01_asap_clean

March 21, 2017

2/20/17 - smiel

1 Cleaning the ASAP essay data.

```
In [2]: %matplotlib inline
        # run me when first starting this notebook
        import os

        import numpy as np
        import pandas as pd

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

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

```
In [1]: import sys
        print(sys.maxunicode > 0xffff)
```

```
True
```

The essays are submitted to specific prompts and each prompt was drawn from a single grade. We'll use this data to get ages.

```
In [4]: prompt_to_grade = {
            1: 8, 2: 10, 3: 10, 4: 10, 5: 8, 6: 10, 7: 7, 8: 10
        }
        df_in = pd.read_csv(os.path.join(path, 'all_essays.csv'), encoding='ISO-8859-2')
        df_in['grade'] = df_in.essay_set.apply(lambda es: prompt_to_grade[es])
        df_in['age'] = df_in.grade + 5

        df_in.to_csv(os.path.join(path, 'all_essays_with_age.csv'), encoding='ISO-8859-2', index
```

```
In [5]: 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_with_age.csv'), encoding='ISO-8859-2'

# convert text to ascii
print('Converting to ASCII')
df_in['ascii_text'] = df_in.essay.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': 'ASAP', 'prompt_id': row.essay_set, 'essay_id': row.essay_id, 'L1': '
        'score': np.nan, 'score_type': '', 'age': row.age
    }
    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('Saving data frame')
        df_out.to_csv(os.path.join(path, 'ASAP_sentences.csv'), encoding='utf8', index=False)
```

```
Converting to ASCII
Splitting sentences
0 of 17677
1000 of 17677
2000 of 17677
3000 of 17677
4000 of 17677
5000 of 17677
6000 of 17677
7000 of 17677
8000 of 17677
9000 of 17677
10000 of 17677
11000 of 17677
12000 of 17677
13000 of 17677
14000 of 17677
15000 of 17677
16000 of 17677
17000 of 17677
Creating data frame
Saving data frame
```

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

```
In [6]: df = pd.read_csv(os.path.join(path, 'ASAP_sentences.csv'), encoding='utf8')
```

```
In [7]: age = df.groupby('age').size()
        print(age)
        print('{} sentences with age data'.format(pd.notnull(df.age).sum()))
```

```
age
12     27949
13     70719
15    138417
dtype: int64
237085 sentences with age data


In [8]: df.text.apply(len).describe()

Out[8]: count    237085.000000
        mean         91.332838
        std          55.109264
        min           1.000000
        25%          55.000000
        50%          81.000000
        75%         115.000000
        max        2072.000000
        Name: text, dtype: float64
```