Reading data

In [2]:

```
import pandas as pd
```

In [3]:

```
data = pd.read_csv('./../Formations_NLP/articles_bbc_
2018_01_30.csv')
```

In [4]:

```
data.shape
```

Out[4]:

```
(309, 2)
```

In [5]:

```
data = data.dropna().reset_index(drop=True)
```

In [6]:

```
data.shape
```

Out[6]:

```
(308, 2)
```

Cleaning

Keeping English articles

In [7]:

```
from langdetect import detect
from tqdm import tqdm_notebook
tqdm_notebook().pandas()
```

```
Widget Javascript not detected.  It may not be instal
led properly. Did you enable the widgetsnbextension?
If not, then run "jupyter nbextension enable --py --s
ys-prefix widgetsnbextension"
```

In [8]:

```
data['lang'] = data.articles.progress_map(detect)
```

```
Widget Javascript not detected.  It may not be instal
led properly. Did you enable the widgetsnbextension?
If not, then run "jupyter nbextension enable --py --s
ys-prefix widgetsnbextension"
```

In [9]:

```
data.lang.value_counts()
```

Out[9]:

```
en    257
fa      9
fr      7
id      5
vi      4
hi      4
ar      4
uk      4
ru      4
sw      3
tr      2
es      2
pt      2
de      1
Name: lang, dtype: int64
```

In [10]:

```
data = data.loc[data.lang=='en']
```

Tokenization

In [11]:

```
from nltk.tokenize import sent_tokenize
```

In [12]:

```
data['sentences'] = data.articles.progress_map(sent_t
okenize)
data['sentences'].head(1).tolist()[0][:3] # Print the
first 3 sentences of the 1st article
```

```
Widget Javascript not detected.  It may not be instal
led properly. Did you enable the widgetsnbextension?
If not, then run "jupyter nbextension enable --py --s
ys-prefix widgetsnbextension"
```

Out[12]:

```
['Image copyright PA/EPA Image caption Oligarch Roman
Abramovich (l) and PM Dmitry Medvedev are on the list
\r\n\r\nRussian President Vladimir Putin says a list
of officials and businessmen close to the Kremlin pub
lished by the US has in effect targeted all Russian p
eople.',
 'The list names 210 top Russians as part of a sancti
ons law aimed at punishing Moscow for meddling in the
US election.',
 'However, the US stressed those named were not subje
ct to new sanctions.']
```

In [13]:

```
from nltk.tokenize import word_tokenize
```

In [14]:

```
data['tokens_sentences'] = data['sentences'].progress_map(lambda sentences: [word_tokenize(sentence) for sentence in sentences])
print(data['tokens_sentences'].head(1).tolist()[0][:3])
```

```
Widget Javascript not detected.  It may not be installed properly. Did you enable the widgetsnbextension? If not, then run "jupyter nbextension enable --py --sys-prefix widgetsnbextension"
```

```
[['Image', 'copyright', 'PA/EPA', 'Image', 'caption', 'Oligarch', 'Roman', 'Abramovich', '(', 'l', ')', 'and', 'PM', 'Dmitry', 'Medvedev', 'are', 'on', 'the', 'list', 'Russian', 'President', 'Vladimir', 'Putin', 'says', 'a', 'list', 'of', 'officials', 'and', 'businessmen', 'close', 'to', 'the', 'Kremlin', 'published', 'by', 'the', 'US', 'has', 'in', 'effect', 'targeted', 'all', 'Russian', 'people', '.'], ['The', 'list', 'names', '210', 'top', 'Russians', 'as', 'part', 'of', 'a', 'sanctions', 'law', 'aimed', 'at', 'punishing', 'Moscow', 'for', 'meddling', 'in', 'the', 'US', 'election', '.'], ['However', ',', 'the', 'US', 'stressed', 'those', 'named', 'were', 'not', 'subject', 'to', 'new', 'sanctions', '.']]
```

