Machine Learning Plus (https://www.machinelearningplus.com/)

Lemmatization Approaches with Examples in Python

by Selva Prabhakaran (https://www.machinelearningplus.com/author/selva86/) | Posted on October 2, 2018 (https://www.machinelearningplus.com/nlp/lemmatization-examples-python/)

Lemmatization is the process of converting a word to its base form. The difference between stemming and lemmatization is, lemmatization considers the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters, often leading to incorrect meanings and spelling errors.

Comparing Lemmatization Approaches in Python. Photo by Jasmin Schreiber (https://unsplash.com/photos/Fpi3B9RMe5E?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

[container] [columnize]

Contents

[/columnize] [/container]

1. Introduction

Search … Search

Recent Posts

data.table in R – The Complete Beginners Guide (https://www.machinelearningplus.com/data-manipulation/datatable-in-r-complete-guide/)

Augmented Dickey Fuller Test (ADF Test) – Must Read Guide (https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/)

KPSS Test for Stationarity (https://www.machinelearningplus.com/time-series/kpss-test-for-stationarity/)

101 R data.table Exercises (https://www.machinelearningplus.com/data-manipulation/101-r-data-table-exercises/)

P-Value – Understanding from Scratch (https://www.machinelearningplus.com/statistics/p-value/)

101 Python datatable Exercises (pydatatable) (https://www.machinelearningplus.com/data-manipulation/101-python-datatable-exercises-pydatatable/)

Vector Autoregression (VAR) – Comprehensive Guide with Examples in Python (https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/)

Mahalanobis Distance – Understanding the math with examples (python) (https://www.machinelearningplus.com/statistics/mah distance/)

datetime in Python – Simplified Guide with Clear Examples (https://www.machinelearningplus.com/python/datet python-examples/)

Principal Component Analysis (PCA) – Better Explained (https://www.machinelearningplus.com/machine-learning/principal-components-analysis-pca-better-explained/)

Python Logging – Simplest Guide with Full Code and Examples (https://www.machinelearningplus.com/python/pytho logging-guide/)

Feedback

Lemmatization is the process of converting a word to its base form. The difference between stemming and lemmatization is, lemmatization considers the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters, often leading to incorrect meanings and spelling errors.

For example, lemmatization would correctly identify the base form of 'caring' to 'care', whereas, stemming would cutoff the 'ing' part and convert it to car.

'Caring' -> Lemmatization -> 'Care'
'Caring' -> Stemming -> 'Car'

Also, sometimes, the same word can have multiple different 'lemma's. So, based on the context it's used, you should identify the 'part-of-speech' (POS) tag for the word in that specific context and extract the appropriate lemma. Examples of implementing this comes in the following sections.

Today, we will see how to implement lemmatization using the following python packages.

1. Wordnet Lemmatizer
2. Spacy Lemmatizer
3. TextBlob
4. CLiPS Pattern
5. Stanford CoreNLP
6. Gensim Lemmatizer
7. TreeTagger

2. Wordnet Lemmatizer with NLTK

Wordnet (https://wordnet.princeton.edu/) is an large, freely and publicly available lexical database for the English language aiming to establish structured semantic relationships between words. It offers lemmatization capabilities as well and is one of the earliest and most commonly used lemmatizers.

NLTK offers an interface to it, but you have to download it first in order to use it. Follow the below instructions to install `nltk` and download `wordnet`.

```
 # How to install and import NLTK
# In terminal or prompt:
# pip install nltk

# # Download Wordnet through NLTK in python console:
import nltk
nltk.download('wordnet')
```

In order to lemmatize, you need to create an instance of the `WordNetLemmatizer()` and call the `lemmatize()` function on a single word.

