train_and_evaluate_percieving_n5000_and_judging_n10000

March 12, 2018

In [9]:
```
import pandas as pd
import os
import requests
import numpy as np
import operator
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import seaborn as sns
import numpy as np
```

In [10]:
```
!pwd
```

```
/Users/mos/Dropbox/memeticscience/typealyzer-dataset/notebooks/jungian_classification
```

In [11]:
```
df_pickle_path = "../../pickles/dataframe_survey_2018-01-23_enriched.pickle"
```

In [12]:
```
indata = pd.read_pickle(df_pickle_path)
indata[["actual","actual_temp","is_s"]].head(5)
```

Out[12]:

	actual	actual_temp	is_s
1	INFJ	nf	0
2	INFP	nf	0
3	INTP	nt	0
5	ENFJ	nf	0
10	INFP	nf	0

In [15]:
```
indata.domain.value_counts()
```

Out[15]:
```
tumblr       21938
blogspot       513
wordpress      468
Name: domain, dtype: int64
```

In [17]:
```
indata.lang.value_counts().head()
```

Out[17]:
```
en    22588
fr       50
da       34
de       25
no       23
Name: lang, dtype: int64
```

1 Filter out English texts only

```
In [18]: len(indata)
```

```
Out[18]: 22919
```

```
In [19]: indata = indata[indata.lang == "en"]
```

```
In [20]: len(indata)
```

```
Out[20]: 22588
```

2 Add columns percieving and judging for evaluation of s/n and t/f classifiers

```
In [ ]: indata["perc_func"] = indata.actual_temp.str.extract("(\w)\w", expand=False)
        indata["judg_func"] = indata.actual_temp.str.extract("\w(\w)", expand=False)
```

```
In [22]: len(indata[pd.isnull(indata["tokens"])])
```

```
Out[22]: 0
```

```
In [23]: indata[pd.isnull(indata["tokens"])]
```

```
Out[23]: Empty DataFrame
         Columns: [url, typealyzer, actual, e, s, t, sntf_s, sntf_n, sntf_t, sntf_f, date, tex
         Index: []

         [0 rows x 117 columns]
```

3 Inspect original data function distributions

```
In [24]: s_series = indata[indata.perc_func == "s"]["tokens"]
         n_series = indata[indata.perc_func == "n"]["tokens"]
         t_series = indata[indata.judg_func == "t"]["tokens"]
         f_series = indata[indata.judg_func == "f"]["tokens"]

         avg_tkns = {
             "s":s_series.mean(),
             "n":n_series.mean(),
             "t":t_series.mean(),
             "f":f_series.mean()
         }
         avg_tkns
```

```
Out[24]: {'f': 488.2908894968084,
          'n': 511.4496560721063,
          's': 457.63189127105665,
          't': 511.27211970074814}
```

```
In [29]: indata.perc_func.value_counts()

Out[29]: n    16864
         s     5224
         Name: perc_func, dtype: int64

In [30]: indata.judg_func.value_counts()

Out[30]: f    12063
         t    10025
         Name: judg_func, dtype: int64
```

4 Sample equal size text chunks for training and evaluation data

See: Pandas sample()

4.1 Percieving function

```
In [129]: # We have 5224 cases in the smallest class s
          perc_samples = pd.concat([
                      indata[indata.perc_func == "s"].sample(5000, random_state=123456)[["text"
                      indata[indata.perc_func == "n"].sample(5000, random_state=123456)[["text"
                      ])

In [130]: len(perc_samples)

Out[130]: 10000

In [131]: perc_samples.perc_func.value_counts()

Out[131]: n    5000
          s    5000
          Name: perc_func, dtype: int64

In [132]: perc_samples.judg_func.value_counts()

Out[132]: f    5648
          t    4352
          Name: judg_func, dtype: int64

In [36]: perc_samples.to_pickle("jung_percieving_functions_samples_blogs_n5000.pickle")

In [37]: perc_samples = pd.read_pickle("jung_percieving_functions_samples_blogs_n5000.pickle")
```

