NLTK-OSL

R. Romero Zaliz

October 2, 2018

1 Procesamiento de lenguaje natural con Python: NLTK toolkit

Rocío Romero Zaliz - rocio@decsai.ugr.es - @RCRZ_UGR
https://github.com/rcrzarg/Docencia.git

1.1 Requisitos

- Python
- NLTK
- WordCloud
- feedparser
- BeautifulSoup

1.1.1 Opcional...

- tweepy

1.2 Primeros pasos

Comenzamos descargando el NLTK de la pagina web http://www.nltk.org

```
In [1]: import nltk
        print('¡Usando NLTK version {}!'.format(nltk.__version__))
        # Hacer nltk.download() solo la primera vez
```

```
¡Usando NLTK version 3.3!
```

1.3 Procesando nuestro primer texto

1.3.1 Cargo un texto

```
In [2]: import urllib.request
        url = "http://www.gutenberg.org/files/2554/2554-0.txt"
        raw = urllib.request.urlopen(url).read()
```

1.3.2 Inspecciono el texto

```
In [3]: type(raw)
```

```
Out[3]: bytes
```

```
In [4]: len(raw)
```

```
Out[4]: 1201733
```

```
In [5]: raw[:80]
```

```
Out[5]: b'\xef\xbb\xbfThe Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevsky\
```

1.3.3 En mi caso debo limpiar el texto a un formato legible

```
In [6]: text = raw.decode("utf-8")
        text[:78]
```

```
Out[6]: '\ufeffThe Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevsky\r\n\r\n
```

```
In [7]: import re

        text = re.sub('\n+', ' ', text)
        plain_text = re.sub('[^A-Za-z0-9 ,\.;:\-\'\"]+', '', text)
        plain_text[:73]
```

```
Out[7]: 'The Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevsky'
```

```
In [8]: type(plain_text)
```

```
Out[8]: str
```

1.4 Analizando el texto

1.4.1 Frases

```
In [9]: sentences = nltk.sent_tokenize(plain_text)
        sentences[0:5]
```

```
Out[9]: ['The Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevsky  This eBook
         'You may copy it, give it away or re-use it under the terms of the Project Gutenberg Li
         'Dostoevsky was the son of a doctor.',
         'His parents were very hard-working and deeply religious people, but so poor that they
         'The father and mother spent their evenings in reading aloud to their children, general
```

1.4.2 Tokens

```
In [10]: print(sentences[22])

         tokens = nltk.word_tokenize(sentences[22])
         tokens
```

```
The intense suffering of this experience left a lasting stamp on Dostoevsky's mind.
```

```
Out[10]: ['The',
          'intense',
          'suffering',
          'of',
          'this',
          'experience',
          'left',
          'a',
          'lasting',
          'stamp',
          'on',
          'Dostoevsky',
          '',
          's',
          'mind',
          '.']
```

1.4.3 Stems

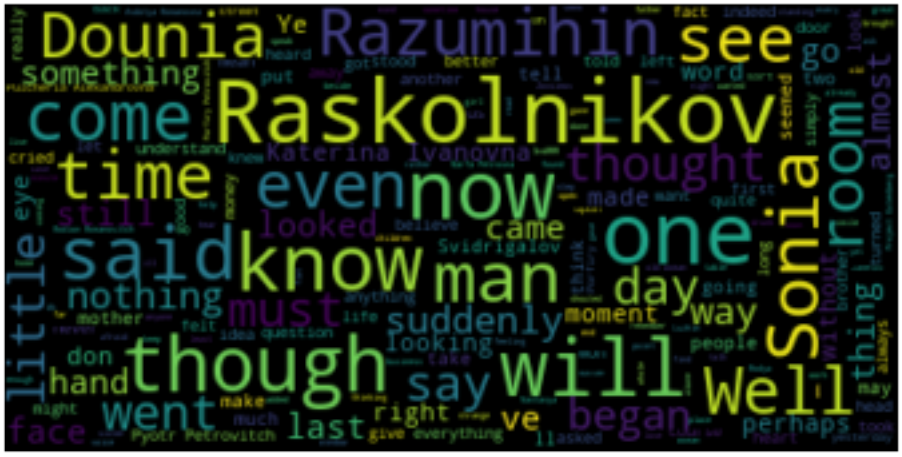
```
In [11]: porter = nltk.PorterStemmer()
         stems = [porter.stem(t) for t in tokens]
         stems
```

```
Out[11]: ['the',
          'intens',
          'suffer',
          'of',
          'thi',
          'experi',
          'left',
          'a',
          'last',
          'stamp',
          'on',
          'dostoevski',
          '',
          's',
          'mind',
          '.']
```

In [12]:
```
# Para instalar WordCloud: en una terminal -> pip install wordcloud
from wordcloud import WordCloud

wordcloud = WordCloud(max_font_size=40).generate(plain_text)

import matplotlib.pyplot as plt
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [13]:
```
all_tokens = [nltk.word_tokenize(t) for t in sentences]
all_stems = [porter.stem(k) for t in all_tokens for k in t]

wordcloud = WordCloud(max_font_size=40).generate(' '.join(all_stems))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

