You may find that, after a lot of chunking, you have some words in your chunk you still do not want, but you have no idea how to get rid of them by chunking. You may find that chinking is your solution.

Chinking is a lot like chunking, it is basically a way for you to remove a chunk from a chunk. The chunk that you remove from your chunk is your chink.

The code is very similar, you just denote the chink, after the chunk, with }{ instead of the chunk's {}.

import nltk

from nltk.corpus import state\_union

from nltk.tokenize import PunktSentenceTokenizer

train\_text = state\_union.raw("2005-GWBush.txt")

sample\_text = state\_union.raw("2006-GWBush.txt")

custom\_sent\_tokenizer = PunktSentenceTokenizer(train\_text)

tokenized = custom\_sent\_tokenizer.tokenize(sample\_text)

def process\_content():

try:

for i in tokenized[5:]:

words = nltk.word\_tokenize(i)

tagged = nltk.pos\_tag(words)

chunkGram = r"""Chunk: {<.\*>+}

}<VB.?|IN|DT|TO>+{"""

chunkParser = nltk.RegexpParser(chunkGram)

chunked = chunkParser.parse(tagged)

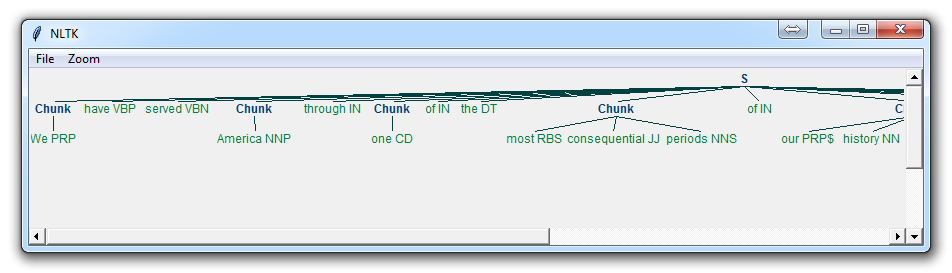
chunked.draw()

except Exception as e:

print(str(e))

process\_content()

With this, you are given something like:



Now, the main difference here is:

}<VB.?|IN|DT|TO>+{

This means we're removing from the chink one or more verbs, prepositions, determiners, or the word 'to'.