Report - Surname Assignment

Oscar Galindo

Dr. Nigel Ward

Topics in Language Processing

August 31, 2020

Surname Assignment

Introduction:

For this assignment a set of rules known as *hypothesis* was implemented to map surnames to countries of origin. The rules implemented were derived from a file that contained samples of last names that were ordered by country of origin. The program was then tested against a list of 10 last names that were not included in the data that was used to produce the *hypothesis* of classification.

Design and Implementation:

Rationale for *hypothesis* creation: Languages contain a structure that tends to be unique because of the population that uses it. For example, although both, North Korea and South Korea use Korean as their language, it is widely known that the vocabulary and the writing styles are extremely different between both nations. Hence, nowadays South Koreans have issues understanding written “North" Korean and vice-versa. But even though this is true for most languages, granted to a lesser degree, languages still share a common architecture to the words that are valid to be used in the language, hence, surnames are no different. For example, the last name “Galindo” is still written “Galindo” in any of the Spanish-speaking countries in the world. So, for this exercise there is an attempt at extracting the common architecture from every language by defining unique use of letters, or specific substrings.

Hypothesis:

Comment: To create the set of rules the examples in “**surnames-dev.csv**” were used to define the substrings for every language.

- Arabic:
  - Substrings:
    ‘fa’,'eif','ha','al','ba','ou','uo','ui','sar','dd','ad','ha','ni','nem','ki''zz','az','azz','am','sha','ai','ouf'

- Chinese:
  - Substrings:
    ‘ao’,’oa','wa','jin','qi','dai','Tz','ai','ia','an','xi','oo','ang','ni','in','sh'

- Czech:
  - Substrings:
    ‘ka’,'ff','sek','wa','sch','deh','ss','jj','ik','pp','vl','ski','ze','rich','cova','tak','ovsky','warz','schm','ich','witz','pp'

Report - Surname Assignment

- Dutch:
  - Substrings: ‘ee’,'jer','aye','oo','gg','ee','aa','ker','ss','nn','kk','ke'

- English:
  - Substrings:
    'll','oo','wood','ers','bb','ss','way','ee','dow','ach','gg','van','tt','pp','down','r
    oss','ver','ker','ren','ran','owe','mc'

- French:
  - Substrings:
    ‘le’,'ss','and','aux','oli','vage','eux','tit','ieu','nna','bon','boi','dú','neau','eau',
    'oy','ias','quet','reau','hur','agne'

- German:
  - Substrings:
    ‘tt’,'ber','oh','eier','ich','altz','ieck','enz','hol','uh','aun','del','ō','eis','mm','kl',
    'aub'

- Greek:
  - Substrings: ‘ous’,'kos','nos','mos','los','is','as'

- Irish:
  - Substrings: “o’",'ach',"ley","lly",'all','an','mac','oi','inn','ald'

- Italian:
  - Substrings:
    “eri","ari","icci","one","chi","ggi","gio","à","tti","di","gari","fin","lli","to","oli","
    dia","zzi","Scor","ini","ecce","acco","ani"

- Japanese:
  - Substrings:
    ‘shi','oku','ida','iko','aka','tsu','awa','aya','ishi','uno','ima','chi','uya','hara','z
    ak','aki','ji','uch'

- Korean:
  - Substrings: "mo","ch","oo","se","ha","san"

- Polish:
  - Substrings: “ń","ka","ski","ó","no","cz","ko","zga"

- Portuguese:
  - Substrings: “ń","ka","ski","ó","no","cz","ko","zga"

- Russian:
  - Substrings:
    “ov”,"nov","ev","ivch","sky","tov","iev","ich","cev","in","ff","hin",""

- Scottish:
  - Substrings: “aw","son","las","ald","ht","hy","ant","at"

- Spanish:
  - Substrings: “ó”,"lix","oj","é","rez","lla","cal","que","res","aya","ey","ez",'á','í'

- Spanish:
    - Substrings: “im","hao","han","ly","ach","ch"

Implementation: Found in predictor.py

Experimental Results:

- Testing for correct classification in only the provided samples in “**surnames-dev.csv**”
    - Results:
        - Accuracy = 0.25234318673395817

```
Accuracy: 0.25234318673395817
```

