topics-csv-gen

April 24, 2019

1 Topic-to-CSV-Generator

In this Jupyter-Notebook the reviews were transformed into topics. There are intermediate steps that should save the results in between as json-files. Currently, they are commented out, but they can be activated if needed.

Content: - Importing packages - Read the data - 0 Split Sentences - 1 Tokenize Words in Sentences - 2 Remove Punctuation - 3.1 Train POS-Tagging - 3.2 Save POS-Tagging - 3.3 Load POS-Tagging - 3.4 Apply POS-Tagging - 4 Lemmatize based on POS-Tagging - 5 Lower Lemmas - 6 Remove Stopwords from Lemmas (and some others) - 7 Compress Dataset - 8 Saving to CSV

1.1 Importing necessary packages

In [1]:
```
import pandas as pd
import json
import nltk
import random
#import numpy as np
#nltk.download("punkt")
import string
exclude = set(string.punctuation)
import time
from tqdm import tqdm
import re
import pickle
```

In [2]:
```
from nltk.tag.sequential import ClassifierBasedTagger

### from https://github.com/ptnplanet/NLTK-Contributions/tree/master/ClassifierBasedGe

class ClassifierBasedGermanTagger(ClassifierBasedTagger):
    """A classifier based German part-of-speech tagger. It has an accuracy of
    96.09% after being trained on 90% of the German TIGER corpus. The tagger
    extends the NLTK ClassifierBasedTagger and implements a slightly modified
    feature detector.
    """

    def feature_detector(self, tokens, index, history):
```

```
    """Implementing a slightly modified feature detector.
    @param tokens: The tokens from the sentence to tag.
    @param index: The current token index to tag.
    @param history: The previous tagged tokens.
    """

    word = tokens[index]
    if index == 0: # At the beginning of the sentence
        prevword = prevprevword = None
        prevtag = prevprevtag = None
        #word = word.lower() # Lowercase at the beginning of sentence
    elif index == 1:
        prevword = tokens[index-1] # Note: no lowercase
        prevprevword = None
        prevtag = history[index-1]
        prevprevtag = None
    else:
        prevword = tokens[index-1]
        prevprevword = tokens[index-2]
        prevtag = history[index-1]
        prevprevtag = history[index-2]

    if re.match('[0-9]+([\.,][0-9]*)?|[0-9]*[\.,][0-9]+$', word):
        # Included "," as decimal point
        shape = 'number'
    elif re.compile('\W+$', re.UNICODE).match(word):
        # Included unicode flag
        shape = 'punct'
    elif re.match('([A-ZÄÖÜ]+[a-zäöü]*-?)+$', word):
        # Included dash for dashed words and umlauts
        shape = 'upcase'
    elif re.match('[a-zäöü]+', word):
        # Included umlauts
        shape = 'downcase'
    elif re.compile("\w+", re.UNICODE).match(word):
        # Included unicode flag
        shape = 'mixedcase'
    else:
        shape = 'other'

    features = {
        'prevtag': prevtag,
        'prevprevtag': prevprevtag,
        'word': word,
        'word.lower': word.lower(),
        'suffix3': word.lower()[-3:],
        #'suffix2': word.lower()[-2:],
        #'suffix1': word.lower()[-1:],
```

```
                    'preffix1': word[:1], # included
                    'prevprevword': prevprevword,
                    'prevword': prevword,
                    'prevtag+word': '%s+%s' % (prevtag, word),
                    'prevprevtag+word': '%s+%s' % (prevprevtag, word),
                    'prevword+word': '%s+%s' % (prevword, word),
                    'shape': shape
                    }
                return features
```

1.2 Read the Data

```
In [2]: data = pd.read_csv('Datensatz_Coding_Challenge.csv', delimiter=";")
        corpus = data.copy()
        corpus.head()
```

```
Out[2]:    StyleID                                               text  rating
        0  1709054             Die sind okay und dann für den Preis.        5
        1  1709054  Qualität und Preis sind gut. Leider sind sie z...       3
        2  8623725                             lässt schlanker aussehen     5
        3  8623725  Material und Farbe gut. Da einige Kundinnen in...       3
        4  9743730  Material ist schön zum verdunkel.  Leider doch...       5
```

