Predicting Star Trek Characters with Naive Bayes

2020-05-03 BY GENE

I have been reading "Data Science from Scratch" and porting the python code to perl. In this episode, we use Naive Bayes to predict whether the speaker of a line is either Kirk, Spock or McCoy, with my new, work-in-progress data science library. tl;dr: is-character.pl

I will not dive into an explanation of the Naive Bayes classifier other than to say that it analyzes a set of words to see if they fall into predetermined categories like "spam" and "not spam" ("ham"). There are many many explanations already out there!

The data science library that I am using has "spam" and "ham" baked in, so we will consider the person of interest as spam and ham as everyone else.

Before proceeding, we need the spoken lines. I have collected these and uploaded the zip file to github. Un-archive them on your local machine and update the $path line in the program parameter section in the code, below.

Okay on with the example! First up is the standard perl preamble and library module usage:

```
#!/usr/bin/env perl
use strict;
use warnings;

use Data::MachineLearning::Elements;
use File::Slurper qw(read_text);
use Lingua::EN::Sentence qw(get_sentences);
```

Next are the crucial program parameters and their defaults:

```
# Who are we interested in?
my $who = shift || 'kirk';
# How big should the training set be as a percent of the total?
my $split = shift || 0.33;
# Probability threshold (confidence) that the person of interest said it
my $threshold = shift || 0.7;
# Provide a custom statement to process
my $statement = shift || 'To be or not to be. That is the question.';

my @statements = (
    'Shall we pick some flowers, Doctor?',                    # Kirk
    'Vulcan has no moon, Miss Uhura.',                        # Spock
    'Is that how you get girls to like you, by bribing them?',  # McCoy
    'Give me warp nine, Scotty.',                             # Fake Kirk
    'That is highly illogical.',                              # Fake Spoc
    "He's dead, Jim.",                                        # Fake McCo
    $statement
);

# Unziped Star Trek scripts in this location
my $path = $ENV{HOME} . '/Documents/lit/Kirk-Spock-McCoy/';
```

Here we have 1. who we are interested in, 2. how big the training set should be as a percentage of the total data size, 3. our threshold defining our confidence that a sentence is spoken by who we are interested in, 4. & 5. the statements to analyze, and 6. the path where the Star Trek script files live. Change this path to your unzipped location.

So we need to gather and process the sentences of each character, classifying them as either “spam” (the person of interest) or “not spam” (everyone else):

```
print "Gathering messages...\n";
my @messages;

# Process the lines of each file
for my $i (qw(kirk spock mccoy)) {
    my $file = $path . $i . '.txt';

    my $content   = read_text($file);
    my $sentences = get_sentences($content);

    for my $sentence (@$sentences) {
        $sentence =~ s/\(.+?\)//; # Remove action instructions

        next if $sentence =~ /\(/; # Skip broken actions

        $sentence =~ s/\n+/ /g; # Make the sentence a single line
        $sentence =~ s/^\s*//; # Trim whitespace
        $sentence =~ s/\s*$//; # Trim whitespace

        next unless $sentence =~ /\w/;
        #print $sentence, "\n\n";

        # The processed messages are a list of 2-key hashrefs
        push @messages, { text => $sentence, is_spam => $i eq $who ? 1 : 0
    }
}
```

With the messages in hand we apply machine learning!

```
# Invoke the data science library
my $ml = Data::MachineLearning::Elements->new;

my ($train, $test) = $ml->split_data($split, @messages);

print "Training on messages...\n";
$ml->train(@$train);
```

After the classifier has been trained we use it to predict a handful of statements (given above in the program parameters section).

```
my $name = ucfirst $who;

print "$name said ", $ml->spam_messages, " sentences.\n";
print "Not-$name said ", $ml->ham_messages, " sentences.\n";

print "Probability that $name said,\n";
for my $text (@statements) {
    next unless $text;
    my $prediction = $ml->nb_predict($text);
    printf qq/\t%.4f = "%s"\n/, $prediction, $text;
}
```