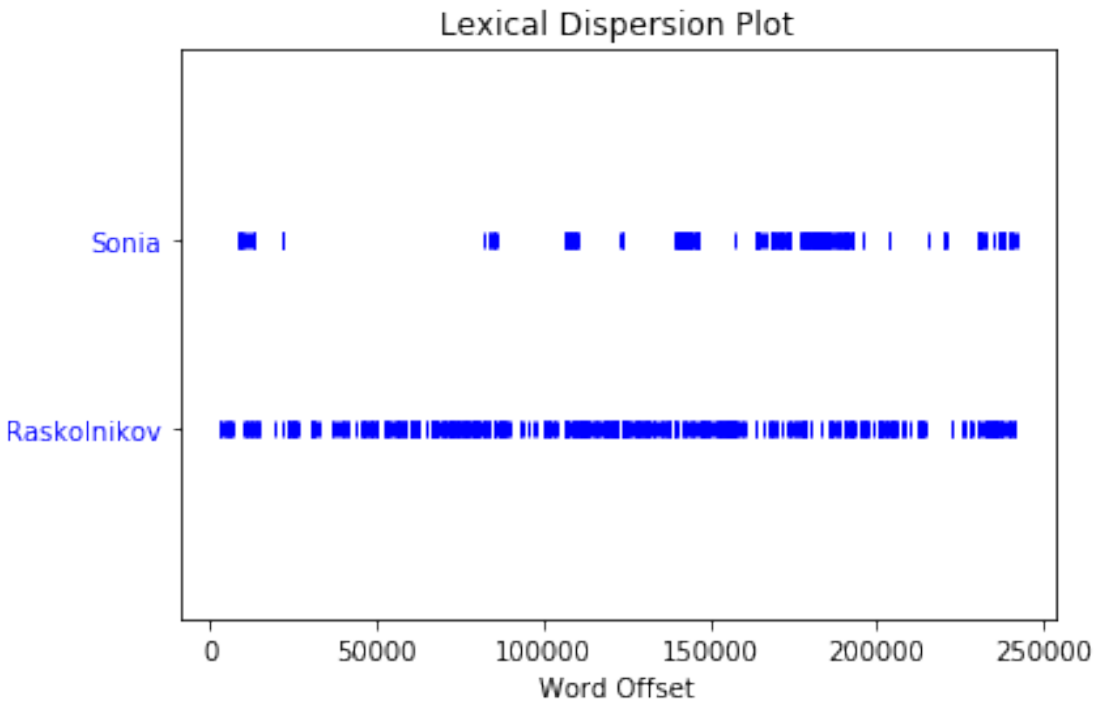
1.4.4 Lemmas

```
In [14]: wnl = nltk.stem.WordNetLemmatizer()
         lemmas = [wnl.lemmatize(t) for t in tokens]
         lemmas # Algo no va bien...

Out[14]: ['The',
          'intense',
          'suffering',
          'of',
          'this',
          'experience',
          'left',
          'a',
          'lasting',
          'stamp',
          'on',
          'Dostoevsky',
          '',
          's',
          'mind',
          '.']

In [15]: full_text = sum(all_tokens, [])
         full_text = nltk.Text(full_text)

In [16]: full_text.dispersion_plot(["Sonia","Raskolnikov"])
```




```
In [17]: full_text.similar("heart")

face mind head mother room eyes hand brother life way wife voice
sister money name illness lips words door hands


In [18]: fdist1 = nltk.FreqDist(full_text)
         fdist1.hapaxes()[:10]

Out[18]: ['Title',
          'Author',
          'Release',
          'Date',
          'March',
          '28',
          '2006',
          '2554',
          'October',
          '27']

In [19]: fdist1.most_common(15)

Out[19]: [(',', 16178),
          ('.', 10454),
          ('the', 7444),
```

```
 ('and', 6284),
 ('to', 5284),
 ('a', 4472),
 ('I', 4399),
 ('’', 4046),
 ('of', 3849),
 ('he', 3535),
 ('you', 3506),
 ('in', 3101),
 ('that', 3082),
 ('it', 2943),
 ('was', 2803)]
```

1.4.5 Part of speech (POS)

```
In [20]: tags = nltk.pos_tag(tokens)
         tags
```

```
Out[20]: [('The', 'DT'),
          ('intense', 'JJ'),
          ('suffering', 'NN'),
          ('of', 'IN'),
          ('this', 'DT'),
          ('experience', 'NN'),
          ('left', 'VBD'),
          ('a', 'DT'),
          ('lasting', 'JJ'),
          ('stamp', 'NN'),
          ('on', 'IN'),
          ('Dostoevsky', 'NNP'),
          ('’', 'NNP'),
          ('s', 'NN'),
          ('mind', 'NN'),
          ('.', '.')]
```

```
In [21]: #nltk.download('tagsets')
         print(nltk.help.upenn_tagset('DT'))
         print(nltk.help.upenn_tagset('IN'))
```

```
DT: determiner
    all an another any both del each either every half la many much nary
    neither no some such that the them these this those
None
IN: preposition or conjunction, subordinating
    astride among uppon whether out inside pro despite on by throughout
    below within for towards near behind atop around if like until below
    next into if beside ...
None
```

In [22]:
```
from nltk.corpus import wordnet

def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

lemmas = [wnl.lemmatize(t[0], get_wordnet_pos(t[1])) for t in tags]
lemmas
```

Out[22]:
```
['The',
 'intense',
 'suffering',
 'of',
 'this',
 'experience',
 'leave',
 'a',
 'lasting',
 'stamp',
 'on',
 'Dostoevsky',
 '',
 's',
 'mind',
 '.']
```

In [23]:
```
all_tags = [nltk.pos_tag(t) for t in all_tokens]
all_lemmas = [wnl.lemmatize(k[0], get_wordnet_pos(k[1])) for t in all_tags for k in t]

wordcloud = WordCloud(max_font_size=40).generate(' '.join(all_lemmas))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```


```
In [24]: 'said' in all_lemmas
```

```
Out[24]: False
```

```
In [25]: 'say' in all_lemmas
```

```
Out[25]: True
```

```
In [26]: full_tags = sum(all_tags, [])
         all_lemmas_v = [all_lemmas[t] for t in range(0,len(all_lemmas)-1) if full_tags[t][1][0]

         wordcloud = WordCloud(max_font_size=40).generate(' '.join(all_lemmas_v))
         plt.imshow(wordcloud, interpolation='bilinear')
         plt.axis("off")
         plt.show()
```

```
In [27]: all_lemmas_n = [all_lemmas[t] for t in range(0,len(all_lemmas)-1) if full_tags[t][1][0]

         wordcloud = WordCloud(max_font_size=40).generate(' '.join(all_lemmas_n))
         plt.imshow(wordcloud, interpolation='bilinear')
         plt.axis("off")
         plt.show()
```

1.4.6 Entidades nombradas

- ORGANIZATION
- PERSON
- GPE (geo-political entities)

```
In [28]: tokens = nltk.word_tokenize(sentences[11])
         tags = nltk.pos_tag(tokens)
         result = nltk.ne_chunk(tags)
         result
```

Out[28]:


```
In [29]: type(result)
```

```
Out[29]: nltk.tree.Tree

In [30]: print(result)

(S
  Though/IN
  neither/RB
  by/IN
  temperament/JJ
  nor/CC
  conviction/NN
  a/DT
  revolutionist/NN
  ,/,
  (PERSON Dostoevsky/NNP)
  was/VBD
  one/CD
  of/IN
  a/DT
  little/JJ
  group/NN
  of/IN
  young/JJ
  men/NNS
  who/WP
  met/VBD
  together/RB
  to/TO
  read/VB
  (PERSON Fourier/NNP)
  and/CC
  (PERSON Proudhon/NNP)
  ./.)