Matplotlib Histogram – How to Visualize Distributions in Python (https://www.machinelearningplus.com/plots/matplot histogram-python-examples/)

ARIMA Model – Complete Guide to Time Series Forecasting in Python (https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/)

Time Series Analysis in Python – A Comprehensive Guide with Examples (https://www.machinelearningplus.com/time-series/time-series-analysis-python/)

Matplotlib Tutorial – A Complete Guide to Python Plot w/ Examples (https://www.machinelearningplus.com/plots/matplot tutorial-complete-guide-python-plot-examples/)

Topic modeling visualization – How to present the results of LDA models? (https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/)

Top 50 matplotlib Visualizations – The Master Plots (with full python code) (https://www.machinelearningplus.com/plots/top-50-matplotlib-visualizations-the-master-plots-python/)

List Comprehensions in Python – My Simplified Guide (https://www.machinelearningplus.com/python/list-comprehensions-in-python/)

Python @Property Explained – How to Use and When? (Full Examples) (https://www.machinelearningplus.com/python/pytho property/)

How Naive Bayes Algorithm Works? (with example and full code) (https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/)

Tags

Bigrams (https://www.machinelearningplus.com/tag/bigrams/)
Classification (https://www.machinelearningplus.com/tag/cla
Corpus (https://www.machinelearningplus.com/tag/corpus/)
Cosine Similarity (https://www.machinelearningplus.com/tag/cosine-similarity/)
data.table (https://www.machinelearningplus.com/tag/data-table/) Data Manipulation (https://www.machinelearningplus.com/tag manipulation/) Debugging (https://www.machinelearningplus.com/tag/debugging/) Doc2Vec (https://www.machinelearningplus.com/tag/doc2vec/)

Feedback

```
import nltk
from nltk.stem import WordNetLemmatizer

# Init the Wordnet Lemmatizer
lemmatizer = WordNetLemmatizer()

# Lemmatize Single Word
print(lemmatizer.lemmatize("bats"))
#> bat

print(lemmatizer.lemmatize("are"))
#> are

print(lemmatizer.lemmatize("feet"))
#> foot
```

Let's lemmatize a simple sentence. We first tokenize the sentence into words using `nltk.word_tokenize` and then we will call `lemmatizer.lemmatize()` on each word. This can be done in a list comprehension (the for-loop inside square brackets to make a list).

```
# Define the sentence to be lemmatized
sentence = "The striped bats are hanging on their feet for best"

# Tokenize: Split the sentence into words
word_list = nltk.word_tokenize(sentence)
print(word_list)
#> ['The', 'striped', 'bats', 'are', 'hanging', 'on', 'their', 'feet', 'for', 'best']

# Lemmatize list of words and join
lemmatized_output = ' '.join([lemmatizer.lemmatize(w) for w in word_list])
print(lemmatized_output)
#> The striped bat are hanging on their foot for best
```

The above code is a simple example of how to use the wordnet lemmatizer on words and sentences.

Notice it didn't do a good job. Because, 'are' is not converted to 'be' and 'hanging' is not converted to 'hang' as expected. This can be corrected if we provide the correct 'part-of-speech' tag (https://www.clips.uantwerpen.be/pages/MBSP-tags) (POS tag) as the second argument to `lemmatize()`.

Sometimes, the same word can have a multiple lemmas based on the meaning / context.

```
print(lemmatizer.lemmatize("stripes", 'v'))
#> strip

print(lemmatizer.lemmatize("stripes", 'n'))
#> stripe
```

It may not be possible manually provide the corrent POS tag for every word for

Evaluation Metrics (https://www.machinelearningplus.com/tag/evaluation-metrics/) FastText (https://www.machinelearningplus.com/tag/fasttext/) Feature Selection (https://www.machinelearningplus.com/tag/feature-selection/) Gensim (https://www.machinelearningplus.com/tag/gensim LDA (https://www.machinelearningplus.com/tag/lda/) Lemmatization (https://www.machinelearningplus.com/tag/lemmatization/) Linear Regression (https://www.machinelearningplus.com/tag/linear-regression/) Logistic (https://www.machinelearningplus.com/tag/logistic/) LSI (https://www.machinelearningplus.com/tag/lsi/) Matplotlib (https://www.machinelearningplus.com/tag/matplo Multiprocessing (https://www.machinelearningplus.com/tag/multiprocessing/) NLP (https://www.machinelearningplus.com/tag/nl NLTK (https://www.machinelearningplus.com/tag/nltk/) Numpy (https://www.machinelearningplus.com/tag/numpy P-Value (https://www.machinelearningplus.com/tag/p-value/) Pandas (https://www.machinelearningplus.com/tag/pandas/) Parallel Processing (https://www.machinelearningplus.com/tag/parallel-processing/) Phraser (https://www.machinelearningplus.com/tag/phraser/) Practice Exercise (https://www.machinelearningplus.com/tag/practice-exercise/) Python (https://www.machinelearni R (https://www.machinelearningplus.co Regex (https://www.machinelearningplus.com/tag/regex/) Regression (https://www.machinelearningplus.com/tag/regression/) Residual Analysis (https://www.machinelearningplus.com/tag/residual-analysis/) Scikit Learn (https://www.machinelearningplus.com/tag/scikit-learn/) Significance Tests (https://www.machinelearningplus.com/tag/signific tests/) Soft Cosine Similarity (https://www.machinelearningplus.com/tag/soft-cosine-similarity/) spaCy (https://www.machinelearningplus.com/tag/spacy/) Stationarity (https://www.machinelearningplus.com/tag/station Summarization (https://www.machinelearningplus.com/tag/summarization/) TaggedDocument (https://www.machinelearningplus.com/tag/taggeddocument/) TextBlob (https://www.machinelearningplus.com/tag/textblob/) TFIDF (https://www.machinelearningplus.com/tag/tfidf/) Time Series (https://www.machinelearningplus.com/tag

