09_nucle_clean

March 21, 2017

2/23/17 - smiel

1 Cleaning the NUCLE essay data.

```
In [1]: %matplotlib inline
        # run me when first starting this notebook
        import os

        import numpy as np
        import pandas as pd

        path = '/research/ella/rivendell/nucle'
```

```
In [5]: # create the essays data frame
        from bs4 import BeautifulSoup
        with open(os.path.join(path, 'release3.2', 'data', 'nucle3.2.sgml'), 'r') as fin:
            soup = BeautifulSoup(fin, 'lxml')

        docs = soup.find_all('doc')
        print('{} documents'.format(len(list(docs))))

        recs = []

        for doc in docs:
            rec = {'essay_id': doc['nid'], 'L1': 'CHN', 'dataset': 'NUCLE'}
            body = doc.find('text')
            paragraphs = body.find_all('p')
            text = ''.join([p.text for p in paragraphs]).strip()
            annotations = doc.annotation
            error_count = len(annotations.find_all('mistake'))
            n_errors = float(error_count) / len(text)

            rec['text'] = text
            rec['N_errors'] = n_errors
            recs.append(rec)

        df = pd.DataFrame.from_records(recs)
        df.to_csv(os.path.join(path, 'all_essays.csv'), encoding='utf8', index=False)
```

```
1397 documents
```

Now let's start building the sentences data frame. For unidecode to work properly, the following should print "True":

```
In [2]: import sys
        print(sys.maxunicode > 0xffff)
```

```
True
```

The essays don't have IDs or ages, but we might know grade level. Let's add an ID and put a placeholder for grade. In addition, we do have two kinds of language scores which we'll compute here. They are also all to the same prompt.

```
In [6]: from utilitybelt.text import get_sentences
        import copy
        from unidecode import unidecode
        import numpy as np

        # load data
        df_in = pd.read_csv(os.path.join(path, 'all_essays.csv'), encoding='utf8')

        # convert text to ascii
        print('Converting to ASCII')
        df_in['ascii_text'] = df_in.text.apply(lambda t: unidecode(t))

        # normalize line endings
        df_in.ascii_text = df_in.ascii_text.str.replace('\r\n', '\n')
        df_in.ascii_text = df_in.ascii_text.str.replace('\r', '\n')

        # use space instead of tab
        df_in.ascii_text = df_in.ascii_text.str.replace('\t', ' ')

        # now remove any non-printable ascii char
        df_in.ascii_text = df_in.ascii_text.str.replace(r'[^ -~\n]', '')

        # # make sure all is printable
        # for i, t in enumerate(df_in.ascii_text.values):
        #     for ci, c in enumerate(t):
        #         if (32 <= ord(c) <= 126) or c in '\n\t':
        #             continue
        #         else:
        #             print u"Unprintable character {} in {} at char {}:\n\n{}\n================
        #                 ord(c), i, ci, t, df_in.iloc[i].clean_text
        #             )
        #             raise ValueError
        # shush the utilitybelt sentence splitter logging
        import logging
        logger = logging.getLogger()
        logger.setLevel(logging.INFO)

        print('Splitting sentences')
        # create records for every sentence
        records = []
        for i, row in df_in.iterrows():
            rec = {
                'dataset': row.dataset, 'prompt_id': 'NUCLE', 'essay_id': row.essay_id, 'L1': ro
                'score': row.N_errors, 'score_type': 'PercentErrors',
                'age': np.nan,
            }
            prev_end = 0
            text = row.ascii_text
            si = 0
            for start, end, sentence in zip(*get_sentences(text)):
                srec = {}
                srec.update(rec)
                srec['text'] = sentence
                srec['sentence_id'] = si
                srec['trailing_whitespace'] = text[prev_end:start]
                si += 1
                prev_end = end
                records.append(srec)

            if i % 1000 == 0:
                print('{} of {}'.format(i, len(df_in)))

        print('Creating data frame')
        df_out = pd.DataFrame.from_records(records)
        df_out['uid'] = df_out[['dataset', 'essay_id', 'sentence_id']].astype(unicode).apply(lam

        print('{} sentences'.format(len(df_out)))
        print('Saving data frame')
        df_out.to_csv(os.path.join(path, 'NUCLE_sentences.csv'), encoding='utf8', index=False)
```

```
Converting to ASCII
Splitting sentences
0 of 1397
1000 of 1397
Creating data frame
57040 sentences
Saving data frame
```