1.3 0 Split Sentences

```
In [3]: for i in tqdm(range(len(corpus["text"]))):
            corpus["text"][i] = nltk.sent_tokenize(corpus["text"][i])
```

```
  0%|                                                                              |
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html

100%|| 19150/19150 [34:06<00:00,  9.08it/s]
```

```
In [4]: #corpus.to_json(r'00sentences.json')
```

```
In [5]: corpus.head()
```

```
Out[5]:    StyleID                                               text  rating
        0  1709054            [Die sind okay und dann für den Preis.]       5
        1  1709054  [Qualität und Preis sind gut., Leider sind sie...       3
        2  8623725                           [lässt schlanker aussehen]     5
        3  8623725  [Material und Farbe gut., Da einige Kundinnen ...       3
        4  9743730  [Material ist schön zum verdunkel., Leider doc...       5
```

1.4 1 Tokenize Words in Sentences

```
In [14]: for i in tqdm(range(len(corpus["text"]))):
             for j in range(len(corpus["text"][i])):
                 corpus["text"][i][j] = nltk.word_tokenize(corpus["text"][i][j])
```

```
100%|| 19150/19150 [00:32<00:00, 586.09it/s]
```

```
In [16]: #corpus.to_json(r'01tokenized.json')
```

```
In [17]: corpus.head()
```

```
Out[17]:    StyleID                                               text  rating
         0  1709054  [[Die, sind, okay, und, dann, für, den, Preis,...       5
         1  1709054  [[Qualität, und, Preis, sind, gut, .], [Leider...       3
         2  8623725                      [[lässt, schlanker, aussehen]]       5
         3  8623725  [[Material, und, Farbe, gut, .], [Da, einige, ...       3
         4  9743730  [[Material, ist, schön, zum, verdunkel, .], [L...       5
```

1.5 2 Remove Punctuation

```
In [27]: for i in tqdm(range(len(corpus["text"]))):
             for j in range(len(corpus["text"][i])):
                 corpus["text"][i][j] = [token for token in corpus["text"][i][j] if token not
                                         exclude and token.isalpha()]
```

```
100%|| 19150/19150 [00:04<00:00, 3929.68it/s]
```

```
In [29]: #corpus.to_json(r'02punctuation.json')
```

```
In [28]: corpus.head()
```

```
Out[28]:       StyleID  rating                                               text
         0     1709054       5    [[Die, sind, okay, und, dann, für, den, Preis]]
         1     1709054       3  [[Qualität, und, Preis, sind, gut], [Leider, s...
         10     709229       5  [[Angenehmer, Stoff], [Füllt, sich, gut, auf, ...
         100    709229       5  [[Angenehmes, und, pflegeleichtes, Material], ...
         1000  9743730       1  [[ACHTUNG, Die, Stofffarbe, creme, ist, nicht,...
```

1.6 3.1 Train POS-Tagging

This part of code is needed to create "nltk_german_classifier_data.pickle". As this was done already, this code is not necessary for the whole code to run.

```
In [8]: #corp = nltk.corpus.ConllCorpusReader('.', 'tiger_release_aug07.corrected.16012013.con
        #                                     ['ignore', 'words', 'ignore', 'ignore', 'pos'],
        #                                     encoding='utf-8')
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-8-48c495b46839> in <module>
----> 1 corp = nltk.corpus.ConllCorpusReader('.', 'tiger_release_aug07.corrected.16012013.
      2                                      ['ignore', 'words', 'ignore', 'ignore', 'pos']
      3                                      encoding='utf-8')


NameError: name 'nltk' is not defined
```

```
In [15]: #tagged_sents = corp.tagged_sents()

In [17]: #tagged_sents2 = [sentence for sentence in tagged_sents]

In [18]: #type(tagged_sents2)

Out[18]: list

In [19]: #random.shuffle(tagged_sents2)

In [20]: #split_perc = 0.1
         #split_size = int(len(tagged_sents) * split_perc)
         #train_sents, test_sents = tagged_sents2[split_size:], tagged_sents2[:split_size]

In [23]: #tagger = ClassifierBasedGermanTagger(train=train_sents)

In [24]: #accuracy = tagger.evaluate(test_sents)

In [25]: #print(accuracy)

0.9418093958519154

In [26]: #tagger.tag(['Ich', 'schreibe', 'gerade', 'an', 'meiner', 'Thesis'])

Out[26]: [('Ich', 'PPER'),
          ('schreibe', 'VVFIN'),
          ('gerade', 'ADV'),
          ('an', 'APPR'),
          ('meiner', 'PPOSAT'),
          ('Thesis', 'NN')]
```