Feedback

3. Wordnet Lemmatizer with appropriate POS tag

large texts. So, instead, we will find out the correct POS tag for each word, map it to the right input character that the WordnetLemmatizer accepts and pass it as the second argument to `lemmatize()`.

So how to get the POS tag for a given word?

In nltk, it is available through the `nltk.pos_tag()` method. It accepts only a list (list of words), even if its a single word.

```
 print(nltk.pos_tag(['feet']))
#> [('feet', 'NNS')]

print(nltk.pos_tag(nltk.word_tokenize(sentence)))
#> [('The', 'DT'), ('striped', 'JJ'), ('bats', 'NNS'), ('are', 'VBP'), ('hanging', 'VBG'), ('on
```

`nltk.pos_tag()` returns a tuple with the POS tag. The key here is to map NLTK's POS tags to the format wordnet lemmatizer would accept. The `get_wordnet_pos()` function defined below does this mapping job.

```
 # Lemmatize with POS Tag
from nltk.corpus import wordnet

def get_wordnet_pos(word):
    """Map POS tag to first character lemmatize() accepts"""
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}

    return tag_dict.get(tag, wordnet.NOUN)


# 1. Init Lemmatizer
lemmatizer = WordNetLemmatizer()

# 2. Lemmatize Single Word with the appropriate POS tag
word = 'feet'
print(lemmatizer.lemmatize(word, get_wordnet_pos(word)))

# 3. Lemmatize a Sentence with the appropriate POS tag
sentence = "The striped bats are hanging on their feet for best"
print([lemmatizer.lemmatize(w, get_wordnet_pos(w)) for w in nltk.word_tokenize(sentence)])
#> ['The', 'strip', 'bat', 'be', 'hang', 'on', 'their', 'foot', 'for', 'best']
```

4. spaCy Lemmatization

series/) Topic Modeling (https://www.machinelearningplus.com/tag/topic-modeling/) Visualization (https://www.machinelearningplus.com/tag/visualiz
Word2Vec (https://www.machinelearningplus.com/tag/word2vec/)

Feedback

spaCy is a relatively new in the space and is billed as an industrial strength NLP engine. It comes with pre-built models (https://spacy.io/usage/models) that can parse text and compute various NLP related features through one single function call. Ofcourse, it provides the lemma of the word too.

Before we begin, let's install spaCy and download the 'en' model.

```
 # Install spaCy (run in terminal/prompt)
import sys
!{sys.executable} -m pip install spacy

# Download spaCy's  'en' Model
!{sys.executable} -m spacy download en
```

spaCy determines the part-of-speech tag by default and assigns the corresponding lemma. It comes with a bunch of prebuilt models where the 'en' we just downloaded above is one of the standard ones for english.

```
 import spacy

# Initialize spacy 'en' model, keeping only tagger component needed for lemmatization
nlp = spacy.load('en', disable=['parser', 'ner'])

sentence = "The striped bats are hanging on their feet for best"

# Parse the sentence using the loaded 'en' model object `nlp`
doc = nlp(sentence)

# Extract the lemma for each token and join
" ".join([token.lemma_ for token in doc])
#> 'the strip bat be hang on -PRON- foot for good'
```

It did all the lemmatizations the Wordnet Lemmatizer supplied with the correct POS tag did. Plus it also lemmatized 'best' to 'good'. Nice!