4.2 Judging function

```
In [145]: # We have 10025 cases in the smallest class t
          judg_samples = pd.concat([
                      indata[indata.judg_func == "t"].sample(10000, random_state=123456)[["text
                      indata[indata.judg_func == "f"].sample(10000, random_state=123456)[["text
                      ])

In [146]: len(judg_samples)

Out[146]: 20000

In [147]: judg_samples.judg_func.value_counts()

Out[147]: t    10000
          f    10000
          Name: judg_func, dtype: int64

In [148]: judg_samples.perc_func.value_counts()

Out[148]: n    15314
          s     4686
          Name: perc_func, dtype: int64

In [43]: judg_samples.to_pickle("jung_judging_functions_samples_blogs_n10000.pickle")

In [44]: judg_samples = pd.read_pickle("jung_judging_functions_samples_blogs_n10000.pickle")
```

5 Setup uClassify classifer and prepare training and evaluation datasets

The variable `os.environ["UCLASSIFY_WRITE"]` is created by adding a line to e.g. `~/.profile`:

```
export UCLASSIFY_WRITE = "<your_uclassify_write_key>"
```

6 Split percieving samples into train and eval subsets.

```
In [133]: len(perc_samples)

Out[133]: 10000

In [134]: # prepare column to keep track of what's been used for training
          zeros = np.zeros(len(perc_samples))
          perc_samples["perc_training_set"] = zeros
          perc_samples["perc_training_set"] = perc_samples.perc_training_set.astype("int")
          perc_samples.head(3)
```

Out[134]:

	text	tokens	perc_func	judg_func	actual_temp	perc_training_set
8623	Sonny Jooooooooon INDEX ASK PAST THEME Sonny J...	386	s	f	sf	0
11987	Log in | Tumblr Sign up Terms Privacy Posted b...	52	s	t	st	0
5340	a thing of blood I' hi im logan and i love the ...	440	s	f	sf	0

In [135]:
```
sn_traing_set_size = 3500 # e.g. 3500 is 70% of 5000 samples
perc_s_train = perc_samples[perc_samples.perc_func == "s"].sample(sn_traing_set_size)
perc_n_train = perc_samples[perc_samples.perc_func == "n"].sample(sn_traing_set_size)

perc_train = perc_s_train.union(perc_n_train)

perc_samples.loc[perc_train, "perc_training_set"] = 1
perc_samples.head(3)
```

Out[135]:

	text	tokens	perc_func	judg_func	actual_temp	perc_training_set
8623	Sonny Jooooooooon INDEX ASK PAST THEME Sonny J...	386	s	f	sf	1
11987	Log in | Tumblr Sign up Terms Privacy Posted b...	52	s	t	st	1
5340	a thing of blood I' hi im logan and i love the ...	440	s	f	sf	1

In [136]:
```
len(perc_samples[perc_samples.perc_training_set == 1])
```

Out[136]: 7000

In [162]:
```
# Separate evaluation DataFrame
perc_eval_set = perc_samples[perc_samples.perc_training_set == 0]
perc_eval_set.head(3)
```

Out[162]:

	text	tokens	perc_func	judg_func	actual_temp	perc_training_set
4604	*tamp tamp* *tamp tamp* Index : Ask : submit :...	543	s	t	st	0
14753	this and this and this this and this and this ...	945	s	f	sf	0
9419	you're the very best of us n you're the very b...	384	s	f	sf	0