1.7 3.2 Save POS-Tagging

```
In [30]: ### SAVING
         #with open("nltk_german_classifier_data.pickle", "wb") as f:
         #    pickle.dump(tagger, f, protocol=2)
```

1.8 3.3 Load POS-Tagging

```
In [3]: ### LOADING
        with open('nltk_german_classifier_data.pickle', 'rb') as f:
            tagger = pickle.load(f)
```

1.9 3.4 Apply POS-Tagging

```
In [33]: for i in tqdm(range(len(corpus["text"]))):
             for j in range(len(corpus["text"][i])):
                 corpus["text"][i][j] = tagger.tag(corpus["text"][i][j])
```

```
100%|| 19150/19150 [22:55<00:00, 13.92it/s]
```

```
In [35]: #corpus.to_json(r'03pos-tagged.json')
```

```
In [60]: corpus.head()
```

```
Out[60]:       StyleID  rating                                               text  \
         0     1709054       5  [[(Die, ART), (sind, VAFIN), (okay, ADJD), (un...
         1     1709054       3  [[(Qualität, NN), (und, KON), (Preis, NN), (si...
         10     709229       5  [[(Angenehmer, NE), (Stoff, NN)], [(Füllt, NE)...
         100    709229       5  [[(Angenehmes, NE), (und, KON), (pflegeleichte...
         1000  9743730       1  [[(ACHTUNG, NN), (Die, ART), (Stofffarbe, NN),...

              lemmas
         0        []
         1        []
         10       []
         100      []
         1000     []
```

1.10 4 Lemmatize based on POS-Tagging

```
In [36]: from germalemma import GermaLemma
         lemmatizer = GermaLemma()
```

```
In [59]: corpus["lemmas"] = [[] for g in range(len(corpus))]
```

```
In [63]: for i in tqdm(range(len(corpus["text"]))):
             for j in range(len(corpus["text"][i])):
                 for k in range(len(corpus["text"][i][j])):
                     try:
                         corpus["lemmas"][i].append(lemmatizer.find_lemma(corpus["text"][i][j]
                     except ValueError:
                         pass
```

```
100%|| 19150/19150 [01:10<00:00, 278.47it/s]
```

```
In [65]: #corpus.to_json(r'04pos-with-lemmas.json')

In [3]: corpus.head()

Out[3]:       StyleID                                             lemmas  rating  \
        0     1709054                          [sein, okay, dann, Preis]       5
        1     1709054  [Qualität, Preis, sein, gut, leider, sein, gro...       3
        10     709229             [Angenehmer, Stoff, Füllt, gut, Haut]       5
        100    709229  [Angenehmes, pflegeleicht, Material, Stoff, kn...       5
        1000  9743730  [ACHTUNG, Stofffarbe, crem, sein, Foto, abbgeb...       1

                                                      text
        0     [[[Die, ART], [sind, VAFIN], [okay, ADJD], [un...
        1     [[[Qualität, NN], [und, KON], [Preis, NN], [si...
        10    [[[Angenehmer, NE], [Stoff, NN]], [[Füllt, NE]...
        100   [[[Angenehmes, NE], [und, KON], [pflegeleichte...
        1000  [[[ACHTUNG, NN], [Die, ART], [Stofffarbe, NN],...
```