You'd see the -PRON- character coming up whenever spacy detects a pronoun.

5. TextBlob Lemmatizer

TexxtBlob is a powerful, fast and convenient NLP package as well. Using the `Word` and `TextBlob` objects, its quite straighforward to parse and lemmatize words and sentences respectively.

```
 # pip install textblob
from textblob import TextBlob, Word

# Lemmatize a word
word = 'stripes'
w = Word(word)
w.lemmatize()
#> stripe
```

However to lemmatize a sentence or paragraph, we parse it using TextBlob and call the lemmatize() function on the parsed words.

```
 # Lemmatize a sentence
sentence = "The striped bats are hanging on their feet for best"
sent = TextBlob(sentence)
" ". join([w.lemmatize() for w in sent.words])
#> 'The striped bat are hanging on their foot for best'
```

It did not do a great job at the outset, because, like NLTK, TextBlob also uses wordnet internally. So, let's pass the appropriate POS tag to the lemmatize() method.

6. TextBlob Lemmatizer with appropriate POS tag

```
 # Define function to lemmatize each word with its POS tag
def lemmatize_with_postag(sentence):
    sent = TextBlob(sentence)
    tag_dict = {"J": 'a',
                "N": 'n',
                "V": 'v',
                "R": 'r'}
    words_and_tags = [(w, tag_dict.get(pos[0], 'n')) for w, pos in sent.tags]
    lemmatized_list = [wd.lemmatize(tag) for wd, tag in words_and_tags]
    return " ".join(lemmatized_list)

# Lemmatize
sentence = "The striped bats are hanging on their feet for best"
lemmatize_with_postag(sentence)
#> 'The striped bat be hang on their foot for best'
```

7. Pattern Lemmatizer

Pattern by CLiPs (https://www.clips.uantwerpen.be/pages/pattern) is a versatile module with many useful NLP capabilities.

```
 !pip install pattern
```

If you run into issues while installing pattern, check out the known issues on github (https://github.com/clips/pattern/issues). I myself faced this issue (https://github.com/clips/pattern/issues/203) when installing on a mac.

```
 import pattern
from pattern.en import lemma, lexeme

sentence = "The striped bats were hanging on their feet and ate best fishes"
" ".join([lemma(wd) for wd in sentence.split()])
#> 'the stripe bat be hang on their feet and eat best fishes'
```

You can also view the possible lexeme's for each word.

```
 # Lexeme's for each word
[lexeme(wd) for wd in sentence.split()]

#> [['the', 'thes', 'thing', 'thed'],
#>  ['stripe', 'stripes', 'striping', 'striped'],
#>  ['bat', 'bats', 'batting', 'batted'],
#>  ['be', 'am', 'are', 'is', 'being', 'was', 'were', 'been',
#> . 'am not', "aren't", "isn't", "wasn't", "weren't"],
#>  ['hang', 'hangs', 'hanging', 'hung'],
#>  ['on', 'ons', 'oning', 'oned'],
#>  ['their', 'theirs', 'theiring', 'theired'],
#>  ['feet', 'feets', 'feeting', 'feeted'],
#>  ['and', 'ands', 'anding', 'anded'],
#>  ['eat', 'eats', 'eating', 'ate', 'eaten'],
#>  ['best', 'bests', 'besting', 'bested'],
#>  ['fishes', 'fishing', 'fishesed']]
```

You could also obtain the lemma by parsing the text.

```
 from pattern.en import parse
print(parse('The striped bats were hanging on their feet and ate best fishes',
            lemmata=True, tags=False, chunks=False))
#> The/DT/the striped/JJ/striped bats/NNS/bat were/VBD/be hanging/VBG/hang on/IN/on their/PRP$/
#>  feet/NNS/foot and/CC/and ate/VBD/eat best/JJ/best fishes/NNS/fish
```

8. Stanford CoreNLP Lemmatization

Standford CoreNLP (https://stanfordnlp.github.io/CoreNLP/index.html) is a popular NLP tool that is originally implemented in Java. There are many python wrappers written around it. The one I use below is one that is quite convenient to use.