In [31]: tokens = nltk.word_tokenize(sentences[13])
         tags = nltk.pos_tag(tokens)
         nltk.ne_chunk(tags)

   Out[31]:
```

1.4.7 Fragmentación

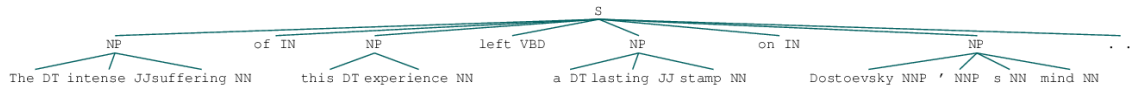
```
In [32]: grammar = "NP: {<DT>?<JJ>*<CD|NN|NNP|NNS>*}"

         tokens = nltk.word_tokenize(sentences[14])
         tags = nltk.pos_tag(tokens)
         cp = nltk.RegexpParser(grammar)
         result = cp.parse(tags)
         result
```

Out[32]:


```
In [33]: tokens = nltk.word_tokenize(sentences[22])
         tags = nltk.pos_tag(tokens)
         cp = nltk.RegexpParser(grammar)
         result = cp.parse(tags)
         result
```

Out[33]:



1.5 Crear nuestros propios POS taggers

1.5.1 Default tagger

```
In [34]: from nltk.corpus import brown
         brown_tagged_sents = brown.tagged_sents(categories='news')
         brown_sents = brown.sents(categories='news')
         type(brown_tagged_sents)
```

```
Out[34]: nltk.corpus.reader.util.ConcatenatedCorpusView
```

```
In [35]: brown_tagged_sents[1]
```

```
Out[35]: [('The', 'AT'),
          ('jury', 'NN'),
          ('further', 'RBR'),
          ('said', 'VBD'),
          ('in', 'IN'),
```

```
 ('term-end', 'NN'),
 ('presentments', 'NNS'),
 ('that', 'CS'),
 ('the', 'AT'),
 ('City', 'NN-TL'),
 ('Executive', 'JJ-TL'),
 ('Committee', 'NN-TL'),
 (',', ','),
 ('which', 'WDT'),
 ('had', 'HVD'),
 ('over-all', 'JJ'),
 ('charge', 'NN'),
 ('of', 'IN'),
 ('the', 'AT'),
 ('election', 'NN'),
 (',', ','),
 ('``', '``'),
 ('deserves', 'VBZ'),
 ('the', 'AT'),
 ('praise', 'NN'),
 ('and', 'CC'),
 ('thanks', 'NNS'),
 ('of', 'IN'),
 ('the', 'AT'),
 ('City', 'NN-TL'),
 ('of', 'IN-TL'),
 ('Atlanta', 'NP-TL'),
 ("''", "''"),
 ('for', 'IN'),
 ('the', 'AT'),
 ('manner', 'NN'),
 ('in', 'IN'),
 ('which', 'WDT'),
 ('the', 'AT'),
 ('election', 'NN'),
 ('was', 'BEDZ'),
 ('conducted', 'VBN'),
 ('.', '.')]
```

```
In [36]: brown_tags = [k[1] for t in brown_tagged_sents for k in t]
         nltk.FreqDist(brown_tags).max()
```

```
Out[36]: 'NN'
```

```
In [37]: tokens = nltk.word_tokenize(sentences[22])
         default_tagger = nltk.DefaultTagger('NN')
         default_tagger.tag(tokens)
```

```
Out[37]: [('The', 'NN'),
          ('intense', 'NN'),
```

```
 ('suffering', 'NN'),
 ('of', 'NN'),
 ('this', 'NN'),
 ('experience', 'NN'),
 ('left', 'NN'),
 ('a', 'NN'),
 ('lasting', 'NN'),
 ('stamp', 'NN'),
 ('on', 'NN'),
 ('Dostoevsky', 'NN'),
 ('', 'NN'),
 ('s', 'NN'),
 ('mind', 'NN'),
 ('.', 'NN')]
```

```
In [38]: default_tagger.evaluate(brown_tagged_sents)
```

```
Out[38]: 0.13089484257215028
```

1.5.2 Regular Expression Tagger

```
In [39]: patterns = [
             (r'.*ing$', 'VBG'),               # gerunds
             (r'.*ed$', 'VBD'),                # simple past
             (r'.*es$', 'VBZ'),                # 3rd singular present
             (r'.*ould$', 'MD'),               # modals
             (r'.*\'s$', 'NN$'),               # possessive nouns
             (r'.*s$', 'NNS'),                 # plural nouns
             (r'^-?[0-9]+(.[0-9]+)?$', 'CD'),  # cardinal numbers
             (r'.*', 'NN')                     # nouns (default)
         ]
         regexp_tagger = nltk.RegexpTagger(patterns)
         regexp_tagger.tag(tokens)
```

```
Out[39]: [('The', 'NN'),
          ('intense', 'NN'),
          ('suffering', 'VBG'),
          ('of', 'NN'),
          ('this', 'NNS'),
          ('experience', 'NN'),
          ('left', 'NN'),
          ('a', 'NN'),
          ('lasting', 'VBG'),
          ('stamp', 'NN'),
          ('on', 'NN'),
          ('Dostoevsky', 'NN'),
          ('', 'NN'),
          ('s', 'NNS'),
```

