Readme

February 6, 2017

1 The Etymological Machine Usage

All files were written and tested with python 3.5

1.1 Pattern-Match Categorizer

`regex_categorizer.py` is a script that takes the contents of `etymonline.tsv`, which should contain the data scraped from the website and cleaned of HTML and prints their categories to `categorized.tsv`

1.2 Statistical Categorizer

`statistical_categorizer.py` contains several functions that are best called from a Jupyter notebook or iPython session.

`run_cv_test` takes a percentage of the dataset (between 0 and 1), a filename prefix that feature vectors will be written to or read from, and several boolean values corresponding to the features to be used, and whether to write to or read from the named file.

Function definition

```
def run_CV_test(test_percent, iofilename, bow=True,
                letters=False,
                year=False, syllables=False,
                verbose=True, new_design_matrix=False):
```

It prints the accuracy, precision, recall, and f-score of a 5-fold Cross Validation test using a linear kernel classifier.

`makelinearmodels` takes a filename, a holdout percentage to test the accuracy of the trained model, and the same boolean keyword arguments as `run_cv_test`.

```
def makelinearmodels(filename, holdout_percent,
                     normalize_X=False,
                     test_percent=1, bow=True, letters=False,
                     year=False, syllables=False,
                     new_design_matrix=False):
```

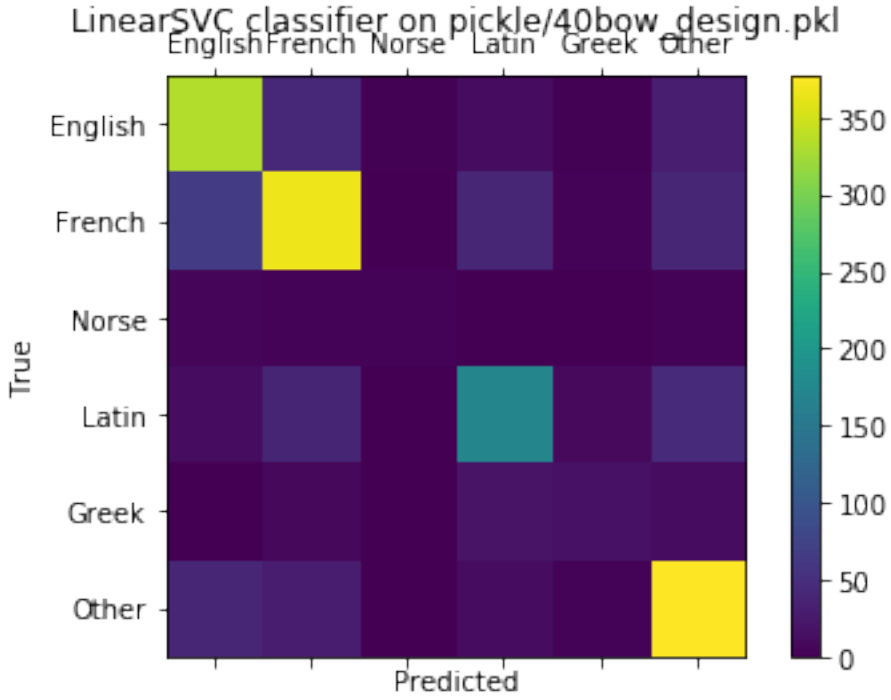
This funcption returns the model it makes, and it prepares a confusion matrix that can be displayed with `pl.show()`

In [7]:
```
from statistical_categorizer import *

        clf = makelinearmodels("pickle/40bow_", .1)

        pl.show()
```

```
Initializing wordlist
Initializing letter list
Initializing syllable list
Reading in design and target matrices
training linearSVC
done
metrics.f1score:
0.587027177347
```



