Elementary text classification lecture notes

Linguistics 165, Professor Roger Levy

23 January 2015

1. The problem.

- Go to http://www.yahoo.com, the fourth-most-visited website in the world. The left sidebar is a number of categories. Each of these categories in turn has subcategories that you can see if you click through. There are lots of webpages in each category and subcategory. In the old days of the Web, Yahoo constructed these hierarchies manually. This might still be somewhat viable because new webpages Yahoo deems worthy of being the hierarchy may appear at a relatively modest rate.

- Now go to http://news.google.com. On the left sidebar there is a similar list of categories for news items (World/U.S./Business/Technology/...), and each one has many subcategories. Thousands of news stories per day have to get slotted into these categories. There’s no way we’d want these categorizations to be done by hand, and Google doesn’t: they do it with automatic text classification.


2. The formal problem specification. Assume that there is a collection of possible documents \( \{d\} \) in the world, and each one falls into one of a predefined, finite set of categories \( C = \{c_1, c_2, \ldots, c_M\} \). We’ll use \( \langle d, c_i \rangle \) to denote a document \( d \) and its category \( c_i \). In some cases, we may know the category \( c_i \) for document \( d_i \), but in other cases we may not. **Goal:** we want a method of guessing a category \( c \) for a document \( d \). This method is called a classifier.

3. Getting familiar by example. Let’s look at the NLTK movie reviews corpus!

```python
>>> import textwrap  # helps format some long strings in this document
>>> from textwrap import fill
>>> import nltk
>>> from nltk.corpus import movie_reviews
```

A useful note: the `dir()` function gives you a list of fields and methods for a Python object.
```python
>>> dir(movie_reviews)[0:2]
['_LazyCorpusLoader__args', '_LazyCorpusLoader__kwargs']
```

For brevity I’ve shown just the first two entries, but further inspection will show that it has a `categories` field. The `type()` function reveals the type of this field to be an instance method:

```python
>>> type(movie_reviews.categories)
<class 'method'>
```

so we call the method to find out the text categories for this corpus:

```python
>>> movie_reviews.categories()
['neg', 'pos']
```

The structure of `movie_reviews` is such that you can pull out the IDs of the positive reviews with `movie_reviews.fileids('pos')` and the IDs of the negative reviews with `movie_reviews.fileids('neg')`. For example:

```python
>>> movie_reviews.fileids('neg')[0]
'neg/cv000_29416.txt'
```

The corpus is balanced between positive and negative reviews:

```python
>>> len(movie_reviews.fileids('pos'))
1000
>>> len(movie_reviews.fileids('neg'))
1000
```

You can access the text of a review as a tokenized list of words with `movie_reviews(words[fileid])` where `fileid` is the file in question:

```python
>>> movie_reviews.words(movie_reviews.fileids('neg')[0])
['plot', ':', 'two', 'teen', 'couples', 'go', 'to', ...]
```

We’ll continue on with examples in code from http://idiom.ucsd.edu/~rlevy/teaching/2015winter/lign165/lectures/lecture7/movie_reviews.py.