- Testing for correct classification in last names of musicians provided in “**composer-s.txt**”
    - Results:
        - Accuracy = 0.3

```
Accuracy: 0.3
```

    - From “**predictions.txt**”:

Name	Actual	Predicted
BelÃ©n	Spanish	Portuguese
Cunningham	Scottish	Korean
Ji	Korean	Russian
GrÃ¶nemeyer	German	Russian
LÃ©vy	French	Portuguese
Sawano	Japanese	Japanese
Yue	Chinese	Russian
Karadimos	Greek	Greek
O'Riordan	Irish	Chinese
VIvaldi	Italian	Italian

Conclusions:

- Based on the data provided for testing, the surnames of composers, seems that the classification based naively in some fragments that might be found in every language has a good accuracy.

- Good identification patterns like ‘mos’, ‘di’, and “o’” can make very obvious if a surname comes from a language, this, at least seems to show that there is a unique structure in Greek, Italian, and Irish that was correctly identified from the sample data in “**surnames-dev.csv**”.

- Comment: Currently the predictor builds a list of the possible languages from which a surname might come from, then the predictor chooses randomly any one of those options then checks if the country choose among the options if found in the list of languages from the original file, basically language choose against the languages the surname comes from. This is not necessarily the best idea, baye’s conditional probability could help to make more realistic choices.

- For the languages where not too many samples were available, like Portuguese it seems obvious there will be a lot of surnames that might get misidentified for coming from other languages. This means languages like Scottish, Vietnamese, as well as others, even including Irish are predicted to have their worst predictive power.

- Comment: Having a bad predictive power, because lack of examples doesn’t mean that there is no way any surname will get classified as coming from that country, for example there are few examples for Irish, but if anything starts with “ O’ ” it will likely get classified as Irish.

- On the opposite hand, the languages with best predictive power are those at which I have lots of examples, like Chinese and Russian.

- The implemented predictor fails if special characters are included like “@“ because it subtracts understating, for example the predictor does not interchange @ as “a” as people do in their head.

Appendix (Python 3.8):

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Aug 27 17:52:13 2020

@author: oscargalindo
"""

import random

def is_arabic(name):

strings=['fa','eif','ha','al','ba','ou','uo','ui','sar','dd','ad','ha','ni','nem','ki''zz','az','azz','am','sha
','ai','ouf']
    name = name.lower()
    for s in strings:
```

```
        if s in name:
            return True
    return False

def is_chinese(name):
    strings=['ao','oa','wa','jin','qi','dai','Tz','ai','ia','an','xi','oo','ang','ni','in','sh']
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_czech(name):

strings=['ka','ff','sek','wa','sch','deh','ss','jj','ik','pp','vl','ski','ze','rich','cova','tak','ovsky','warz
','schm','ich','witz','pp']
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_dutch(name):
    strings=['ee','jer','aye','oo','gg','ee','aa','ker','ss','nn','kk','ke']
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_english(name):

strings=['ll','oo','wood','ers','bb','ss','way','ee','dow','ach','gg','van','tt','pp','down','ross','ver',
'ker','ren','ran','owe','mc']
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_french(name):

strings=['le','ss','and','aux','oli','vage','eux','tit','ieu','nna','bon','boi','dú','neau','eau','oy','ias','
quet','reau','hur','agne']
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_german(name):
    strings=['tt','ber','oh','eier','ich','altz','ieck','enz','hol','uh','aun','del','ō','eis','mm','kl','aub']
    name = name.lower()
    for s in strings:
```

```
        if s in name:
            return True
    return False

def is_greek(name):
    strings=['ous','kos','nos','mos','los','is','as']
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_irish(name):
    strings=["o'",'ach',"ley","lly",'all','an','mac','oi','inn','ald']
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_italian(name):

strings=["eri","ari","icci","one","chi","ggi","gio","à","tti","di","gari","fin","lli","to","oli","dia","zzi",
"Scor","ini","ecce","acco","ani"]
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_japanese(name):

strings=['shi','oku','ida','iko','aka','tsu','awa','aya','ishi','uno','ima','chi','uya','hara','zak','aki','ji
','uch']
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_korean(name):
    strings=["mo","ch","oo","se","ha","san"]
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_polish(name):
    strings=["ń","ka","ski","ó","no","cz","ko","zga"]
    name = name.lower()
    for s in strings:
        if s in name:
            return True
```

