05_gachon_clean

March 21, 2017

2/20/17 - smiel

1 Cleaning the Gachon 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/gachon'
```

The Gachon data comes to us in a bunch of text files, with an index csv. We'll use these to create an essays csv.

```
In [2]: essays = pd.read_csv(os.path.join(path, '2.1', 'Gachon_LC_ver2.1.a.df.txt'), encoding='E

        # save progress
        essays.to_csv(os.path.join(path, 'all_essays_meta.csv'), encoding='EUC-KR', index=False)

        print(essays.columns)
        print(essays.head())
```

```
Index([u'Num', u'Time', u'Type', u'Token', u'Student', u'Class', u'Assign',
       u'Waiver', u'Gender', u'StudyYears', u'TOEIC', u'TOEFL', u'IELTS',
       u'Major', u'StudyAbroad', u'Languages', u'BirthYear', u'SchoolYear',
       u'NativeLg', u'FatherNativeLg', u'MotherNativeLg', u'HomeLg',
       u'EleSchLg', u'MidSchLg', u'HighSchLg', u'Confidence', u'StudyReason'],
      dtype='object')
   Num         Time  Type  Token Student              Class  Assign  Waiver  \
0    1  40987.12444   100    180  st0431  09832038  9:00       1       1
1    2   40994.1285    85    132  st0431  09832038  9:00       3       1
2    3  41043.89439    77    125  st0431  09832038  9:00       6       1
3    4  41051.11963    94    138  st0431  09832038  9:00       8       1
4    5  41058.68552   102    149  st0431  09832038  9:00      10       1
```

```
             Gender  StudyYears  \
0  Female (Woman)          7.0
1  Female (Woman)          7.0
2  Female (Woman)          7.0
3  Female (Woman)          7.0
4  Female (Woman)          7.0

                         ...                              SchoolYear  \
0                        ...                          Sophomore (2)
1                        ...                          Sophomore (2)
2                        ...                          Sophomore (2)
3                        ...                          Sophomore (2)
4                        ...                          Sophomore (2)

  NativeLg  FatherNativeLg MotherNativeLg   HomeLg EleSchLg MidSchLg  \
0  Chinese         Chinese        Chinese  Chinese  Chinese  Chinese
1  Chinese         Chinese        Chinese  Chinese  Chinese  Chinese
2  Chinese         Chinese        Chinese  Chinese  Chinese  Chinese
3  Chinese         Chinese        Chinese  Chinese  Chinese  Chinese
4  Chinese         Chinese        Chinese  Chinese  Chinese  Chinese

  HighSchLg Confidence                                      StudyReason
0   Chinese          2  Enjoyment  , Personal Development   ...
1   Chinese          2  Enjoyment  , Personal Development   ...
2   Chinese          2  Enjoyment  , Personal Development   ...
3   Chinese          2  Enjoyment  , Personal Development   ...
4   Chinese          2  Enjoyment  , Personal Development   ...

[5 rows x 27 columns]
```

Let's drop the columns we won't use to prevent weird formatting errors.

```
In [3]: essays.drop(
            [
                u'Time', u'Type', u'Token', u'Class',
                u'Waiver', u'Gender', u'StudyYears',
                u'Major', u'StudyAbroad', u'Languages', u'BirthYear',
                u'NativeLg', u'FatherNativeLg', u'MotherNativeLg',
                u'EleSchLg', u'MidSchLg', u'HighSchLg', u'Confidence', u'StudyReason'
            ],
            inplace=True,
            axis=1
        )
        print(essays.head())

   Num Student  Assign TOEIC TOEFL  IELTS       SchoolYear   HomeLg
0    1  st0431       1   NaN   NaN    NaN  Sophomore (2)  Chinese
```

```
1    2  st0431       3   NaN   NaN    NaN  Sophomore (2)  Chinese
2    3  st0431       6   NaN   NaN    NaN  Sophomore (2)  Chinese
3    4  st0431       8   NaN   NaN    NaN  Sophomore (2)  Chinese
4    5  st0431      10   NaN   NaN    NaN  Sophomore (2)  Chinese
```

```
In [4]: print(essays.groupby('HomeLg').size())
```

```
HomeLg
Chinese                                          212
German                                             1
Korean                                         15831
Korean, Spanish                                    9
Spanish, and only with my parents: Korean          3
chinese                                           55
dtype: int64
```

To ensure we have a regular sample, let's drop the students whose home language is not Korean.

