# 12_teccl_clean

March 21, 2017

2/20/17 - smiel

## 1 Cleaning the TECCL 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/teccl'
```

The TECCL data comes to us in a bunch of text files, with almost no metadata. It does have the prompt names, but they would need to be cleaned by hand since they are not standardized and some are in Chinese. It may be easier to just do a topic clustering to find prompts.

```
In [6]: file_list = pd.read_csv(os.path.join(path, 'TECCL_V1.1_list_of_texts.csv'), encoding='ut
        print(file_list.columns)
        print(file_list.head())
```

```
Index([u'Filename', u'Prompt', u'Region', u'Uni type', u'School/uni',
       u'Submission year', u'Submission date', u'Submission time'],
      dtype='object')
     Filename                                   Prompt Region  Uni type  \
0  TECCL00001                    Network Real-name System            NaN
1  TECCL00002  We need parents,we also need independent            NaN
2  TECCL00003   The  Spring  Festival  in  My  Hometown            NaN
3  TECCL00004      Unhealthy Habits of College Students            NaN
4  TECCL00005            Computer and Short-sightedness            NaN

  School/uni  Submission year Submission date Submission time
0                        2011           07-14         9:36:07
1                        2011           12-18        12:55:21
2                        2012           02-16        18:59:54
3                        2014           03-27        23:52:41
4                        2014           06-06        19:57:46
```

Ok. Time to go get the essay texts.

```
In [8]: essays = pd.DataFrame()
        essays['Filename'] = file_list.Filename.values
        texts = []
        for filename in essays.Filename.values:
            with open(os.path.join(path, '01TECCL_V1.1_RAW', '{}.txt'.format(filename)), 'r') as
                texts.append(fin.read().strip())

        essays['text'] = texts
        essays['L1'] = 'CHN'
        essays['essay_id'] = essays.Filename

        # save our progress
        essays.to_csv(os.path.join(path, 'all_essays.csv'), encoding='utf8', index=False)
```

## 1.1 From Essays to Sentences

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

```
In [13]: import sys
         print(sys.maxunicode > 0xffff)

True
```

```
In [9]: 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')

        # while we're at it, let's add a little more metadata
        # the TOEFL is taken as a college entrance test, so let's assume the students were all 1
        df_in['age'] = 17

        # 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': 'TECCL', 'prompt_id': '?', 'essay_id': row.essay_id, 'L1': row.L1,
        'score': np.nan, 'score_type': '', '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, 'TECCL_sentences.csv'), encoding='utf8', index=False)
```

```
Converting to ASCII
Splitting sentences
0 of 9864
1000 of 9864
2000 of 9864
3000 of 9864
4000 of 9864
5000 of 9864
6000 of 9864
7000 of 9864
8000 of 9864
9000 of 9864
Creating data frame
125227 sentences
Saving data frame
```

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

```
In [10]: df = pd.read_csv(os.path.join(path, 'TECCL_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.groupby('score').size()
         print(score)
```

```
Series([], dtype: int64)
```

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

```
Out[13]: count    125227.000000
         mean         79.325736
         std          50.490849
         min           1.000000
         25%          47.000000
         50%          69.000000
         75%         100.000000
         max        2730.000000
         Name: text, dtype: float64
```