In [121]:
```
len(perc_eval_set)
```

Out[121]: 249

6.1 Train SN classifer

https://uclassify.com/manage/classifiers/jungian-cognitive-function-sensing-intuition

```
In [51]: def train_jung_cognitive_functions_en_classes(func, classifier):
             """Presupposes that classifier is created and that setup_jung_functions_en_classe
             func: expects one of ["s","n","t","f"]
             classifier: expects on of ["sntf", "tf", "sn"]

             """
             trained_ix = []
             text_count = 1
             if classifier == "sn":
                 for ix, row in perc_samples.loc[(perc_samples.perc_func == name) & (perc_sampl
                     trained_ix.append(ix)
                     data = {"texts":[row["text"]]}
                     header = {"Content-Type": "application/json",
                          "Authorization": "Token " + os.environ["UCLASSIFY_WRITE"]}

                     response = requests.post('https://api.uclassify.com/v1/me/jungian-cogniti
                             json = data,
                             headers = header)
                     print("{}:{}".format(name, text_count))
                     text_count += 1

             elif classifier == "tf":
                 for ix, row in judg_samples.loc[(judg_samples.judg_func == name) & (judg_sampl
                     trained_ix.append(ix)
                     data = {"texts":[row["text"]]}
                     header = {"Content-Type": "application/json",
                          "Authorization": "Token " + os.environ["UCLASSIFY_WRITE"]}

                     response = requests.post('https://api.uclassify.com/v1/me/jungian-cogniti
                             json = data,
                             headers = header)
                     if text_count % 100 == 0:
                         print("{}:{}".format(name, text_count))
                     text_count += 1


             print("Finished training Jung Cognitive Functions.")
             return trained_ix
```

```
In [ ]: perc_trained_ix = []
        for name in ["s","n"]:
            functions_trained_ix = train_jung_cognitive_functions_en_classes(name, classifier="
            perc_trained_ix.append(perc_trained_ix)
```

```
In [53]: print("length perc_eval_set: {}".format(len(perc_eval_set)))
         print("length perc_trained_ix: {}".format(len(perc_trained_ix)))
```

```
length perc_eval_set: 3000
length perc_trained_ix: 2
```

7 Split judging samples into train and eval subsets.

In [149]:
```
# prepare column to keep track of what's been used for training
zeros = np.zeros(len(judg_samples))
judg_samples["judg_training_set"] = zeros
judg_samples["judg_training_set"] = judg_samples.judg_training_set.astype("int")
judg_samples.head(3)
```

Out[149]:

	text	tokens	perc_func	\
22981	it is what it is About Name: Heidi Age:16 Wher...	565	s	
24378	https://www.tumblr.com/themes/by/leentheme htt...	582	n	
5187	three things cannot be long hidden I' three thi...	516	n	

	judg_func	actual_temp	judg_training_set
22981	t	st	0
24378	t	nt	0
5187	t	nt	0

In [150]:
```
tf_traing_set_size = 7000 # e.g. 7000 is 70% of 10000 samples
judg_t_train = judg_samples[judg_samples.judg_func == "t"].sample(tf_traing_set_size)
judg_f_train = judg_samples[judg_samples.judg_func == "f"].sample(tf_traing_set_size)

judg_train = judg_t_train.union(judg_f_train)

judg_samples.loc[judg_train, "judg_training_set"] = 1
judg_samples.head(15)
```

Out[150]:

	text	tokens	perc_func	\
22981	it is what it is About Name: Heidi Age:16 Wher...	565	s	
24378	https://www.tumblr.com/themes/by/leentheme htt...	582	n	
5187	three things cannot be long hidden I' three thi...	516	n	
4307	none gf with left feel 12442 August 1st, 201...	499	s	
2606	God, Faith, & Fitness God, Faith, & Fitness Me...	924	n	
18214	big hype, big letdown i'm charlotte and i li...	101	s	
10718	Love the life you live Live the life you love ...	353	n	
20277	- - - - - - - | momo | 14 | | ESTJ | |...	473	s	
846	Cynically Marvelous | It's Axiomatic. Cynicall...	6973	n	
16405	I'll just pretend that youth will never end I'...	693	n	
11753	Something-or-other Something-or-other Let's ju...	483	n	
18004	/ ._> | | | '_>/ ._>| . \/ ._>| || |/ ._> | _/...	180	n	
23577	WINTERFELL About Mariele. 21. Germany. ISTJ. M...	102	s	
4829	The bears are in 684,715 plays we-r-who-...	445	s	
4225	XANADU XANADU	2	n	

```
      judg_func actual_temp  judg_training_set
22981         t          st                  0
24378         t          nt                  0
5187          t          nt                  0
4307          t          st                  1
2606          t          nt                  1
18214         t          st                  1
10718         t          nt                  1
20277         t          st                  1
846           t          nt                  1
16405         t          nt                  1
11753         t          nt                  0
18004         t          nt                  1
23577         t          st                  1
4829          t          st                  0
4225          t          nt                  0
```

```
In [168]: # Separate evaluation DataFrame
          judg_eval_set = judg_samples[judg_samples.judg_training_set == 0]
          judg_eval_set.head(3)
```

```
Out[168]:                                                    text  tokens perc_func  \
22981  it is what it is About Name: Heidi Age:16 Wher...     565         s
24378  https://www.tumblr.com/themes/by/leentheme htt...     582         n
5187   three things cannot be long hidden I' three thi...     516         n

      judg_func actual_temp  judg_training_set
22981         t          st                  0
24378         t          nt                  0
5187          t          nt                  0
```