```
    return False

def is_portuguese(name):
    strings=["inho","õ","lho","ã","ro","é"]
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_russian(name):
    strings=["ov","nov","ev","ivch","sky","tov","iev","ich","cev","in","ff","hin",""]
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_scottish(name):
    strings=["aw","son","las","ald","ht","hy","ant","at"]
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_spanish(name):
    strings=["ó","lix","oj","é","rez","lla","cal","que","res","aya","ey","ez",'á','í']
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def is_vietnamese(name):
    strings=["im","hao","han","ly","ach","ch"]
    name = name.lower()
    for s in strings:
        if s in name:
            return True
    return False

def get_origins(names):
    origin = []
    for name in names:
        options = []
        if is_arabic(name): options.append('Arabic')
        if is_chinese(name): options.append('Chinese')
        if is_czech(name): options.append('Czech')
        if is_dutch(name): options.append('Dutch')
        if is_english(name): options.append('English')
        if is_french(name): options.append('French')
        if is_german(name): options.append('German')
        if is_greek(name): options.append('Greek')
```

```
        if is_irish(name): options.append('Irish')
        if is_italian(name): options.append("Italian")
        if is_japanese(name): options.append("Japanese")
        if is_korean(name): options.append('Korean')
        if is_polish(name): options.append('Polish')
        if is_portuguese(name): options.append('Portuguese')
        if is_russian(name): options.append('Russian')
        if is_scottish(name): options.append('Scottish')
        if is_spanish(name): options.append("Spanish")
        if is_vietnamese(name): options.append("Vietnamese")
        if len(options) > 0:
            i = random.randint(0, len(options)-1)
            origin.append([name,options[i]])
        else:
            o =
['Arabic','Chinese','Czech','Dutch','English','French','German','Greek','Irish','Italian','Ja-
panese','Korean','Polish','Portuguese','Russian','Scottish','Spanish','Vietnamese']
            i = random.randint(0,len(o)-1)
            origin.append([name,options[i]])
    return origin

def process_file(src):
    lines = open(src).readlines()
    languages = dict()
    for line in lines:
        if 'To The First' in line: continue
        line = line.strip("\n")
        line = line.strip('"')
        line = line.strip('"')
        s = line.split(",")
        if "Jevolojnov" in line:
            s.remove('"')
            print(s)
        if not s[1] in languages:
            languages[s[1]] = []
        languages[s[1]].append(s[0])
    return languages

def process_file_names(src):
    lines = open(src).readlines()
    names = dict()
    for line in lines:
        if 'To The First' in line: continue
        line = line.strip("\n")
        line = line.strip('"')
        line = line.strip('"')
        s = line.split(",")
        if "Jevolojnov" in line:
            s.remove('"')
            print(s)
        if not s[0] in names:
            names[s[0]] = []
        if s[1] in names[s[0]]:
            continue
```

```
        else:
            names[s[0]].append(s[1])
    return names

def calculate_precision(names,origins):
    correct = 0
    for e in origins:
        if e[1] in names[e[0]]:
            correct+=1
    print("Accuracy:",correct/len(names))

def write_predictions(src,true,predictions):
    f = open(src+"prediction.txt","w")
    f.write("Name"+"\t"+"Actual"+"\t"+"Predicted\n")
    for e in predictions:
        # row = e[0]+"\t"+true[e[0]]"\t"+e[1]+"\n"
        f.write(e[0]+"\t"+true[e[0]][0]+"\t"+e[1]+"\n")
    f.close()

if __name__ == "__main__":
    src = '/Users/oscargalindo/Desktop/Classes/CS 5319/Surnames Assignment/'
    countries = process_file(src+'surnames-dev.csv')
    names = process_file_names(src+'surnames-dev.csv')
    n = []
    for k in names.keys():
        n.append(k)
    origins = get_origins(n)
    calculate_precision(names, origins)
    ##############Musicians###################
    musicians = process_file_names(src+'composers.txt')
    n = []
    for k in musicians.keys():
        n.append(k)
    origins = get_origins(n)
    calculate_precision(musicians, origins)
    write_predictions(src, musicians, origins)
```