In [2]:
```
# clf = makelinearmodels("pickle/12_bow_letters_year_", .1)


        # if you want to redo the vectors, set new_design_matrix to true

        # default values
        # def makelinearmodels(filename, holdout_percent, normalize_X=False,
        #                      test_percent=1, bow=True, letters=False, year=False,
```

```
#                       syllables=False):

# filename must match the testpercent/letters/year arguments.

# pl.show()


# classify everything, including the data that didn't have a classificatio
# new_category_dict = {}
# transformer = {
#     0: "English",
#     1: "French",
#     2: "Norse",
#     3: "Latin",
#     4: "Greek",
#     5: "Other",
# }

# for word in etymdict.keys():
#     vector = featurizer(word, etymdict[word], letters=True, year=True)
#     prediction = clf.predict(vector)
#     prediction = prediction[0]
#     new_category_dict[word] = transformer[prediction]

# if the result is satisfactory,
# write new_category_dict out to a file using tsvopener.writeitout
```

1.3 Etymachine

`etymachine.py` contains visualization tools for the analysis of real texts.

`make_lexicon_pie` takes an input dictionary (should be `new_category_dict`) and makes a pie chart of the full lexicon. This should be approximately the same as the "bad chart from the internet"

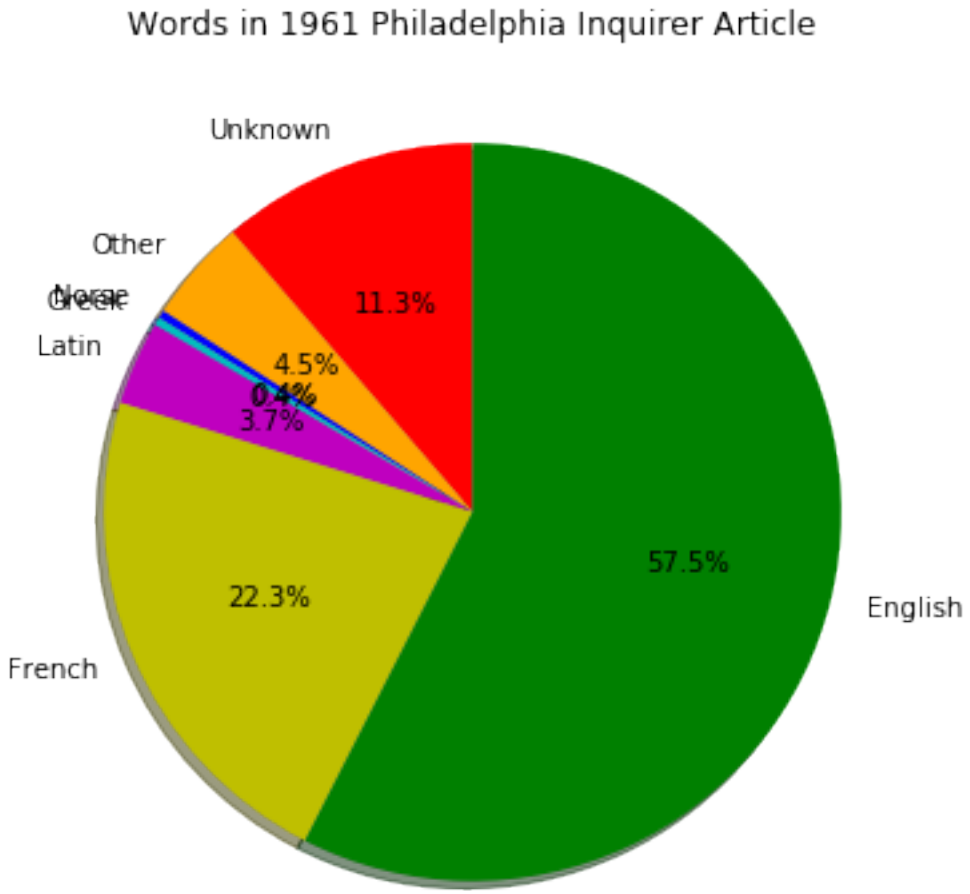
`make_analysis_pie` takes a a list of sentences, in nltk.corpus format (lists of `(word, tag)` tuples), a title, and several boolean arguments: whether to separate them into types, whether to show the chart on completion, whether to include unknown words in the total.