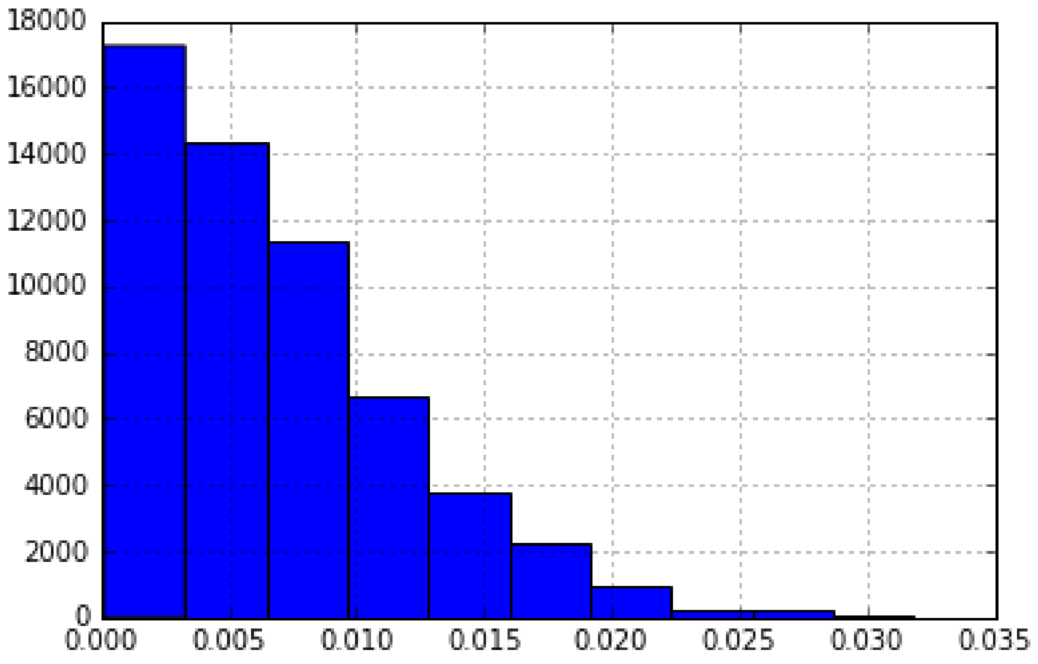
Let's do a little descriptive analysis to make sure we got what we want.

```
In [7]: df = pd.read_csv(os.path.join(path, 'NUCLE_sentences.csv'), encoding='utf8')

In [11]: age = df.groupby('age').size()
         print(age)
         print('{} sentences with age data'.format(pd.notnull(df.age).sum()))

Series([], dtype: int64)
0 sentences with age data


In [12]: score = df.score.hist()
```



```
In [13]: df.text.apply(len).describe()

Out[13]: count    57040.000000
         mean       112.407311
         std         66.918090
         min          1.000000
         25%         67.000000
         50%        107.000000
         75%        150.000000
         max       1005.000000
         Name: text, dtype: float64

In [8]: df.trailing_whitespace = df.trailing_whitespace.fillna('')
        essay1_id = df.essay_id.values[0]
        essay1 = df[df.essay_id == essay1_id]
        essay1['text_plus'] = essay1.trailing_whitespace + essay1.text
        text = ''.join(essay1.text_plus.values)
        print(text)
        print(essay1_id)
```

```
Humans have many basic needs and one of them is to have an environment that can sustain their li

Some countries are having difficulties in managing a place to live for their citizen as they ten

Countries with a lot of inhospitable space need not only to achieve a better space usage, but al

As the number of people grows, the need of habitable environment is unquestionably essential. In
829


/home/smiel/.venvs/rivendell/lib/python2.7/site-packages/ipykernel/__main__.py:4: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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