Improving a System – ExpertSystem Search

CS410 Course Project, Fall 2020

pmanden2@illinois.edu

Objectives

From *Improving a System – ExpertSystem Search*

1

- **Identifying faculty directory pages:** First, we need to identify the pages from where faculty web-pages can be mined. In MP2.1, we used faculty directory pages as the starting point to find faculty webpages. So, given a university website, can we automatically identify the directory pages? This can be posed as a classification task, i.e. classify a URL into a directory page vs. non-directory page. We have a huge resource of directory page URLs available in the sign-up sheet. These can be the "positive" examples. You can get a list of some random URLs online or crawl some other pages to get URLs(e.g. other URLs on the university websites, product websites, news sites,etc.). These would be the "negative" examples.

2

- Identifying faculty webpage URLs: Next, we need to extract the faculty webpages from the directory pages. This can again be posed as a classification task. Given a URL, can we identify whether it is a faculty webpage or not? We have a huge resource of faculty webpage URLs (available under MP2.3 on Coursera). These would be the "positive" examples. You can get a list of some random URLs online or crawl some other pages to get URLs (e.g. other URLs on the university websites, product websites, news sites, etc.) to get the "negative" labels.

Objectives

- Given a URL classify it as directory vs non-directory page
- Given a URL classify it as faculty or non-faculty page

[This project was done by myself as a one person team]

Naïve Bayes classifier

$$\hat{P}(x_i \mid \omega_j) = \frac{\sum tf(x_i, d \in \omega_j) + \alpha}{\sum N_{d \in \omega j} + \alpha \cdot V}$$

where

- x_i: A word from the feature vector $\mathbf{x}$ of a particular sample.
- $\sum tf(x_i, d \in \omega_j)$: The sum of raw term frequencies of word x_i from all documents in the training sample that belong to class ω_j.
- $\sum N_{d \in \omega j}$: The sum of all term frequencies in the training dataset for class ω_j.
- α: An additive smoothing parameter ($\alpha = 1$ for Laplace smoothing).
- V: The size of the vocabulary (number of different words in the training set).

The class-conditional probability of encountering the text $\mathbf{x}$ can be calculated as the product from the likelihoods of the individual words (under the *naive* assumption of conditional independence).

$$P(\mathbf{x} \mid \omega_j) = P(x_1 \mid \omega_j) \cdot P(x_2 \mid \omega_j) \cdot \ldots \cdot P(x_n \mid \omega_j) = \prod_{i=1}^{m} P(x_i \mid \omega_j)$$

Credit - https://sebastianraschka.com/Articles/2014_naive_bayes_1.html

Text representation – Bag of words

Samples in Class	Terms														
	cs	illinois	edu	about	people	all-faculty	csd	cmu	directory	faculty	stanford	faculty-staff	uic	uchicago	
https://cs.illinois.edu/about/people/all-faculty	1	1	1	1	1	1									
https://www.csd.cs.cmu.edu/directory/faculty	1		1				1	1	1	1					
https://www.cs.stanford.edu/directory/faculty	1		1						1	1	1				
https://cs.uic.edu/faculty-staff/faculty/	1		1							1		1	1		
https://www.cs.uchicago.edu/people/faculty/	1		1		1					1				1	
Term frequency	5	1	5	1	2	1	1	1	2	4	1	1	1	1	**27**

N = 27

V = {cs, illinois, edu, about, people, all-faculty, csd, cmu, directory, faculty, stanford, faculty-staff, uic, uchicago} = 14

P(cs|class) = (5 +1)/(27+14) = 0.146

P(illinois|class) = (1 +1)/(27+14) = 0.049

P(edu|class) = (5 +1)/(27+14) = 0.146

P(cs.illinois.ed|class) = 0.146*0.048*0.146 = 0.001023168

Implementation

- 2 Classifiers built
  - Directory link classifier
  - Faculty link classifier
- Directory link classifier Training data
  - Positive Samples collected from MP2.1 signup spreadsheet (as suggested)
    - 900 Samples are available (in file ‘directory-positives.txt’ in source code)
  - Negative Samples
    - A scraper utility was written to collect links from university webpages, which excluded all links with keywords such as ‘directory’, ‘staff’ etc. to ensure negative samples
    - 6592 samples are available (in file ‘directory-negatives.txt’ in source code)

Implementation

- Faculty link classifier Training data
  - Positive samples
    - Faculty pages from MP2.3 data on Coursera was used as suggested
    - 16492 samples are available (in file ‘faculty-pages-positives.txt’)
  - Negative samples
    - The same scraper was used to generate links from university websites. Links with keywords such as ‘faculty’, ‘staff’ are removed to ensure negative links
    - 6592 samples available (in file ‘faculty-pages-negatives.txt’)

Source code – (Core file)

- Classifier.py
  - Implements Naïve Bayes Classifier in ‘class naive_bayes_classifier’
  - Input
    - Name of file that contains positive samples
    - Name of file that contains negative samples
    - Number of samples to be used for training
  - ‘initialize_classifier’ method
    - Loads the samples from the specified files
    - Calculates and saves term document matrix, term frequency, no of terms, total counts for each terms for both positive and negative samples
  - ‘classify’ method
    - Accepts a url
    - Calculates probability of the words in url with laplace smoothing for both positive and negative classes
    - Returns True if the positive class probability is > negative class probability
- It can be run to test the code independently
  - To test, Run – ‘python classify.py’, tests model accuracy and shows results

Model Accuracy

- Directory Classifier (with 800 samples, 100 test data)
  - Precision 0.96
  - Recall 0.94
  - F1 score 0.94
- Faculty Classifier (with 6000 samples, 300 test data)
  - Precision 0.99%
  - Recall 0.98
  - F1 score 0.99

Source Code - Utilities

- scraper.py : Scrapes all universities listed at
  - http://doors.stanford.edu/~sr/universities.html
  - 1,088 universities available
  - Identify 10 links from the home page of each university
  - Removes all links with key words that identify directory pages, such as ‘directory’, etc.
  - Removes all links with key words that identify faculty pages, such as ‘faculty’ etc.
  - Generates a list that is used as negative samples for both classifiers
- Run – python scraper.py
  - Will take 3-4 hours to run. (Change ‘no_universities_to_scrape’ line 137 to test on limited number of universities)

Source Code – UI, Data files

- ExpertSearch app UI was modified to test URLs
  - UI changes are only for testing & and not relevant to improving the system!
- templates/index.html
  - Modified to support a new pull down menu
- static/index.js
  - Java script code changes to interface with backend
- Data files
  - directory_positives.txt (Directory positive samples)
  - directory_negatives.txt (Directory negative samples)
  - faculty_positives.txt (Faculty positive samples)
  - faculty_negatives.txt (Faculty negative samples)

Running the Application

- To run ExpertSearch App run
  - python server.py
  - Open a browser and point to http://localhost:8095/

- Demo! – Play ProjectDemo.mp4