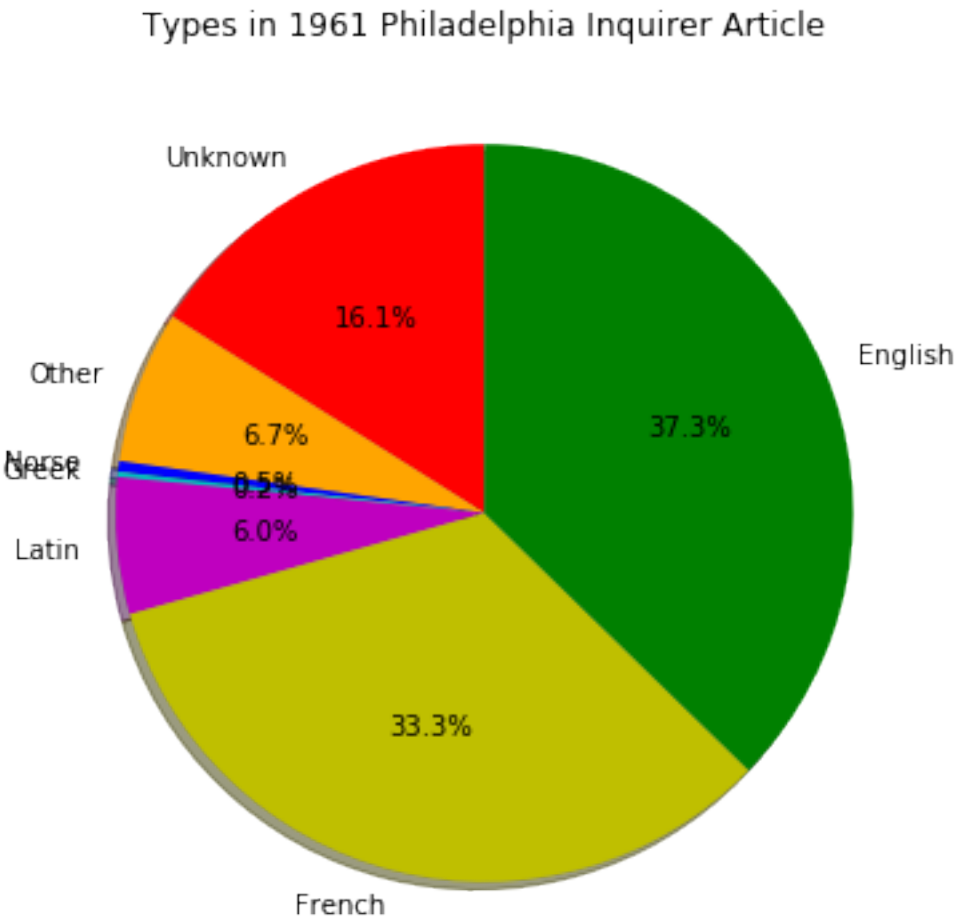
`plot_clustered_stacked` creates a stacked-bar-chart from a set of pandas dataframes. This code was mostly borrowed from this Stack Overflow answer. http://stackoverflow.com/questions/22787209/how-to-have-clusters-of-stacked-bars-with-python-pandas

```
In [5]: from Etymachine import *

        sentences = brown.tagged_sents("ca09")
        title = "Words in 1961 Philadelphia Inquirer Article"
        make_analysis_pie(sentences, title, show=False)
        title = "Types in 1961 Philadelphia Inquirer Article"
```

```
make_analysis_pie(sentences, title, show=False, token=True)
pl.show()
```




Types in 1961 Philadelphia Inquirer Article
Unknown
16.1%
English
37.3%
Other
6.7%
Norse
Greek
0.5%
0.2%
Latin
6.0%
33.3%
French