In [29]:
```
import spacy
```

In [30]:
```
nlp = spacy.load('en_core_web_sm')
```

In [31]:
```
doc = nlp(u"This is the first sentence. This is another sentence. This i
```

In [32]:
```
for sent in doc.sents:
    print(sent)
```

```
This is the first sentence.
This is another sentence.
This is the last sentence.
```

In [33]:
```
doc[1]
```

Out[33]:
```
is
```

In [34]:
```
# we get the second token in doc.
```

In [35]:
```
print(doc.sents[1])
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-35-2bc012eee1da> in <module>
----> 1 print(doc.sents[1])

TypeError: 'generator' object is not subscriptable
```

In [36]:
```
# To get the second sentence
list(doc.sents)[1]
```

Out[36]:
```
This is another sentence.
```

In [37]:
```
type(doc.sents)
```

Out[37]:
```
generator
```

In [38]:
```
type(list(doc.sents)[0])
```

Out[38]:
```
spacy.tokens.span.Span
```

In [39]:
```
# Another way to get a sentence
doc_sents = [sent for sent in doc.sents]
print(doc_sents[1])
```

```
This is another sentence.
```

Adding Rules

In [40]:
```
doc = nlp(u'"Management is doing things right; leadership is doing the r
```

In [41]:
```
list(doc.sents)
```

Out[41]:
```
["Management is doing things right; leadership is doing the right thin
gs" -Peter Drucker]
```

In [42]:
```
for sent in list(doc.sents):
    print(sent)
    print('------------------')
```

```
"Management is doing things right; leadership is doing the right thing
s" -Peter Drucker
------------------
```

In [43]:
```
for token in doc:
    print("token = ", token, " | token.index = ",  token.i)
```

```
token =  "  | token.index =  0
token =  Management  | token.index =  1
token =  is  | token.index =  2
token =  doing  | token.index =  3
token =  things  | token.index =  4
token =  right  | token.index =  5
token =  ;  | token.index =  6
token =  leadership  | token.index =  7
token =  is  | token.index =  8
token =  doing  | token.index =  9
token =  the  | token.index =  10
token =  right  | token.index =  11
token =  things  | token.index =  12
token =  "  | token.index =  13
token =  -Peter  | token.index =  14
token =  Drucker  | token.index =  15
```

In [44]:
```
# Add a new rule to the pipeline
def set_custom_boundaries(doc):
    for token in doc[:-1]:
        if token.text == ';':
            doc[token.i+1].is_sent_start = True
    return(doc)

nlp.add_pipe(set_custom_boundaries, before='parser')
print(nlp.pipeline)
```

```
[('tagger', <spacy.pipeline.Tagger object at 0x1295b82d0>), ('set_cust
om_boundaries', <function set_custom_boundaries at 0x12aaf4ef0>), ('pa
rser', <spacy.pipeline.DependencyParser object at 0x129a2d530>), ('ner
', <spacy.pipeline.EntityRecognizer object at 0x129a2dad0>)]
```

In [45]:
```
print(nlp.pipe_names)
```

```
['tagger', 'set_custom_boundaries', 'parser', 'ner']
```

In [47]:
```
for sent in doc.sents:
    print(sent)
    print('------------------')
```

```
"Management is doing things right; leadership is doing the right thing
s" -Peter Drucker
------------------
```

In [51]:
```
doc4 = nlp(u'"Management is doing things right; leadership is doing the
```

In [52]:
```
for sent in doc4.sents:
    print(sent)
    print('------------------')
```

```
"Management is doing things right;
------------------
leadership is doing the right things."
------------------
-Peter Drucker
------------------
```

Change Segmentation Rules

In [54]:
```
# Reload Spacy library so that we can set it back to default.
#  Now set_custom_boundaries to break sentence on semi-colon will not wo
nlp = spacy.load('en_core_web_sm')
```

In [55]:
```
mystring = u"This is a sentence. This is another sentence. \n\nThis is a
```

In [57]:
```
print(mystring)
```

```
This is a sentence. This is another sentence.

This is a
third sentence.
```

In [58]:
```
# Let us look at the default sentence segmentation
doc = nlp(mystring)
for sent in doc.sents:
    print(sent)
```

```
This is a sentence.
This is another sentence.


This is a
third sentence.
```

In [59]:
```
### Observation: This is an issue because we want the newline to be an i
# of a sentence and not a period as we see above after the first sentenc
```

In [66]:
```
# Changing the rules
from spacy.pipeline import SentenceSegmenter
```

In [67]:
```
def split_on_newline(doc):
    start = 0
    seen_newline = False
    for word in doc:
        if seen_newline:
            yield doc[start: word.i]
            start = word.i
            seen_newline = False
        elif word.text.startswith('\n'):
            seen_newline = True
    yield(doc[start: ])  # Handles the last group of tokens

    sbd = SentenceSegmenter(nlp.vocab, strategy = split_on_newline)
    nlp.add_pipe(sbd)
```

In [68]:
```
doc = nlp(mystring)
for sent in doc.sents:
    print([token.text for token in sent])
```

```
['This', 'is', 'a', 'sentence', '.']
['This', 'is', 'another', 'sentence', '.', '\n\n']
['This', 'is', 'a', '\n', 'third', 'sentence', '.']
```

In [ ]:

In [ ]: