Naive Bayes Classification

Practical exercises

Leon Derczynski

Exercise format

- Fill in the blanks (___)
  - Add your solution
  - Uncomment the line to run it

- The problem is broken into different parts
- Parts separated by sys.exit() calls in the file
- Migrate this to a Python notebook

- You may accidentally develop NTLK / Py skills
- Code is incomplete bug-free

Predicting gender from name

- Use names.py
- Complete the function “gender_features” to convert an input word to a last_letter representation

```
>>> gender_features('Shrek')
{'last_letter': 'k'}
```

Predicting gender from name

- Next, we need to load up a set of names

```
>>> from nltk.corpus import names
>>> labeled_names = ([(name, 'male') for name in names.words('male.txt')] +
... [(name, 'female') for name in names.words('female.txt')])
>>> import random
>>> random.shuffle(labeled_names)
```

- The first part gives a list of all male then all female
  - Partitioning the dataset will give mostly just one class
  - Solution: shuffle the data after loading, but before splitting
- random.shuffle() mixes up a list in-place (i.e. returns nothing)
  - Use a seed – else, multiple runs will mean you see test data

Predicting gender from name

- Let's add more features
- Upgrade the gender_features function so it returns both last letter and last letter pair

```
>>> gender_features('Shrek')
{'last_letter': 'k', 'last_2_letters': 'ek'}
```

Augmenting for Danish

- How well does this do on Danish names?
  - Trine vs. Bjarne
- Make new Danish names sets, dk_names & dk_features
  - Load names_danish_f.lst, names_danish_m.list
    - See labeled_names setup
    - Next, convert to features; see featuresets setup
  - How does the model perform on this data?
    - nltk.classify.accuracy
- Integrate the Danish names in the model, and re-evaluate
  - Merge all the names into one feature set, before shuffle and split
  - What's the problem with comparing these scores?

More features!

- Does adding more data solve our problem?
  - How are Trine and Bjarne represented?
  - How about Diane?
- Maybe we can start to capture more features

```
>>> gender_features2('John')
{'count(j)': 1, 'has(d)': False, 'count(b)': 0, …}
```

- Re-evaluate, using new code
- Is it better?

Error Analysis

- Productive method for refining features
- First, select a development set
  - Contains corpus data for creating model
  - Subdivided into training set and dev-test set

```
>>> train_names = labeled_names[1500:]
>>> devtest_names = labeled_names[500:1500]
>>> test_names = labeled_names[:500]
```

- New feature set:

```
…
>>> devtest_set = [(gender_features(n), gender) for (n,
gender) in devtest_names]
…
```

Error Analysis

- Generate a list of the errors that the classifier makes when predicting name genders
- Extract the misclassified data:

```
for (name, tag) in devtest_names:
   guess = classifier.classify(gender_features(name))
   if guess != tag:
```

- Next, we can take a look at the errors
  - Look in the code

```
correct=female    guess=male      name=Abigail
correct=female    guess=male      name=Katheryn
correct=female    guess=male      name=Kathryn
correct=male      guess=female    name=Aldrich
correct=male      guess=female    name=Mitch
```

Biases

- What social biases are in this system?
- What harm could its widespread deployment cause?
- Is this classification task a useful one, or one that we should be doing?

Feature engineering

- Write your own gender_features extraction function
  - Tune it using the devtest set
  - How high can you get?
- When you're ready, test it on the test data
  - You cannot go back once you've done this
- Questions to consider:
  - What features help?
  - Which ones hurt?
  - What happens when you have too many?