```
In [5]: essays = essays[essays.HomeLg == 'Korean']
        print(len(essays))
        essays.drop(['HomeLg'], inplace=True, axis=1)
        print(essays.head())
```

```
15831
     Num Student  Assign TOEIC TOEFL  IELTS       SchoolYear
249  250  st0394      12   860   NaN    NaN  Sophomore (2)
250  251  st0394      13   860   NaN    NaN  Sophomore (2)
251  252  st0394      14   860   NaN    NaN  Sophomore (2)
252  253  st0394      15   860   NaN    NaN  Sophomore (2)
253  254  st0394      16   860   NaN    NaN  Sophomore (2)
```

```
In [6]: essays['L1'] = 'KOR'
        essays['essay_id'] = essays.Num

        # save progress
        essays.to_csv(os.path.join(path, 'all_essays_meta.csv'), encoding='EUC-KR', index=False)
```

Let's take a look at SchoolYear to get grade info

```
In [7]: print(essays.groupby('SchoolYear').size())
```

```
SchoolYear
Freshman (1)       126
Junior (3)        2017
Senior (4)         234
Sophomore (2)    13454
dtype: int64
```

```
In [8]: # translate that to age
        school_year_to_grade = {'Fre': 13, 'Sop': 14, 'Jun': 15, 'Sen': 16}
        essays['grade'] = essays.SchoolYear.apply(lambda sy: school_year_to_grade[sy[:3]])
        essays['age'] = essays.grade + 5

        # save progress
        essays.to_csv(os.path.join(path, 'all_essays_meta.csv'), encoding='EUC-KR', index=False)
```

Ok. Time to go get the essay texts.

```
In [9]: import codecs

        texts = []
        for essay_id in essays.essay_id.values:
            file_path = os.path.join(path, '2.1', 'GLCfiles2.1', '{}.txt'.format(essay_id))
            with codecs.open(file_path, 'r', encoding='EUC-KR') as fin:
                texts.append(fin.read().strip())

        df_in = essays.copy()

        df_in['text'] = texts

        # save our progress
        df_in.to_csv(os.path.join(path, 'all_essays.csv'), encoding='EUC-KR', index=False)
```

```
In [10]: # drop school year so there are no foreign characters
         df_in.drop(['SchoolYear'], inplace=True, axis=1)
         df_in.to_csv(os.path.join(path, 'all_essays.csv'), encoding='EUC-KR', 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 [11]: import sys
         print(sys.maxunicode > 0xffff)
```

```
True
```

```
In [30]: 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='EUC-KR')

         # 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)

def asfloat(x):
    try:
        return float(x)
    except:
        print '{} not a float'.format(x)
        return np.nan

print('Splitting sentences')
# create records for every sentence
records = []
for i, row in df_in.iterrows():
    rec = {
        'dataset': 'Gachon', 'prompt_id': row.Assign, 'essay_id': '{}_{}'.format(row.St
        'L1': row.L1, 'score': np.nan, 'score_type': '', 'age': row.age,
        'student_level_TOEIC': asfloat(285.0 if row.TOEIC == '270~300' else row.TOEIC),
        'student_level_TOEFL': asfloat(row.TOEFL),
        'student_level_IELTS': asfloat(row.IELTS)
    }
    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(la

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

```
Converting to ASCII
Splitting sentences
0 of 15831
1000 of 15831
2000 of 15831
3000 of 15831
4000 of 15831
 x not a float
 x not a float
 x not a float
 x not a float
 x not a float
 x not a float
 x not a float
5000 of 15831
6000 of 15831
7000 of 15831
8000 of 15831
9000 of 15831
10000 of 15831
11000 of 15831
12000 of 15831
13000 of 15831
14000 of 15831
15000 of 15831
```

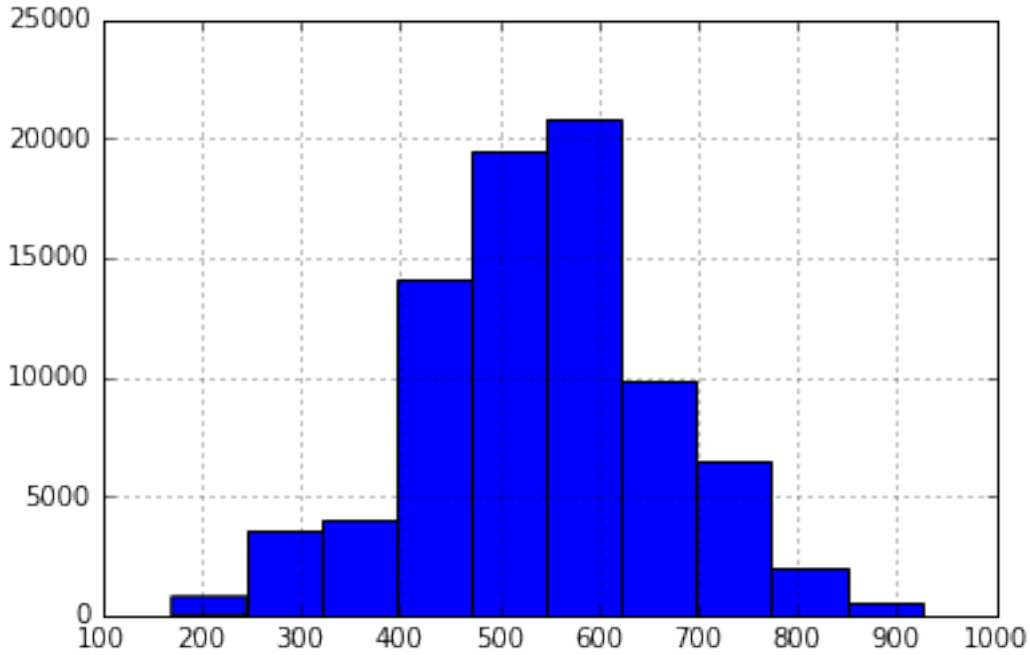
```
Creating data frame
147844 sentences
Saving data frame
```

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

```
In [31]: df = pd.read_csv(os.path.join(path, 'Gachon_sentences.csv'), encoding='utf8')
```

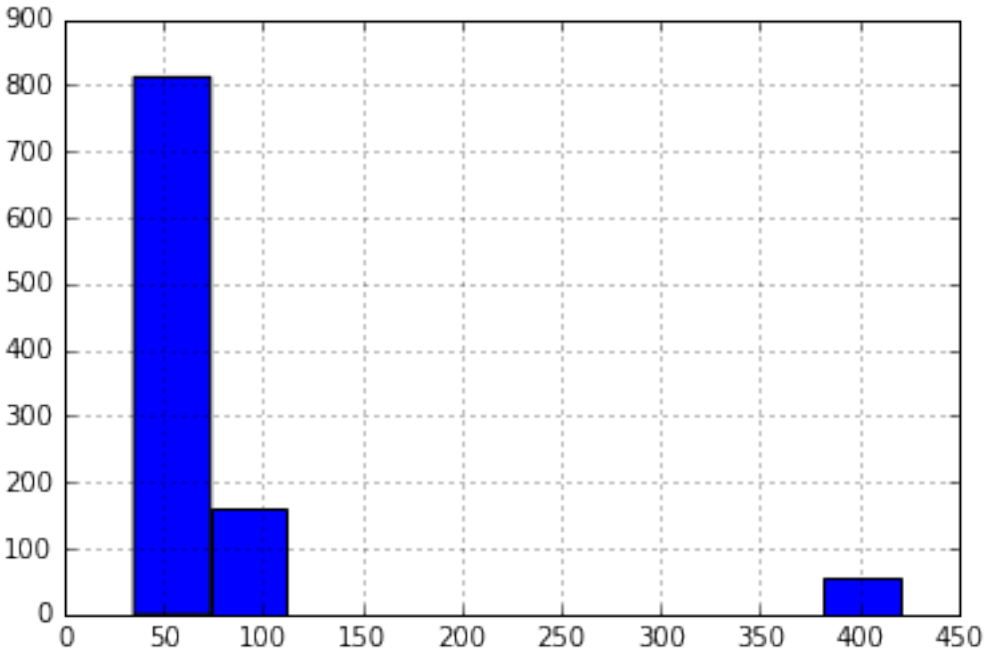
```
In [36]: df.student_level_TOEIC[pd.notnull(df.student_level_TOEIC)].hist()
```

```
Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7e238cec50>
```



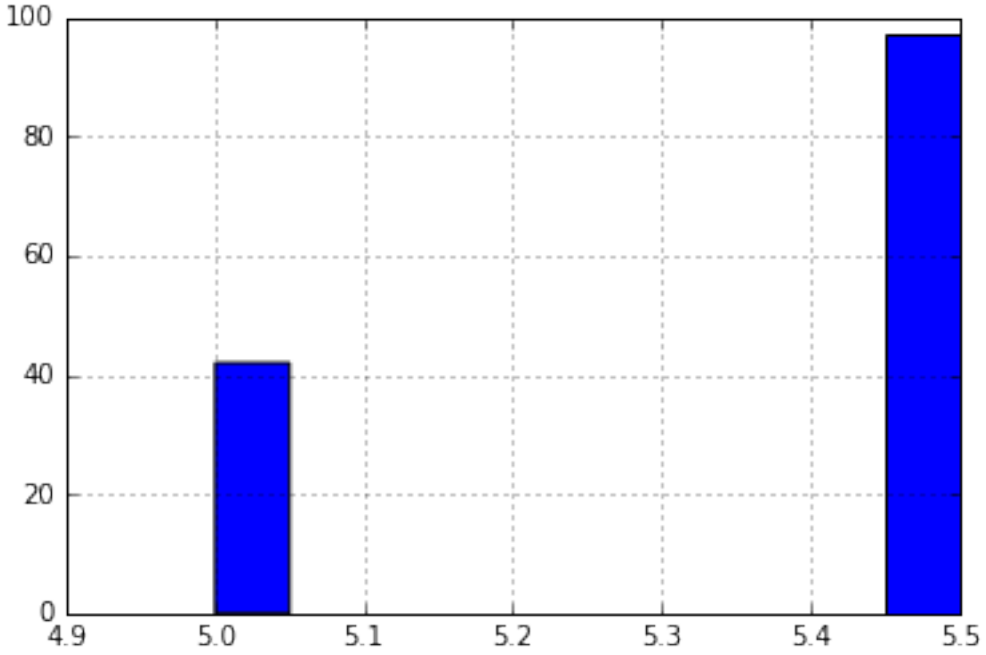

```
In [40]: df.student_level_TOEFL[pd.notnull(df.student_level_TOEFL)].hist()
```

```
Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7e22c9e250>
```



In [41]: `df.student_level_IELTS[pd.notnull(df.student_level_IELTS)].hist()`

Out[41]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f7e23190590>`



```
In [37]: age = df.groupby('age').size()
         print(age)
         print('{} sentences with age data'.format(pd.notnull(df.age).sum()))
```

```
age
18      1063
19    125946
20     18735
21      2100
dtype: int64
147844 sentences with age data
```

```
In [38]: score = df.groupby('score').size()
         print(score)
```

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

```
In [39]: df.text.apply(len).describe()
```

```
Out[39]: count    147844.000000
         mean         57.759239
         std          32.728292
         min           1.000000
         25%          37.000000
         50%          52.000000
         75%          71.000000
         max        1527.000000
         Name: text, dtype: float64
```

```
In [42]: 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)
```

```
I think people are eating less healthy foods than they used to.
 First of all, people are getting busier so they don't have enough time to prepare foods. So peo
 Also, most instant foods are including not so many vegetables but food addictives. These food a
 Therefore, people eat less healthy foods than they used to.
st0394_250
```

```
/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#
```