Ok. That's a bit of insight into the results that can be produced. But how accurate is our classifier? Here is the code to compute that:

```
print "Computing accuracy, etc...\n";
my ($tp, $fp, $fn, $tn) = (0,0,0,0);
my %confusion_matrix;

for my $i (@$test) {
    my $predicted = $ml->nb_predict($i->{text});

    my $true_pos  =  $i->{is_spam} && $predicted >= $threshold ? 1 : 0;
    my $false_pos = !$i->{is_spam} && $predicted >= $threshold ? 1 : 0;
    my $false_neg =  $i->{is_spam} && $predicted <  $threshold ? 1 : 0;
    my $true_neg  = !$i->{is_spam} && $predicted <  $threshold ? 1 : 0;

    $confusion_matrix{"$true_pos,$false_pos,$false_neg,$true_neg"}++;

    $tp += $true_pos;
    $fp += $false_pos;
    $fn += $false_neg;
    $tn += $true_neg;
}

print "Confusion matrix (true_pos,false_pos,false_neg,true_neg):\n",
    join("\n", map { "\t$_ => $confusion_matrix{$_}" } sort keys %confusion
    "\n";
printf "Accuracy = %.4f\nPrecision = %.4f\nRecall = %.4f\nf1_score = %.4f\n
    $ml->accuracy($tp, $fp, $fn, $tn),
    $ml->precision($tp, $fp, $fn, $tn),
    $ml->recall($tp, $fp, $fn, $tn),
    $ml->f1_score($tp, $fp, $fn, $tn);
```

The results for Kirk are:

```
> time perl -Ilib is-character.pl kirk

Gathering messages...
Training on messages...
Kirk said 5587 sentences.
Not-Kirk said 3994 sentences.
Probability that Kirk said,
        0.7706 = "Shall we pick some flowers, Doctor?"
        0.2317 = "Vulcan has no moon, Miss Uhura."
        0.6189 = "Is that how you get girls to like you, by bribing them?"
        0.9611 = "Give me warp nine, Scotty."
        0.2352 = "That is highly illogical."
        0.0466 = "He's dead, Jim."
        0.2176 = "To be or not to be. That is the question."
Computing accuracy, etc...
Confusion matrix (true_pos,false_pos,false_neg,true_neg):
        0,0,0,1 => 5873
        0,0,1,0 => 3939
        0,1,0,0 => 2491
        1,0,0,0 => 7151
Accuracy = 0.6695
Precision = 0.7417
Recall = 0.6448
f1_score = 0.6899

real    5m50.978s
```

The “f1_score” is probably the best metric here. And the classifier gets it right approximately 69% of the time. Not the most fabulous results. By the way, the number of sentences said by Kirk are based on the size of the *training data* (split at 33%) – not the total number of things he said. Also, the program takes about six minutes to run on my older Macbook… YMMV.

For Spock we get:

```
Spock said 2620 sentences.
Not-Spock said 6961 sentences.
Probability that Spock said,
```

```
        0.5974 = "Shall we pick some flowers, Doctor?"
        0.7845 = "Vulcan has no moon, Miss Uhura."
        0.0019 = "Is that how you get girls to like you, by bribing them?"
        0.0051 = "Give me warp nine, Scotty."
        0.9170 = "That is highly illogical."
        0.0161 = "He's dead, Jim."
        0.5952 = "To be or not to be. That is the question."
Computing accuracy, etc...
Confusion matrix (true_pos,false_pos,false_neg,true_neg):
        0,0,0,1 => 12831
        0,0,1,0 => 2858
        0,1,0,0 => 1400
        1,0,0,0 => 2365
Accuracy = 0.7811
Precision = 0.6282
Recall = 0.4528
f1_score = 0.5263
```

For McCoy this is:

```
Mccoy said 1491 sentences.
Not-Mccoy said 8090 sentences.
Probability that Mccoy said,
        0.0342 = "Shall we pick some flowers, Doctor?"
        0.0195 = "Vulcan has no moon, Miss Uhura."
        0.9982 = "Is that how you get girls to like you, by bribing them?"
        0.0015 = "Give me warp nine, Scotty."
        0.0713 = "That is highly illogical."
        0.9791 = "He's dead, Jim."
        0.1675 = "To be or not to be. That is the question."
Computing accuracy, etc...
Confusion matrix (true_pos,false_pos,false_neg,true_neg):
        0,0,0,1 => 15237
        0,0,1,0 => 2154
        0,1,0,0 => 1193
        1,0,0,0 => 870
Accuracy = 0.8280
Precision = 0.4217
Recall = 0.2877
f1_score = 0.3420
```

Not enough unique data for Bones to make an impact.

FILED UNDER: SOFTWARE
TAGGED WITH: MACHINE LEARNING, PERL, STAR TREK

Epistemologist-at-large

^ Top