Lemmatizing with POS tagging

In [15]:

```
from nltk import pos_tag
```

In [16]:

```
data['POS_tokens'] = data['tokens_sentences'].progress_map(lambda tokens_sentences: [pos_tag(tokens) for tokens in tokens_sentences])
print(data['POS_tokens'].head(1).tolist()[0][:3])
```

```
Widget Javascript not detected.  It may not be installed properly. Did you enable the widgetsnbextension? If not, then run "jupyter nbextension enable --py --sys-prefix widgetsnbextension"
```

```
[[('Image', 'NN'), ('copyright', 'NN'), ('PA/EPA', 'NNP'), ('Image', 'NNP'), ('caption', 'NN'), ('Oligarch', 'NNP'), ('Roman', 'NNP'), ('Abramovich', 'NNP'), ('(', '('), ('l', 'NN'), (')', ')'), ('and', 'CC'), ('PM', 'NNP'), ('Dmitry', 'NNP'), ('Medvedev', 'NNP'), ('are', 'VBP'), ('on', 'IN'), ('the', 'DT'), ('list', 'NN'), ('Russian', 'NNP'), ('President', 'NNP'), ('Vladimir', 'NNP'), ('Putin', 'NNP'), ('says', 'VBZ'), ('a', 'DT'), ('list', 'NN'), ('of', 'IN'),
```

```
('officials', 'NNS'), ('and', 'CC'), ('businessmen',
'NNS'), ('close', 'RB'), ('to', 'TO'), ('the', 'DT'),
('Kremlin', 'NNP'), ('published', 'VBN'), ('by', 'I
N'), ('the', 'DT'), ('US', 'NNP'), ('has', 'VBZ'),
('in', 'IN'), ('effect', 'NN'), ('targeted', 'VBN'),
('all', 'DT'), ('Russian', 'JJ'), ('people', 'NNS'),
('.', '.')], [('The', 'DT'), ('list', 'NN'), ('name
s', 'RB'), ('210', 'CD'), ('top', 'JJ'), ('Russians',
'NNPS'), ('as', 'IN'), ('part', 'NN'), ('of', 'IN'),
('a', 'DT'), ('sanctions', 'NNS'), ('law', 'NN'), ('a
imed', 'VBN'), ('at', 'IN'), ('punishing', 'VBG'),
('Moscow', 'NNP'), ('for', 'IN'), ('meddling', 'VB
G'), ('in', 'IN'), ('the', 'DT'), ('US', 'NNP'), ('el
ection', 'NN'), ('.', '.')], [('However', 'RB'),
(',', ','), ('the', 'DT'), ('US', 'NNP'), ('stresse
d', 'VBD'), ('those', 'DT'), ('named', 'VBN'), ('wer
e', 'VBD'), ('not', 'RB'), ('subject', 'JJ'), ('to',
'TO'), ('new', 'JJ'), ('sanctions', 'NNS'), ('.',
'.')]]
```

In [17]:

```
# Inspired from https://stackoverflow.com/a/15590384
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 ''

from nltk.stem.wordnet import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

In [18]:

```
# Lemmatizing each word with its POS tag, in each sen
tence
data['tokens_sentences_lemmatized'] = data['POS_token
s'].progress_map(
    lambda list_tokens_POS: [
        [
            lemmatizer.lemmatize(el[0], get_wordnet_p
os(el[1]))
            if get_wordnet_pos(el[1]) != '' else el[0
] for el in tokens_POS
        ]
        for tokens_POS in list_tokens_POS
    ]
)
```

```
Widget Javascript not detected.  It may not be instal
led properly. Did you enable the widgetsnbextension?
If not, then run "jupyter nbextension enable --py --s
ys-prefix widgetsnbextension"
```

In [19]:

```
data['tokens_sentences_lemmatized'].head(1).tolist()[
0][:3]
```

Out[19]:

```
[['Image',
  'copyright',
  'PA/EPA',
  'Image',
  'caption',
  'Oligarch',
  'Roman',
  'Abramovich',
  '(',
  'l',
  ')',
  'and',
  'PM',
  'Dmitry',
  'Medvedev',
  'be',
  'on',
  'the',
  'list',
  'Russian',
  'President',
  'Vladimir',
  'Putin',
  'say',
  'a',
  'list',
  'of',
  'official',
  'and',
  'businessmen',
  'close',
  'to',
  'the',
  'Kremlin',
  'publish',
  'by',
  'the',
  'US',
  'have',
  'in',
  'effect',
  'target',
  'all',
  'Russian',
  'people',
  '.'],
```

```
 ['The',
  'list',
  'names',
  '210',
  'top',
  'Russians',
  'as',
  'part',
  'of',
  'a',
  'sanction',
  'law',
  'aim',
  'at',
  'punish',
  'Moscow',
  'for',
  'meddle',
  'in',
  'the',
  'US',
  'election',
  '.'],
 ['However',
  ',',
  'the',
  'US',
  'stress',
  'those',
  'name',
  'be',
  'not',
  'subject',
  'to',
  'new',
  'sanction',
  '.']]
```

Regrouping tokens and removing stop words

In [20]:

```
from nltk.corpus import stopwords
stopwords_verbs = ['say', 'get', 'go', 'know', 'may',
'need', 'like', 'make', 'see', 'want', 'come', 'take'
, 'use', 'would', 'can']
stopwords_other = ['one', 'mr', 'bbc', 'image', 'gett
y', 'de', 'en', 'caption', 'also', 'copyright', 'some
thing']
my_stopwords = stopwords.words('English') + stopwords
_verbs + stopwords_other
```

In [21]:

```
from itertools import chain # to flatten list of sent
ences of tokens into list of tokens
```

In [22]:

```
data['tokens'] = data['tokens_sentences_lemmatized'].
map(lambda sentences: list(chain.from_iterable(senten
ces)))
data['tokens'] = data['tokens'].map(lambda tokens: [t
oken.lower() for token in tokens if token.isalpha()
                                                    a
nd token.lower() not in my_stopwords and len(token)>1
])
```

In [23]:

```
data['tokens'].head(1).tolist()[0][:30]
```

Out[23]:

```
['oligarch',
 'roman',
 'abramovich',
 'pm',
 'dmitry',
 'medvedev',
 'list',
 'russian',
 'president',
 'vladimir',
 'putin',
 'list',
 'official',
 'businessmen',
 'close',
 'kremlin',
 'publish',
 'us',
 'effect',
 'target',
 'russian',
 'people',
 'list',
 'names',
 'top',
 'russians',
 'part',
 'sanction',
 'law',
 'aim']
```

LDA

Data preparation

Prepare bi-grams and tri-grams

In [24]:

```
from gensim.models import Phrases
```

```
C:\ProgramData\Anaconda3\lib\site-packages\gensim\uti
ls.py:865: UserWarning: detected Windows; aliasing ch
unkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize
to chunkize_serial")
Using TensorFlow backend.
```

In [25]:

```
tokens = data['tokens'].tolist()
bigram_model = Phrases(tokens)
trigram_model = Phrases(bigram_model[tokens], min_cou
nt=1)
tokens = list(trigram_model[bigram_model[tokens]])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\gensim\mod
els\phrases.py:316: UserWarning: For a faster impleme
ntation, use the gensim.models.phrases.Phraser class
  warnings.warn("For a faster implementation, use the
gensim.models.phrases.Phraser class")
```

Prepare objects for LDA gensim implementation

In [26]:

```
from gensim import corpora
```

In [27]:

```
dictionary_LDA = corpora.Dictionary(tokens)
dictionary_LDA.filter_extremes(no_below=3)
corpus = [dictionary_LDA.doc2bow(tok) for tok in toke
ns]
```

Running LDA

In [28]:

```
from gensim import models
import numpy as np
```

In [29]:

```
np.random.seed(123456)
num_topics = 20
%time lda_model = models.LdaModel(corpus, num_topics=
num_topics, \
                                  id2word=dictionary_
LDA, \
                                  passes=4, alpha=[0.
01]*num_topics, \
                                  eta=[0.01]*len(dict
ionary_LDA.keys()))
```

```
Wall time: 7.75 s
```

Quick exploration of LDA results

Looking at topics

In [271]:

```
for i,topic in lda_model.show_topics(formatted=True,
num_topics=num_topics, num_words=20):
    print(str(i)+": "+ topic)
    print()
```

```
0: 0.024*"base" + 0.018*"data" + 0.015*"security" +
0.015*"show" + 0.015*"plan" + 0.011*"part" + 0.010*"a
ctivity" + 0.010*"road" + 0.008*"afghanistan" + 0.008
*"track" + 0.007*"former" + 0.007*"add" + 0.007*"arou
nd_world" + 0.007*"university" + 0.007*"building" +
0.006*"mobile_phone" + 0.006*"point" + 0.006*"new" +
0.006*"exercise" + 0.006*"open"

1: 0.014*"woman" + 0.010*"child" + 0.010*"tunnel" +
0.007*"law" + 0.007*"customer" + 0.007*"continue" +
0.006*"india" + 0.006*"hospital" + 0.006*"live" + 0.0
06*"public" + 0.006*"video" + 0.005*"couple" + 0.005
*"place" + 0.005*"people" + 0.005*"another" + 0.005
*"case" + 0.005*"government" + 0.005*"health" + 0.005
*"part" + 0.005*"underground"

2: 0.011*"government" + 0.008*"become" + 0.008*"call"
+ 0.007*"report" + 0.007*"northern_mali" + 0.007*"gro
up" + 0.007*"ansar_dine" + 0.007*"tuareg" + 0.007*"co
uld" + 0.007*"us" + 0.006*"journalist" + 0.006*"reall
y" + 0.006*"story" + 0.006*"post" + 0.006*"islamist"
+ 0.005*"data" + 0.005*"news" + 0.005*"new" + 0.005
*"local" + 0.005*"part"

3: 0.011*"people" + 0.008*"work" + 0.008*"benefit" +
0.007*"healthcare" + 0.007*"tell" + 0.006*"israel" +
0.006*"dish" + 0.006*"us" + 0.006*"find" + 0.006*"rad
io" + 0.006*"sanction" + 0.006*"david" + 0.006*"presi
dent" + 0.006*"amazon" + 0.005*"final" + 0.005*"play"
+ 0.004*"ask" + 0.004*"try" + 0.004*"images" + 0.004
*"food"

4: 0.018*"us" + 0.007*"company" + 0.007*"first" + 0.0
07*"think" + 0.007*"people" + 0.006*"light" + 0.006
*"uk" + 0.005*"back" + 0.005*"work" + 0.005*"much" +
0.005*"help" + 0.005*"thing" + 0.005*"time" + 0.005
*"year" + 0.005*"day" + 0.004*"product" + 0.004*"coul
d" + 0.004*"new" + 0.004*"long" + 0.004*"tell"

5: 0.019*"dutch" + 0.010*"border" + 0.010*"idea" + 0.
009*"write" + 0.006*"without" + 0.006*"machine" + 0.0
05*"building" + 0.005*"system" + 0.005*"report" + 0.0
05*"audience" + 0.005*"writer" + 0.005*"note" + 0.005
*"data" + 0.005*"trust" + 0.005*"way" + 0.005*"nether
```

```
"data" + 0.005*"trust" + 0.005*"way" + 0.005*"nether
lands" + 0.005*"human" + 0.005*"piece" + 0.005*"peopl
e" + 0.005*"govern"

6: 0.012*"brexit" + 0.009*"uk" + 0.009*"brexiteers" +
0.008*"eu" + 0.007*"could" + 0.007*"part" + 0.007*"ch
ange" + 0.006*"time" + 0.006*"theresa" + 0.006*"polit
ical" + 0.005*"club" + 0.005*"think" + 0.005*"radio"
+ 0.005*"government" + 0.005*"celebration" + 0.005*"m
inister" + 0.005*"party" + 0.005*"transition" + 0.005
*"england" + 0.004*"work"