7.1 Train TF classifier

https://uclassify.com/manage/classifiers/jungian-cognitive-function-thinking-feeling

```
In [ ]: tf_trained_ix = []
        for name in ["t","f"]:
            try:
                tf_trained_ix = train_jung_cognitive_functions_en_classes(name, classifier="tf
                tf_trained_ix.append(tf_trained_ix)
            except Exception as e:
                print(e)
```

```
In [58]: print("length judg_eval_set: {}".format(len(judg_eval_set)))
         print("length tf_trained_ix: {}".format(len(tf_trained_ix)))
```

```
length judg_eval_set: 6000
length tf_trained_ix: 7001
```

8 Classify percieving function

```
In [59]: def classify_jung_percieving_function_of_text(text):
             """Does what it says, pretty much."""
             header = {"Content-Type": "application/json",
                       "Authorization": "Token " + os.environ["UCLASSIFY_READ"]}
             data = {"texts":[text]} # send a one-item list for now, since we don't have a fee
             result = requests.post("https://api.uclassify.com/v1/prfekt/jungian-cognitive-func
                                    json = data,
                                    headers = header)
             json_result = result.json()

             res_dict = {"s":0, "n":0}

             for classItem in json_result[0]["classification"]:
                 res_dict[classItem["className"]] = classItem["p"]

             sorted_dict = sorted(res_dict.items(), key=operator.itemgetter(1), reverse=True)
             return sorted_dict

In [60]: zeros = np.zeros(len(perc_eval_set))
         sn_results = []
         row_cnt = 1
         for ix, row in perc_eval_set.iterrows():
             print("row: {} of {}".format(row_cnt, len(perc_eval_set)),end="\r")
             res = classify_jung_percieving_function_of_text(row["text"])
             sn_results.append(res[0][0])
             row_cnt += 1

row: 3000 of 3000

In [61]: len(sn_results)

Out[61]: 3000
```

9 Classify TF

```
In [62]: def classify_jung_judging_function_of_text(text):
             """Does what it says, pretty much."""
             header = {"Content-Type": "application/json",
                       "Authorization": "Token " + os.environ["UCLASSIFY_READ"]}
             data = {"texts":[text]} # send a one-item list for now, since we don't have a fee
             result = requests.post("https://api.uclassify.com/v1/prfekt/jungian-cognitive-func
                                    json = data,
                                    headers = header)
             json_result = result.json()

             res_dict = {"t":0, "f":0}
```

```
            for classItem in json_result[0]["classification"]:
                res_dict[classItem["className"]] = classItem["p"]

            sorted_dict = sorted(res_dict.items(), key=operator.itemgetter(1), reverse=True)
            return sorted_dict
```

```
In [63]: zeros = np.zeros(len(judg_eval_set))
         tf_results = []
         row_cnt = 1
         for ix, row in judg_eval_set.iterrows():
             print("row: {} of {}".format(row_cnt, len(judg_eval_set)),end="\r")
             res = classify_jung_judging_function_of_text(row["text"])
             tf_results.append(res[0][0])
             row_cnt += 1
```

```
row: 6000 of 6000
```