```
          ('mind', 'NN'),
          ('.', 'NN')]
```

```
In [40]: regexp_tagger.evaluate(brown_tagged_sents)
```

```
Out[40]: 0.20326391789486245
```

1.5.3 N-Gram tagger

```
In [41]: size = int(len(brown_tagged_sents) * 0.9)
         unigram_tagger = nltk.UnigramTagger(brown_tagged_sents[:size])
         unigram_tagger.tag(brown_sents[2007])
```

```
Out[41]: [('Various', 'JJ'),
          ('of', 'IN'),
          ('the', 'AT'),
          ('apartments', 'NNS'),
          ('are', 'BER'),
          ('of', 'IN'),
          ('the', 'AT'),
          ('terrace', 'NN'),
          ('type', 'NN'),
          (',', ','),
          ('being', 'BEG'),
          ('on', 'IN'),
          ('the', 'AT'),
          ('ground', 'NN'),
          ('floor', 'NN'),
          ('so', 'QL'),
          ('that', 'CS'),
          ('entrance', 'NN'),
          ('is', 'BEZ'),
          ('direct', 'JJ'),
          ('.', '.')]
```

```
In [42]: brown_tagged_sents
```

```
Out[42]: [[('The', 'AT'), ('Fulton', 'NP-TL'), ('County', 'NN-TL'), ('Grand', 'JJ-TL'), ('Jury',
```

```
In [43]: unigram_tagger.evaluate(brown_tagged_sents[size:])
```

```
Out[43]: 0.8121200039868434
```

```
In [44]: bigram_tagger = nltk.BigramTagger(brown_tagged_sents[:size])
         bigram_tagger.tag(brown_sents[2007])
```

```
Out[44]: [('Various', 'JJ'),
          ('of', 'IN'),
          ('the', 'AT'),
```

```
          ('apartments', 'NNS'),
          ('are', 'BER'),
          ('of', 'IN'),
          ('the', 'AT'),
          ('terrace', 'NN'),
          ('type', 'NN'),
          (',', ','),
          ('being', 'BEG'),
          ('on', 'IN'),
          ('the', 'AT'),
          ('ground', 'NN'),
          ('floor', 'NN'),
          ('so', 'CS'),
          ('that', 'CS'),
          ('entrance', 'NN'),
          ('is', 'BEZ'),
          ('direct', 'JJ'),
          ('.', '.')]
```

```
In [45]: bigram_tagger.evaluate(brown_tagged_sents[size:])
```

```
Out[45]: 0.10206319146815508
```

```
In [46]: t0 = nltk.DefaultTagger('NN')
         t1 = nltk.UnigramTagger(brown_tagged_sents[:size], backoff=t0)
         t2 = nltk.BigramTagger(brown_tagged_sents[:size], backoff=t1)
         t2.evaluate(brown_tagged_sents[size:])
```

```
Out[46]: 0.8452108043456593
```

```
In [47]: real = [k[1] for t in brown_tagged_sents[size:] for k in t]
         calculated = [k[1] for t in [t2.tag(t) for t in brown_sents[size:]] for k in t]

         cm = nltk.ConfusionMatrix(real, calculated)
         print(cm.pretty_format(sort_by_count=True, show_percents=True, truncate=10))
```

```
    |                           N                                           |
    |      N      I      A      N             J             N      C      V |
    |      N      N      T      S      ,      J      .      P      C      B |
----+-----------------------------------------------------------------------+
 NN | <12.0%>     .   0.0%   0.1%      .   0.2%      .      .      .   0.2% |
 IN |   0.0% <10.1%>     .      .      .      .      .      .      .      . |
 AT |      .      . <8.5%>     .      .      .      .      .      .      . |
NNS |   1.9%      .      . <4.3%>     .      .      .      .      .      . |
  , |      .      .      .      . <6.0%>     .      .      .      .      . |
 JJ |   1.3%      .      .      .      . <3.4%>     .   0.0%      .   0.0% |
  . |      .      .      .      .      .      . <4.8%>     .      .      . |
 NP |   1.7%      .      .      .      .      .      . <2.7%>     .      . |
 CC |      .      .      .      .      .      .      .      . <2.8%>     . |
```

```
 VB |   0.7%      .      .      .      .   0.0%      .      .      .  <2.0%>|
----+---------------------------------------------------------------------+
(row = reference; col = test)
```

1.5.4 Arreglando el etiquetado

```
In [48]: templates = nltk.tag.brill.fntbl37()
         templates
```

```
Out[48]: [Template(Word([0]),Word([1]),Word([2])),
          Template(Word([-1]),Word([0]),Word([1])),
          Template(Word([0]),Word([-1])),
          Template(Word([0]),Word([1])),
          Template(Word([0]),Word([2])),
          Template(Word([0]),Word([-2])),
          Template(Word([1, 2])),
          Template(Word([-2, -1])),
          Template(Word([1, 2, 3])),
          Template(Word([-3, -2, -1])),
          Template(Word([0]),Pos([2])),
          Template(Word([0]),Pos([-2])),
          Template(Word([0]),Pos([1])),
          Template(Word([0]),Pos([-1])),
          Template(Word([0])),
          Template(Word([-2])),
          Template(Word([2])),
          Template(Word([1])),
          Template(Word([-1])),
          Template(Pos([-1]),Pos([1])),
          Template(Pos([1]),Pos([2])),
          Template(Pos([-1]),Pos([-2])),
          Template(Pos([1])),
          Template(Pos([-1])),
          Template(Pos([-2])),
          Template(Pos([2])),
          Template(Pos([1, 2, 3])),
          Template(Pos([1, 2])),
          Template(Pos([-3, -2, -1])),
          Template(Pos([-2, -1])),
          Template(Pos([1]),Word([0]),Word([1])),
          Template(Pos([1]),Word([0]),Word([-1])),
          Template(Pos([-1]),Word([-1]),Word([0])),
          Template(Pos([-1]),Word([0]),Word([1])),
          Template(Pos([-2]),Pos([-1])),
          Template(Pos([1]),Pos([2])),
          Template(Pos([1]),Pos([2]),Word([1]))]
```

```
In [49]: t3 = nltk.BrillTaggerTrainer(t2, templates)
         t3

Out[49]: <nltk.tag.brill_trainer.BrillTaggerTrainer at 0x1a30588c18>

In [50]: braubt_tagger = t3.train(brown_tagged_sents[:size], max_rules=100, min_score=3)
         braubt_tagger.evaluate(brown_tagged_sents[size:])

Out[50]: 0.8525864646666003

In [51]: t3 = nltk.BrillTaggerTrainer(t2, templates, trace=3)
         braubt_tagger = t3.train(brown_tagged_sents[:size], max_rules=100, min_score=3)

TBL train (fast) (seqs: 4160; tokens: 90521; tpls: 37; min score: 3; min acc: None)
Finding initial useful rules...
    Found 62455 useful rules.

           B      |
   S   F   r   O  |        Score = Fixed - Broken
   c   i   o   t  |  R     Fixed = num tags changed incorrect -> correct
   o   x   k   h  |  u     Broken = num tags changed correct -> incorrect
   r   e   e   e  |  l     Other = num tags changed incorrect -> incorrect
   e   d   n   r  |  e
------------------+-------------------------------------------------------
 260 260   0   1  | TO->IN if Pos:AT@[1]
  81  81   0   1  | TO->IN if Pos:NP@[1]
  52  52   0   0  | TO->IN if Word:to@[0] & Pos:NN@[1]
  43  43   0   0  | TO->IN if Pos:NNS@[1]
  42  43   1   0  | CS->QL if Word:as@[1,2,3]
  34  36   2   0  | TO->IN if Word:to@[0] & Pos:JJ@[1]
  28  28   0   0  | IN->TO if Word:to@[0] & Pos:VB@[1]
  25  25   0   0  | TO->IN if Pos:PP$@[1]
  22  22   0   0  | TO->IN if Word:to@[0] & Pos:CD@[1]
  18  18   0   0  | CS->WPS if Word:that@[0] & Pos:VBD@[1]
  18  19   1   0  | IN->RP if Word:in@[0] & Pos:IN@[1]
  18  24   6   1  | TO->IN if Pos:NN-TL@[1,2]
  13  13   0   1  | CS->WPS if Pos:NN@[-1] & Pos:MD@[1]
  10  10   0   0  | PPO->PPS if Word:it@[0] & Word:said@[-1]
  10  10   0   0  | IN->TO if Word:to@[0] & Word:be@[1]
  10  10   0   0  | TO->IN if Word:to@[0] & Pos:PPO@[1]
  10  10   0   0  | WPS->CS if Word:that@[0] & Pos:AT@[1]
  10  20  10   0  | NP->NP-TL if Pos:AT@[-1] & Pos:NN-TL@[1]
   9   9   0   0  | CS->WPS if Word:that@[0] & Pos:VBZ@[1]
   9  10   1   0  | JJ->JJ-TL if Word:American@[0] & Pos:NN-TL@[1]
   9   9   0   0  | TO->IN if Word:to@[0] & Pos:JJ-TL@[1]
   9   9   0   1  | QL->AP if Word:than@[1]
   8   8   0   4  | CS->WPS if Word:that@[0] & Word:is@[1]
   8   8   0   0  | JJ->ABL if Word:such@[0] & Word:a@[1]
   8   9   1   0  | NP->NP-TL if Word:County@[1,2]
```

```
8   9   1   0  | AT-TL->AT if Word:the@[0] & Pos:NN-TL@[2]
8  10   2   0  | AP->QL if Word:more@[0] & Pos:JJ@[1]
8   8   0   0  | IN->CS if Word:before@[0] & Pos:VBG@[1]
8   9   1   0  | IN-TL->IN if Word:of@[0] & Pos:AT@[1]
8   8   0   0  | TO->IN if Word:to@[0] & Pos:AP@[1]
8  14   6   0  | JJ->NN if Pos:AT@[-1] & Pos:IN@[1]
7   8   1   0  | IN->RB if Word:about@[0] & Pos:CD@[1]
7   7   0   0  | NN->JJ if Word:past@[0] & Pos:NN@[1]
7   8   1   0  | TO->IN if Word:to@[0] & Pos:VBG@[1]
7   7   0   0  | WPS->CS if Word:that@[0] & Pos:PPS@[1]
7   7   0   0  | JJS->NN if Word:of@[1]
7   8   1   0  | VBN->VBD if Word:that@[-1]
7   7   0   0  | CS->DT if Pos:VB@[-1] & Pos:NN@[1]
7   7   0   0  | IN->CS if Pos:PPS@[1]
7   9   2   5  | QL->JJ if Pos:NN@[1]
6   6   0   0  | CS->IN if Word:after@[0] & Word:a@[1]
6   6   0   0  | CS->WPS if Word:that@[0] & Word:has@[1]
6   7   1   1  | CS->WPS if Word:that@[0] & Pos:VBD@[1]
6   7   1   0  | IN->RP if Word:on@[0] & Pos:IN@[1]
6   6   0   0  | IN->TO if Word:to@[0] & Pos:VB@[1]
6   6   0   0  | TO->IN if Word:to@[0] & Pos:CS@[1]
6   6   0   0  | NP-TL->NP if Pos:AT@[-1] & Pos:NN@[1]
5   6   1   0  | IN->IN-TL if Word:on@[0] & Word:Committee@[-1]
5   6   1   0  | CS->IN if Word:as@[0] & Word:to@[1]
5   5   0   0  | JJ->RB if Word:long@[0] & Word:as@[1]
5   5   0   0  | NR->NN if Word:home@[0] & Word:run@[1]
5   5   0   0  | RB->IN if Word:around@[0] & Word:the@[1]
5   6   1   0  | NP->NP-TL if Word:University@[1,2]
5   5   0   0  | NN->AP if Word:past@[0] & Pos:NNS@[2]
5   5   0   0  | CS->IN if Word:after@[0] & Pos:CD@[1]
5   5   0   0  | TO->IN if Word:to@[0] & Pos:NP$@[1]
5   5   0   0  | TO->IN if Word:to@[0] & Pos:PN@[1]
5   6   1   0  | NP->NP-TL if Pos:IN@[-1] & Pos:JJ-TL@[1]
5   5   0   1  | NP-HL->NP if Pos:VBD@[1,2,3]
4   4   0   0  | IN->RP if Word:on@[0] & Word:to@[1]
4   4   0   0  | NR->NN if Word:home@[0] & Word:runs@[1]
4   4   0   0  | PP$->PPO if Word:her@[0] & Word:.@[1]
4   4   0   0  | CS->IN if Word:after@[0] & Word:a@[-2]
4   4   0   0  | CS->DT if Word:''@[1,2]
4   4   0   0  | NP->NP-TL if Word:Government@[1,2]
4   4   0   0  | CC->CC-TL if Word:and@[0] & Pos:NNS-TL@[1]
4   4   0   0  | PPO->PP$ if Word:her@[0] & Pos:NN@[1]
4   8   4   0  | RP->IN if Word:down@[0] & Pos:AT@[1]
4   4   0   0  | TO->IN if Word:to@[0] & Pos:DTI@[1]
4   4   0   0  | VBN->VBD if Word:made@[0] & Pos:NNS@[1]
4   4   0   0  | NR-TL->JJ-TL if Pos:AT@[-1] & Pos:NN-TL@[1]
4   5   1   0  | VBN->VBD if Pos:NN@[-1] & Pos:AT@[1]
4   4   0   0  | QL->CS if Pos:CS@[1]
```

```
4   5   1   0  | CS->RB if Pos:NN@[-1] & Word:because@[0] & Word:of@[1]
3   3   0   0  | JJ->JJ-TL if Word:the@[-1] & Word:American@[0] &
                |   Word:League's@[1]
3   3   0   0  | IN->RP if Word:in@[0] & Word:bring@[-1]
3   3   0   0  | IN->RP if Word:in@[0] & Word:drove@[-1]
3   3   0   0  | PN->CD if Word:one@[0] & Word:the@[-1]
3   3   0   0  | PPS->PPO if Word:it@[0] & Word:to@[-1]
3   4   1   0  | CS->DT if Word:that@[0] & Word:of@[1]
3   3   0   0  | CS->WPS if Word:that@[0] & Word:had@[1]
3   3   0   0  | IN->RB if Word:before@[0] & Word:,@[1]
3   3   0   0  | IN->RP if Word:in@[0] & Word:,@[1]
3   3   0   1  | NN-TL->NP-TL if Word:St.@[0] & Word:Louis@[1]
3   3   0   0  | RB->EX if Word:there@[0] & Word:is@[1]
3   3   0   0  | RB->IN if Word:along@[0] & Word:the@[1]
3   7   4   0  | RP->IN if Word:off@[0] & Word:the@[1]
4   4   0   0  | IN->RP if Word:off@[0] & Pos:VB@[-1]
3   3   0   0  | TO->IN if Word:to@[0] & Word:about@[1]
3   3   0   0  | TO->IN if Word:to@[0] & Word:it@[1]
3   3   0   0  | TO->IN if Word:to@[0] & Word:left@[1]
3   3   0   0  | WRB->QL if Word:how@[0] & Word:good@[1]
3   3   0   0  | NP-TL->NP if Word:open@[1,2]
3   3   0   0  | IN->IN-TL if Word:Crusade@[-2,-1]
3   3   0   0  | IN-TL->IN if Word:on@[1,2,3]
3   3   0   0  | IN->CS if Word:after@[0] & Pos:VBD@[2]
3   3   0   0  | IN->CS if Word:before@[0] & Pos:VBD@[2]
4   4   0   0  | JJ->RB if Pos:CS@[1] & Pos:CD@[2]
3   3   0   0  | AT->AT-TL if Word:The@[0] & Pos:IN@[-2]
3   3   0   0  | ABL->JJ if Word:such@[0] & Pos:NN@[1]
```

1.6 ¡Manos a la obra!

Podéis probar descargar texto de diferentes fuentes.

1.6.1 Cómo leer de un RSS

```
In [52]: import feedparser
         rss_text = feedparser.parse("http://osl.ugr.es/?feed=atom")
         rss_text['feed']['title']

Out[52]: 'Oficina de Software Libre'

In [53]: rss_text.entries[0]

Out[53]: {'author': 'Maria Isabel García Arenas',
          'author_detail': {'name': 'Maria Isabel García Arenas'},
          'authors': [{'name': 'Maria Isabel García Arenas'}],
          'content': [{'base': 'http://osl.ugr.es/2018/09/26/jornadas-de-software-libre-de-la-un
            'language': 'es-ES',
```

```
  'type': 'text/html',
  'value': '<p><img class="size-medium wp-image-9542 alignleft" src="http://osl.ugr.es
 'guidislink': False,
 'id': 'http://osl.ugr.es/?p=9535',
 'link': 'http://osl.ugr.es/2018/09/26/jornadas-de-software-libre-de-la-universidad-de-
 'links': [{'href': 'http://osl.ugr.es/2018/09/26/jornadas-de-software-libre-de-la-univ
  'rel': 'alternate',
  'type': 'text/html'},
 {'count': '0',
  'href': 'http://osl.ugr.es/2018/09/26/jornadas-de-software-libre-de-la-universidad-d
  'rel': 'replies',
  'thr:count': '0',
  'type': 'text/html'},
 {'count': '0',
  'href': 'http://osl.ugr.es/2018/09/26/jornadas-de-software-libre-de-la-universidad-d
  'rel': 'replies',
  'thr:count': '0',
  'type': 'application/atom+xml'}],
 'published': '2018-09-26T18:24:10Z',
 'published_parsed': time.struct_time(tm_year=2018, tm_mon=9, tm_mday=26, tm_hour=18, t
 'summary': 'Como ya venimos avisando desde hace meses, este jueves 28 de septiembre, c
 'summary_detail': {'base': 'http://osl.ugr.es/wp-atom.php',
  'language': 'es-ES',
  'type': 'text/html',
  'value': 'Como ya venimos avisando desde hace meses, este jueves 28 de septiembre, co
 'tags': [{'label': None, 'scheme': 'http://osl.ugr.es', 'term': 'Cronica'},
  {'label': None, 'scheme': 'http://osl.ugr.es', 'term': 'Destacados'},
  {'label': None, 'scheme': 'http://osl.ugr.es', 'term': 'Eventos'},
  {'label': None, 'scheme': 'http://osl.ugr.es', 'term': 'Sin categoría'}],
 'thr_total': '0',
 'title': 'Jornadas de Software Libre de la Universidad de Granada',
 'title_detail': {'base': 'http://osl.ugr.es/wp-atom.php',
  'language': 'es-ES',
  'type': 'text/html',
  'value': 'Jornadas de Software Libre de la Universidad de Granada'},
 'updated': '2018-09-26T18:28:52Z',
 'updated_parsed': time.struct_time(tm_year=2018, tm_mon=9, tm_mday=26, tm_hour=18, tm_
```

In [54]:
```
entrada = rss_text.entries[0].content[0]['value']
entrada
```

Out[54]:
```
'<p><img class="size-medium wp-image-9542 alignleft" src="http://osl.ugr.es/wp-content/
```

In [55]:
```
from bs4 import BeautifulSoup