7: 0.013*"find" + 0.012*"transparent" + 0.012*"light"
+ 0.011*"live" + 0.011*"predator" + 0.007*"kick" + 0.
006*"small" + 0.006*"look" + 0.006*"eye" + 0.006*"ani
mal" + 0.006*"specie" + 0.006*"hide" + 0.006*"call" +
0.006*"creature" + 0.006*"study" + 0.006*"act" + 0.00
6*"however" + 0.005*"could" + 0.005*"fast" + 0.005*"t
ime"

8: 0.010*"year" + 0.010*"include" + 0.010*"flight" +
0.008*"work" + 0.007*"day" + 0.007*"film" + 0.006*"pe
ople" + 0.006*"award" + 0.005*"give" + 0.005*"best" +
0.005*"prize" + 0.005*"actor" + 0.005*"travel" + 0.00
5*"set" + 0.005*"show" + 0.005*"holiday" + 0.005*"lau
nch" + 0.004*"team" + 0.004*"king" + 0.004*"country"

9: 0.046*"news" + 0.029*"link" + 0.028*"output" + 0.0
19*"report" + 0.017*"section" + 0.017*"story" + 0.013
*"value" + 0.013*"data" + 0.013*"website" + 0.012*"fo
llowing" + 0.012*"week" + 0.011*"provide" + 0.011*"re
levant" + 0.011*"journalism" + 0.010*"close" + 0.010
*"content" + 0.010*"audience" + 0.009*"issue" + 0.009
*"standard" + 0.009*"uk"

10: 0.008*"people" + 0.007*"president" + 0.006*"commu
nity" + 0.005*"country" + 0.005*"call" + 0.005*"area"
+ 0.005*"tell" + 0.005*"world" + 0.004*"animal" + 0.0
04*"work" + 0.004*"could" + 0.004*"service" + 0.004
*"organisation" + 0.004*"many" + 0.004*"mean" + 0.004
*"risk" + 0.004*"day" + 0.004*"help" + 0.004*"trump"
+ 0.004*"local"

11: 0.010*"game" + 0.009*"design" + 0.009*"wave" + 0.
009*"people" + 0.008*"stress" + 0.008*"time" + 0.008
*"sleep" + 0.008*"drug" + 0.007*"ocean" + 0.007*"fin
d" + 0.006*"test" + 0.006*"work" + 0.006*"fin" + 0.00
6*"create" + 0.006*"could" + 0.006*"water" + 0.006*"g
ive" + 0.006*"help" + 0.006*"run" + 0.005*"energy"

12: 0.015*"show" + 0.011*"question" + 0.008*"student"
+ 0.008*"people" + 0.007*"think" + 0.006*"man" + 0.00
6*"part" + 0.006*"university" + 0.006*"tell" + 0.006
*"win" + 0.006*"school" + 0.005*"series" + 0.005*"giv
e" + 0.005*"hear" + 0.005*"whole" + 0.005*"even" + 0.
005*"person" + 0.005*"time" + 0.005*"kill" + 0.005*"c
aptain"
```

```
13: 0.023*"eu" + 0.010*"girl" + 0.009*"boy" + 0.009
*"call" + 0.008*"india" + 0.008*"state" + 0.008*"euro
pe" + 0.007*"die" + 0.007*"daughter" + 0.007*"life" +
0.007*"think" + 0.006*"many" + 0.006*"find" + 0.006
*"include" + 0.006*"never" + 0.006*"woman" + 0.006*"l
ead" + 0.006*"follow" + 0.006*"interview" + 0.005*"sh
are"