10 Evaluation of percieving classification

```
In [158]: len(perc_eval_set)
```

```
Out[158]: 3000
```

```
In [159]: len(sn_results)
```

```
Out[159]: 3000
```

```
In [163]: perc_eval_set = pd.concat([perc_eval_set,
                                    pd.DataFrame(sn_results, index=perc_eval_set.index)
                                   ], axis=1, ignore_index=True)
          perc_eval_set.columns = ["text","tokens","perc_func","judg_fuc","actual_temp","perc_t
          perc_eval_set.to_pickle("classification_results_percieving_function_blogs_n5000_data
          print(perc_eval_set.head(3))
```

```
                                                   text  tokens perc_func  \
4604   *tamp tamp* *tamp tamp* Index : Ask : submit :...     543         s
14753  this and this and this this and this and this ...     945         s
9419   you're the very best of us n you're the very b...     384         s

      judg_fuc actual_temp  perc_training_set sn
4604         t          st                  0  s
14753        f          sf                  0  n
9419         f          sf                  0  s
```

10.1 Classification report percieving

```
In [164]: sn_cr = classification_report(perc_eval_set['perc_func'], perc_eval_set['sn'])
          print(sn_cr)
```

```
             precision    recall  f1-score   support

          n       0.59      0.52      0.55      1500
          s       0.57      0.64      0.60      1500

avg / total       0.58      0.58      0.58      3000
```

10.2 Percieving accuracy

```
In [165]: sn_accuracy = sum(perc_eval_set['perc_func']==perc_eval_set['sn'])/len(perc_eval_set)
          print(sn_accuracy)
```

```
0.5803333333333334
```

10.3 Percieving Kappa

```
In [166]: sn_kappa = sn_accuracy - (1/len(perc_eval_set))/(1 - len(perc_eval_set))
          print(sn_kappa)
```

```
0.5803334444814938
```

11 Evaluation of judging classificiation

```
In [169]: judg_eval_set = pd.concat([judg_eval_set,
                                     pd.DataFrame(tf_results, index=judg_eval_set.index)
                                    ], axis=1, ignore_index=True)
          judg_eval_set.columns = ["text","tokens","perc_func","judg_func","actual_temp","judg_
          judg_eval_set.to_pickle("classification_results_judging_function_blogs_n10000_datafra
          print(judg_eval_set.head(3))
```

```
                                                    text  tokens perc_func  \
22981  it is what it is About Name: Heidi Age:16 Wher...     565         s
24378  https://www.tumblr.com/themes/by/leentheme htt...     582         n
5187   three things cannot be long hidden I three thi...     516         n

      judg_func actual_temp  judg_training_set tf
22981         t          st                  0  t
24378         t          nt                  0  t
5187          t          nt                  0  f
```

11.1 Classification report judging

```
In [170]: tf_cr = classification_report(judg_eval_set['judg_func'], judg_eval_set['tf'])
          print(tf_cr)
```

```
             precision    recall  f1-score   support

          f       0.57      0.65      0.61      3000
          t       0.60      0.52      0.55      3000

avg / total       0.59      0.58      0.58      6000
```

11.2 Judging accuracy

```
In [171]: tf_accuracy = sum(judg_eval_set['judg_func']==judg_eval_set['tf'])/len(judg_eval_set)
          print(tf_accuracy)
```

```
0.585
```

11.3 Judging Kappa

```
In [172]: tf_kappa = tf_accuracy - (1/len(judg_eval_set))/(1 - len(judg_eval_set))
          print(tf_kappa)
```

```
0.5850000277824081
```

12 Conclusion and further research

- Definately an improvement in classification results.
- Since Jungs cognitive functions are really dichotomies of sensing <-> intuition and thinking <-> feeling, two new classifiers should be created to see if results improve.
- Another important thing is that no pre-processing of the texts have been done. Use TF-IDF to separate out noice from relevant features?