But before that, you need to download Java and the Standford CoreNLP software. Make sure you have the following requirements before getting to the lemmatization code:

Step 1: Java 8 Installed

You can download and install from Java download page (https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html).

Mac users can check the java version by typing `java -version` in terminal. If its 1.8+, then its Ok. Else follow below steps.

```
 brew update
brew install jenv
brew cask install java
```

Step 2: Download Standford CoreNLP software (https://stanfordnlp.github.io/CoreNLP/index.html#download) and unzip it.

Step 3: Start the Stanford CoreNLP server from terminal. How? `cd` to the folder you just unzipped and run below command in terminal:

```
 cd stanford-corenlp-full-2018-02-27
java -mx4g -cp "*" edu.stanford.nlp.pipeline.StanfordCoreNLPServer -annotators "tokenize,ssplit
```

This will start a StanfordCoreNLPServer listening at port 9000. Now, we are ready to extract the lemmas in python.

In the `stanfordcorenlp` package, the lemma is embedded in the output of the `annotate()` method of the StanfordCoreNLP connection object (see code below).

```
 # Run `pip install stanfordcorenlp` to install stanfordcorenlp package
from stanfordcorenlp import StanfordCoreNLP
import json

# Connect to the CoreNLP server we just started
nlp = StanfordCoreNLP('http://localhost', port=9000, timeout=30000)

# Define proporties needed to get lemma
props = {'annotators': 'pos,lemma',
         'pipelineLanguage': 'en',
         'outputFormat': 'json'}


sentence = "The striped bats were hanging on their feet and ate best fishes"
parsed_str = nlp.annotate(sentence, properties=props)
parsed_dict = json.loads(parsed_str)
parsed_dict
#> {'sentences': [{'index': 0,
#>    'tokens': [{'after': ' ',
#>      'before': '',
#>      'characterOffsetBegin': 0,
#>      'characterOffsetEnd': 3,
#>      'index': 1,
#>      'lemma': 'the',      << ----------- LEMMA
#>      'originalText': 'The',
#>      'pos': 'DT',
#>      'word': 'The'},
#>     {'after': ' ',
#>      'before': ' ',
#>      'characterOffsetBegin': 4,
#>      'characterOffsetEnd': 11,
#>      'index': 2,
#>      'lemma': 'striped',  << ----------- LEMMA
#>      'originalText': 'striped',
#>      'pos': 'JJ',
#>      'word': 'striped'},
#>     {'after': ' ',
#>      'before': ' ',
#>      'characterOffsetBegin': 12,
#>      'characterOffsetEnd': 16,
#>      'index': 3,
#>      'lemma': 'bat',      << ----------- LEMMA
#>      'originalText': 'bats',
#>      'pos': 'NNS',
#>      'word': 'bats'}
#> ...
#> ...
```

The output of nlp.annotate() was converted to a dict using json.loads. Now the lemma we need is embedded a couple of layers inside the parsed_dict. So here, we need to just the lemma value from each dict. I use list comprehensions below to do the trick.

```
 lemma_list = [v for d in parsed_dict['sentences'][0]['tokens'] for k,v in d.items() if k == 'l
" ".join(lemma_list)
#> 'the striped bat be hang on they foot and eat best fish'
```

Let's generalize this a nice function so as to handle larger paragraphs.

```
 from stanfordcorenlp import StanfordCoreNLP
import json, string

def lemmatize_corenlp(conn_nlp, sentence):
    props = {
        'annotators': 'pos,lemma',
        'pipelineLanguage': 'en',
        'outputFormat': 'json'
    }

    # tokenize into words
    sents = conn_nlp.word_tokenize(sentence)

    # remove punctuations from tokenised list
    sents_no_punct = [s for s in sents if s not in string.punctuation]

    # form sentence
    sentence2 = " ".join(sents_no_punct)

    # annotate to get lemma
    parsed_str = conn_nlp.annotate(sentence2, properties=props)
    parsed_dict = json.loads(parsed_str)

    # extract the lemma for each word
    lemma_list = [v for d in parsed_dict['sentences'][0]['tokens'] for k,v in d.items() if k ==

    # form sentence and return it
    return " ".join(lemma_list)


# make the connection and call `lemmatize_corenlp`
nlp = StanfordCoreNLP('http://localhost', port=9000, timeout=30000)
lemmatize_corenlp(conn_nlp=nlp, sentence=sentence)
#> 'the striped bat be hang on they foot and eat best fish'
```

9. Gensim Lemmatize

Gensim provide lemmatization facilities based on the `pattern` package. It can be implemented using the `lemmatize()` method in the `utils` module. By default `lemmatize()` allows only the 'JJ', 'VB', 'NN' and 'RB' tags.

```
from gensim.utils import lemmatize
sentence = "The striped bats were hanging on their feet and ate best fishes"
lemmatized_out = [wd.decode('utf-8').split('/')[0] for wd in lemmatize(sentence)]
#> ['striped', 'bat', 'be', 'hang', 'foot', 'eat', 'best', 'fish']
```

10. TreeTagger

Treetagger is a Part-of-Speech tagger for many languages. And it provides the lemma of the word as well.

You will need to download and install the TreeTagger software (http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/) itself in order to use it by following steps mentioned.

```
 # pip install treetaggerwrapper

import treetaggerwrapper as ttpw
tagger = ttpw.TreeTagger(TAGLANG='en', TAGDIR='/Users/ecom-selva.p/Documents/MLPlus/11_Lemmatiz
tags = tagger.tag_text("The striped bats were hanging on their feet and ate best fishes")
lemmas = [t.split('\t')[-1] for t in tags]
#> ['the', 'striped', 'bat', 'be', 'hang', 'on', 'their', 'foot', 'and', 'eat', 'good', 'fish']
```

Treetagger indeed does a good job in converting ‘best’ to ‘good’ and for other words as well. For further reading, refer to TreeTaggerWrapper’s documentation (https://treetaggerwrapper.readthedocs.io/en/latest/).

11. Comparing NLTK, TextBlob, spaCy, Pattern and Stanford CoreNLP

Let’s run lemmatization using the 5 implementations on the following sentence and compare output.

```
 sentence = """Following mice attacks, caring farmers were marching to Delhi for better living
Delhi police on Tuesday fired water cannons and teargas shells at protesting farmers as they tr
break barricades with their cars, automobiles and tractors."""

# NLTK
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
pprint(" ".join([lemmatizer.lemmatize(w, get_wordnet_pos(w)) for w in nltk.word_tokenize(senten
# ('Following mouse attack care farmer be march to Delhi for well living '
#  'condition Delhi police on Tuesday fire water cannon and teargas shell at '
#  'protest farmer a they try to break barricade with their car automobile and '
#  'tractor')

# Spacy
import spacy
nlp = spacy.load('en', disable=['parser', 'ner'])
doc = nlp(sentence)
pprint(" ".join([token.lemma_ for token in doc]))
# ('follow mice attack , care farmer be march to delhi for good living condition '
#  '. delhi police on tuesday fire water cannon and teargas shell at protest '
#  'farmer as -PRON- try to break barricade with -PRON- car , automobile and '
#  'tractor .')

# TextBlob
pprint(lemmatize_with_postag(sentence))
# ('Following mouse attack care farmer be march to Delhi for good living '
#  'condition Delhi police on Tuesday fire water cannon and teargas shell at '
#  'protest farmer a they try to break barricade with their car automobile and '
#  'tractor')

# Pattern
from pattern.en import lemma
pprint(" ".join([lemma(wd) for wd in sentence.split()]))
# ('follow mice attacks, care farmer be march to delhi for better live '
#  'conditions. delhi police on tuesday fire water cannon and tearga shell at '
#  'protest farmer a they try to break barricade with their cars, automobile and '
#  'tractors.')

# Stanford
pprint(lemmatize_corenlp(conn_nlp=conn_nlp, sentence=sentence))
# ('follow mouse attack care farmer be march to Delhi for better living '
#  'condition Delhi police on Tuesday fire water cannon and teargas shell at '
#  'protest farmer as they try to break barricade with they car automobile and '
#  'tractor')
```

12. Conclusion

So those are the methods you can use the text time you take up an NLP project. I would be happy to know if you have any new approaches or suggestions through your comments. Happy learning!