1.11 5 Lower Lemmas

```
In [4]: for i in tqdm(range(len(corpus))):
            corpus["lemmas"][i] = [word.lower() for word in corpus["lemmas"][i]]
            #corpus["lemmas"][i] = [word for word in corpus["lemmas"][i] if word not in stop]

  0%|                                                                                          |
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html

100%|| 19150/19150 [1:11:40<00:00,  4.45it/s]


In [6]: #corpus.to_json(r'05lower-pos-with-lemmas.json')

In [5]: corpus.head()

Out[5]:       StyleID                                             lemmas  rating  \
        0     1709054                          [sein, okay, dann, preis]       5
        1     1709054  [qualität, preis, sein, gut, leider, sein, gro...       3
        10     709229             [angenehmer, stoff, füllt, gut, haut]       5
        100    709229  [angenehmes, pflegeleicht, material, stoff, kn...       5
        1000  9743730  [achtung, stofffarbe, crem, sein, foto, abbgeb...       1

                                                      text
        0     [[[Die, ART], [sind, VAFIN], [okay, ADJD], [un...
        1     [[[Qualität, NN], [und, KON], [Preis, NN], [si...
        10    [[[Angenehmer, NE], [Stoff, NN]], [[Füllt, NE]...
        100   [[[Angenehmes, NE], [und, KON], [pflegeleichte...
        1000  [[[ACHTUNG, NN], [Die, ART], [Stofffarbe, NN],...
```

1.12 6 Remove Stopwords from Lemmas (and some others??)

```
In [19]: from nltk.corpus import stopwords
         stop = stopwords.words('german')

In [28]: remove = [line.rstrip('\n') for line in open('add-stopwords.txt', encoding="utf-8")]

In [30]: exclude = remove + stop

In [32]: for g in tqdm(range(len(corpus))):
             corpus["lemmas"][g] = [word for word in corpus["lemmas"][g] if word not in exclude

  0%|                                                                                          |
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html

100%|| 19150/19150 [2:49:26<00:00,  2.87it/s]


In [34]: #corpus.to_json(r'06lower-pos-with-lemmas-without-stopwords.json')

In [33]: corpus.head()

Out[33]:       StyleID                                            lemmas  rating  \
         0     1709054                                  [okay, preis]       5
         1     1709054  [qualität, preis, gro, gröe, fallen, untersc...       3
         10     709229                  [angenehmer, stoff, füllt, haut]       5
         100    709229  [angenehmes, pflegeleicht, material, stoff, kn...       5
         1000  9743730  [achtung, stofffarbe, crem, foto, abbgebilden,...       1

                                                    text
         0     [[[Die, ART], [sind, VAFIN], [okay, ADJD], [un...
         1     [[[Qualität, NN], [und, KON], [Preis, NN], [si...
         10    [[[Angenehmer, NE], [Stoff, NN]], [[Füllt, NE]...
         100   [[[Angenehmes, NE], [und, KON], [pflegeleichte...
         1000  [[[ACHTUNG, NN], [Die, ART], [Stofffarbe, NN],...
```

1.13 7 Compress Dataset

```
In [48]: new = corpus.lemmas.apply(pd.Series).merge(corpus, left_index = True, right_index = Tr

In [72]: topics = new.drop(["lemmas"], axis=1)
         topics = topics.drop(["text"], axis=1)
         topics = topics.drop(["rating"], axis=1)

In [73]: topics['id'] = topics.index

In [74]: topics = pd.melt(topics, id_vars = ['StyleID', 'id'], value_name = "topic")
```

```
In [76]: topics = topics.drop(["variable"], axis=1)
         topics = topics.dropna()

In [78]: topics[topics["id"]=="1"]
```

Out[78]:

	StyleID	id	topic
1	1709054	1	qualität
19151	1709054	1	preis
38301	1709054	1	gro
57451	1709054	1	gröe
76601	1709054	1	fallen
95751	1709054	1	unterschiedlich
114901	1709054	1	shirt
134051	1709054	1	passen
153201	1709054	1	schade

```
In [79]: topics = topics[["topic", "StyleID", "id"]]

In [14]: topics['topic_safe'] = topics['topic']

In [16]: topics = topics.replace({'topic_safe': {'ä': 'ae', 'ü': 'ue', 'ö': 'oe', '': 'ss'}}, r

In [17]: topics.head()
```

Out[17]:

	topic	StyleID	id	topic_safe
0	okay	1709054	0	okay
1	qualität	1709054	1	qualitaet
2	angenehmer	709229	10	angenehmer
3	angenehmes	709229	100	angenehmes
4	achtung	9743730	1000	achtung

1.14 8 Saving to CSV

```
In [18]: topics.to_csv("topics.csv", sep=";")
```