soup = BeautifulSoup(entrada, 'html.parser')
rss_text = soup.get_text()
rss_text
```

Out[55]:
```
'Como ya venimos avisando desde hace meses, este jueves 28 de septiembre, comienzan las
```

1.6.2 Cómo leer de un PDF

```
In [56]: from PyPDF2 import PdfFileReader

         pdf_text = PdfFileReader(open("Parcial2_Practico_Grupo_A.pdf", "rb"))
         pdf_text = pdf_text.getPage(0).extractText().encode("utf8")
         pdf_text
```

```
Out[56]: b'Fundamentos de Bases de Datos. Grupo A \nParcial 2. Ejercicio Pr\xe2\x81\x84ctico \nN
```

```
In [57]: pdf_text = PdfFileReader(open("programa_jsl.pdf", "rb"))
         pdf_text.getNumPages()
```

```
Out[57]: 2
```

```
In [58]: pdf_text.getPage(0).extractText().encode("utf8")
```

```
Out[58]: b'\n\n\n\n\xcb\x98\xcb\x87\xcb\x86\n\xcb\x99\xcb\x9d\xcb\x9c\n\xcb\x99\xcb\x9d \n!"#%!\
```

1.6.3 Cómo leer de Twitter

Si teneis cuenta de desarrollador en Twitter podeis usarla o crear una nueva. Deberéis instalar algunas cosillas siguiendo este tutorial: http://www.nltk.org/howto/twitter.html

```
In [59]: from tweepy import OAuthHandler

         consumer_key = 'YOUR-CONSUMER-KEY'
         consumer_secret = 'YOUR-CONSUMER-SECRET'
         access_token = 'YOUR-ACCESS-TOKEN'
         access_secret = 'YOUR-ACCESS-SECRET'
```

```
In [64]: import os

         tfile = os.environ['TWITTER']

         f = open(tfile + 'credentials.txt', "r")
         dict = {}
         for line in f:
             out = line. partition('=')
             dict[out[0]] = out[2].rstrip()

         consumer_key = dict['app_key']
         consumer_secret = dict['app_secret']
         access_token = dict['oauth_token']
         access_secret = dict['oauth_token_secret']
```

```
In [66]: import tweepy

         auth = OAuthHandler(consumer_key, consumer_secret)
         auth.set_access_token(access_token, access_secret)

         api = tweepy.API(auth)
```

```
In [67]: for status in tweepy.Cursor(api.home_timeline).items(3):
             print(status.text)
             print('---')

¡El Buque Oceanográfico Sarmiento de Gamboa del @CSIC es un auténtico centro de investigación fl
---
Estic a Barcelona coneixent alguns dels projectes científics recolzats pel Ministeri. Estem orgu
---
RT @Dimatematicas: She Does Maths: Amalia Pizarro

#DivulgandoMatemáticas
 https://t.co/NXv11cg5JN vía @mujerconciencia
---


In [68]: myTweet = tweepy.Cursor(api.search, q='#JSLUGR18').items(5)

         for tweet in myTweet:
             print(tweet.created_at, tweet.text)
             print('---')

2018-09-30 08:01:26 RT @RCRZ_UGR: Para cuando la próxima? Ya tengo mono #JSLUGR18 https://t.co/I
---
2018-09-30 01:56:28 RT @OSLUGR: Terminamos la tarde con el taller ‘‘OpenFOAM para simulación en m
---
2018-09-29 22:00:00 RT @GuyikCGG: Hablando de Ciberseguridad en las Jornadas de Software libre d
---
2018-09-29 14:26:40 RT @RCRZ_UGR: Para cuando la próxima? Ya tengo mono #JSLUGR18 https://t.co/I
---
2018-09-29 14:25:06 RT @RCRZ_UGR: Para cuando la próxima? Ya tengo mono #JSLUGR18 https://t.co/I
---
```

1.6.4 Probemos con un texto en español...

- Probar dividir en frases, tokens, lemmas, etc.

```
In [69]: sentences = nltk.sent_tokenize(rss_text)
         sentences

Out[69]: ['Como ya venimos avisando desde hace meses, este jueves 28 de septiembre, comienzan la
          'En ellas queremos acercar a toda la comunidad, universitaria o no, algunas herramient
          'Para ello abrimos convocatoria de propuestas, seleccionamos entre ellas y elegimos la
          'Nuestra idea es convenceros con estas opciones libres, puesto que son opciones válida
          'En esta primera edición de las Jornadas podréis aprender sobre temas diversos, tanto
          'Además, contaremos con una ponencia inaugural y otra de clausura que llevarán a cabo
          'En concreto, la inauguración está a cargo de Albert Astals, miembro de la comunidad d
          'Para la clausura hemos invitado a\xa0Remedios Fernández\xa0es experta en tecnologías
          'Autora del blog\xa0MomandGeek.com, autora de la sección tecnológica #AlmeríaTech de\x
```

