb

if ((word.text.lower() in other\_singular\_pronouns) and nextword.text.endswith('s') and nextword.text.lower() != 'was') and not nextword.text.endswith('ss'):

Sentence1.p = 1

5) subject is plural followed by 3rd person singular verb

if (((word.tag\_ == 'NNS' or word.tag\_ == 'NNPS' or (word in other\_singular\_pronouns))) and nextword.tag\_ =='VBZ'):

Sentence1.p = 1

6) subject is plural or in second person followed by 'was'

if ((((word.text.lower() in thirdparty\_plural\_pronouns) or (word.text.lower() in personal\_object\_pronouns\_plural)) or (word.pos\_ == 'NUM' and (word.text != "1" and word.text.lower() != "one")) or (word.text.lower() in pluralquantifiers) or((word.tag\_ == 'NNS' or word.tag\_ == 'NNPS'))) and (nextword.text.lower() == "was") and (parsed[i+1].tag\_ != 'PRON' and parsed[i+1].tag\_ != 'PRP')):

Sentence1.p = 1

7) compound subject connected through 'and' where the verb is ‘was’ and/or is not plural

if (parsed[i+1].text == "and" or word.dep\_ == "conj") and (nextword.tag\_ =='VBZ' or nextword.text == 'was'):

Sentence1.p = 1

8) usage of there, e.g. determines whether 'were' or 'was' is appropriate given the context and checks whether that matches the subject tense

if word.text.lower() == 'there' and (nextword.text.lower() == 'was' or nextword.text.lower() == 'were'):

nextnext = 1

while len(parsed) >= i+count+nextnext and (parsed[i+count+nextnext].pos\_ == 'ADJ' or parsed[i+count+nextnext].pos\_ == 'DET' or parsed[i+count+nextnext].pos\_ == 'ADV' or parsed[i+count+nextnext].dep\_ == 'compound'):

nextnext += 1

if (parsed[i+count+nextnext].pos\_ == 'NUM' and parsed[i+count+nextnext].text != "1" and parsed[i+count+nextnext].text.lower() != "one") or (parsed[i+count+nextnext].text.lower() in pluralquantifiers) or ((parsed[i+count+nextnext].tag\_ == 'NNS' or parsed[i+count+nextnext].tag\_ == 'NNPS') and parsed[i+count+nextnext].pos\_ != 'PROPN'):

if (nextword.text.lower() == "was"):

Sentence1.p = 1

else:

Sentence1.p = 0

elif (nextword.text.lower() == "were"):

Sentence1.p= 1

else:

Sentence1.p = 0

*Key/Definitions*

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

**Results**

Below is the classification report of the Person Number Disagreement tagger.

A screenshot of a computer

AI-generated content may be incorrect.

As you can see above, the accuracy and recall are quite high (above 0.85). Disagreement precision is low, unfortunately, however this may be in part due to the large disparity in the number of agreement and disagreement sentences. Additionally, the goal is to maximize disagreement recall, and unfortunately this comes at some cost of disagreement precision. Observing the falsely predicted values, there are a mix of person number agreement sentences falsely predicted as disagreement and vice versa.