14: 0.028*"list" + 0.016*"name" + 0.013*"us" + 0.013
*"russia" + 0.009*"putin" + 0.008*"russian" + 0.008
*"sanction" + 0.008*"election" + 0.007*"new_sanction"
+ 0.007*"day" + 0.006*"report" + 0.006*"good" + 0.006
*"president_donald_trump" + 0.006*"gun" + 0.006*"top"
+ 0.006*"add" + 0.006*"shy" + 0.006*"even" + 0.006*"l
ater" + 0.006*"oligarch"

15: 0.009*"village" + 0.009*"people" + 0.008*"find" +
0.008*"case" + 0.006*"good" + 0.006*"many" + 0.006*"t
ell" + 0.006*"police" + 0.005*"judge" + 0.005*"benefi
t" + 0.005*"city" + 0.005*"add" + 0.005*"house" + 0.0
05*"thomas" + 0.005*"worth" + 0.005*"treatment" + 0.0
05*"brain" + 0.004*"late" + 0.004*"look" + 0.004*"str
ong"

16: 0.007*"study" + 0.007*"report" + 0.006*"time" +
0.006*"news" + 0.006*"human" + 0.004*"us" + 0.004*"te
st" + 0.004*"follow" + 0.004*"water" + 0.004*"brexit"
+ 0.004*"add" + 0.004*"year" + 0.004*"place" + 0.004
*"deal" + 0.004*"tell" + 0.004*"work" + 0.004*"find"
+ 0.004*"experiment" + 0.004*"publish" + 0.003*"docum
ent"

17: 0.010*"prey" + 0.008*"people" + 0.008*"work" + 0.
008*"call" + 0.006*"specie" + 0.006*"find" + 0.006*"n
ight" + 0.006*"beetle" + 0.005*"could" + 0.005*"image
s" + 0.005*"move" + 0.004*"country" + 0.004*"instagra
m" + 0.004*"include" + 0.004*"spanish" + 0.004*"even"
+ 0.004*"day" + 0.003*"strange" + 0.003*"insect" + 0.
003*"seem"

18: 0.007*"first" + 0.006*"find" + 0.005*"year" + 0.0
05*"back" + 0.005*"time" + 0.005*"us" + 0.004*"peopl
e" + 0.004*"work" + 0.004*"city" + 0.004*"even" + 0.0
04*"could" + 0.004*"history" + 0.004*"two" + 0.003*"e
nd" + 0.003*"leave" + 0.003*"think" + 0.003*"family"
+ 0.003*"return" + 0.003*"show" + 0.003*"much"

19: 0.014*"video" + 0.014*"artist" + 0.011*"award" +
0.010*"back" + 0.010*"ask" + 0.010*"pink" + 0.009*"ac
ademy" + 0.009*"grammys" + 0.009*"female" + 0.007*"ye
ar" + 0.007*"woman" + 0.006*"producer" + 0.006*"much"
+ 0.006*"images" + 0.006*"time" + 0.006*"couple" + 0.
006*"wish" + 0.006*"remember" + 0.006*"criticism" +
0.006*"help"
```

Allocating topics to documents

In [171]:

```
print(data.articles.loc[0][:500])
```

```
Image copyright PA/EPA Image caption Oligarch Roman A
bramovich (l) and PM Dmitry Medvedev are on the list

Russian President Vladimir Putin says a list of offic
ials and businessmen close to the Kremlin published b
y the US has in effect targeted all Russian people.

The list names 210 top Russians as part of a sanction
s law aimed at punishing Moscow for meddling in the U
S election.

However, the US stressed those named were not subject
to new sanctions.

Mr Putin said the list was an unfr
```

In [172]:

```
lda_model[corpus[0]]
```

Out[172]:

```
[(14, 0.9983065953654187)]
```

Predicting topics on unseen documents

In [44]:

```
document = '''Eric Tucker, a 35-year-old co-founder o
f a marketing company in Austin, Tex., had just about
40 Twitter followers. But his recent tweet about paid
protesters being bused to demonstrations against Pres
ident-elect Donald J. Trump fueled a nationwide consp
iracy theory — one that Mr. Trump joined in promotin
g.
```