```
 'Ella nos contará cómo se puede enseñar, a todo el mundo, usando Software Libre, y nos
 'También queremos aprovechar este post, para dar las gracias por la acogida de esta in
 'En números, tenemos\xa0 más de 100 personas inscritas, más de 20 ponentes, y en defin
 'Y por supuesto, hay que agradecer a la Delegación de la Rectora para la Universidad D
 'Para terminar, os hacemos un guiño más a todos aquellos que no usáis Software Libre,
 'Como ya sabéis, es nuestro personaje humorístico y con él queremos enseñar con un toq
 'Si realmente es necesario seguir engordando a empresas con licencias y licencias que
```

```
In [70]: tokens = nltk.word_tokenize(sentences[0])
         tokens
```

```
Out[70]: ['Como',
          'ya',
          'venimos',
          'avisando',
          'desde',
          'hace',
          'meses',
          ',',
          'este',
          'jueves',
          '28',
          'de',
          'septiembre',
          ',',
          'comienzan',
          'las',
          'Jornadas',
          'de',
          'Software',
          'Libre',
          'de',
          'esta',
          'universidad',
          'que',
          'celebraremos',
          'en',
          'el',
          'Instituto',
          'de',
          'Matemáticas',
          '.']
```

```
In [71]: tags = nltk.pos_tag(tokens)
         tags # Algo va muy mal...
```

```
Out[71]: [('Como', 'NNP'),
          ('ya', 'PRP'),
          ('venimos', 'VBP'),
```

```
 ('avisando', 'JJ'),
 ('desde', 'NN'),
 ('hace', 'NN'),
 ('meses', 'NNS'),
 (',', ','),
 ('este', 'NN'),
 ('jueves', 'NNS'),
 ('28', 'CD'),
 ('de', 'IN'),
 ('septiembre', 'NN'),
 (',', ','),
 ('comienzan', 'NN'),
 ('las', 'NNS'),
 ('Jornadas', 'NNP'),
 ('de', 'FW'),
 ('Software', 'NNP'),
 ('Libre', 'NNP'),
 ('de', 'IN'),
 ('esta', 'FW'),
 ('universidad', 'JJ'),
 ('que', 'NN'),
 ('celebraremos', 'NN'),
 ('en', 'IN'),
 ('el', 'JJ'),
 ('Instituto', 'NNP'),
 ('de', 'FW'),
 ('Matemáticas', 'NNP'),
 ('.', '.')]
```

1.6.5 Probemos un POS tagger para español: https://nlp.stanford.edu/software/spanish-faq.shtml

```
In [72]: from nltk.tag import StanfordPOSTagger

         spanish_postagger = StanfordPOSTagger('models/spanish.tagger', 'stanford-postagger.jar'
         tagged_words = spanish_postagger.tag(tokens)
         tagged_words

Out[72]: [('Como', 'cs'),
          ('ya', 'rg'),
          ('venimos', 'vmip000'),
          ('avisando', 'vmg0000'),
          ('desde', 'sp000'),
          ('hace', 'vmip000'),
          ('meses', 'nc0p000'),
          (',', 'fc'),
          ('este', 'dd0000'),
          ('jueves', 'w'),
```

```
 ('28', 'z0'),
 ('de', 'sp000'),
 ('septiembre', 'w'),
 (',', 'fc'),
 ('comienzan', 'vmip000'),
 ('las', 'da0000'),
 ('Jornadas', 'np00000'),
 ('de', 'sp000'),
 ('Software', 'np00000'),
 ('Libre', 'np00000'),
 ('de', 'sp000'),
 ('esta', 'dd0000'),
 ('universidad', 'nc0s000'),
 ('que', 'pr000000'),
 ('celebraremos', 'vmif000'),
 ('en', 'sp000'),
 ('el', 'da0000'),
 ('Instituto', 'np00000'),
 ('de', 'sp000'),
 ('Matemáticas', 'aq0000'),
 ('.', 'fp')]
```

```
In [73]: def get_stanford_pos(tag):
             if tag.startswith('a'):
                 return wordnet.ADJ
             elif tag.startswith('v'):
                 return wordnet.VERB
             elif tag.startswith('n'):
                 return wordnet.NOUN
             elif tag.startswith('r'):
                 return wordnet.ADV
             else:
                 return wordnet.NOUN

         tagged = [(t[0], get_stanford_pos(t[1])) for t in tagged_words]
         tagged
```

```
Out[73]: [('Como', 'n'),
          ('ya', 'r'),
          ('venimos', 'v'),
          ('avisando', 'v'),
          ('desde', 'n'),
          ('hace', 'v'),
          ('meses', 'n'),
          (',', 'n'),
          ('este', 'n'),
          ('jueves', 'n'),
          ('28', 'n'),
```

```
          ('de', 'n'),
          ('septiembre', 'n'),
          (',', 'n'),
          ('comienzan', 'v'),
          ('las', 'n'),
          ('Jornadas', 'n'),
          ('de', 'n'),
          ('Software', 'n'),
          ('Libre', 'n'),
          ('de', 'n'),
          ('esta', 'n'),
          ('universidad', 'n'),
          ('que', 'n'),
          ('celebraremos', 'v'),
          ('en', 'n'),
          ('el', 'n'),
          ('Instituto', 'n'),
          ('de', 'n'),
          ('Matemáticas', 'a'),
          ('.', 'n')]
```

```
In [74]: result = nltk.ne_chunk(tagged)
         print(result)
```

```
(S
  Como/n
  ya/r
  venimos/v
  avisando/v
  desde/n
  hace/v
  meses/n
  ,/n
  este/n
  jueves/n
  28/n
  de/n
  septiembre/n
  ,/n
  comienzan/v
  las/n
  Jornadas/n
  de/n
  Software/n
  Libre/n
  de/n
  esta/n
  universidad/n
```

```
que/n
celebraremos/v
en/n
el/n
Instituto/n
de/n
Matemáticas/a
./n)
```

1.7 Ahora es tu turno, ponte a prueba...

```
In [75]: tokens[:9]
```

```
Out[75]: ['Como', 'ya', 'venimos', 'avisando', 'desde', 'hace', 'meses', ',', 'este']
```

2 Bibliografia

https://www.nltk.org/book/

3 Licencia