Fall 2018 CS 410

Implementing Generative Feature Language Models for Mining Implicit Features

Hannah Wilder and Norbert Freundlich

Project Overview

Goal: generalize and optimize Python code originally written by Shubhra Kanti Karmaker Santu to perform algorithm presented in Generative Feature Language Models for Mining Implicit Features from Customer Reviews

Tasks:

- Build classes and methods to perform paper algorithm
- Verify code generates same results
- Create installable library from code
- Thoroughly document code and methodology

Process

1. Review Santu's original code
2. Research relevant topics
3. Write our own implementation
4. Write and execute unit tests
5. Optimize where possible
6. Document code

Topic Background

What is an explicit feature?

A topic/term that is mentioned directly in a section of text.
ex. “I like the size of the phone”

What is an implicit feature?

A topic/term that is referred to or described indirectly in a section of text.
ex. “The phone fits nicely in my pocket”

Why are implicit features important?

GFLM paper estimate on manually tagged data suggests ~20% of customer reviews may contain implicit feature mentions

Feature tagging can be used in sentiment analysis to help pinpoint cause of dissatisfaction

Can be used for product and competitor comparisons

Code Structure

fm (wrapper class)

Handles user interface and logistics - calls all below functionality in the correct order with given parameters

parse_and_model

Contains functions to read in files, format data and calculate explicit feature and background models

em_base

Skeleton for the expectation maximization algorithm implementations

em_original

Santu's original for loop implementation for testing purposes

em_vector

Vector implementation using sparse matrices, looped over sections

em_vector_by_feature

Optimized vector implementation using sparse matrices, looped over features

gflm_tagger

Contains functions to calculate GFLM word and sentence from EM algorithm results

Algorithm Input

- List of explicit features to search for
  Ex. size, battery, screen

- Data set to search and tag
  Pre-annotated or raw text

Preparation for EM Algorithm

- Input data is processed and formatted
  Can include stopword removal and lemmatization based on arguments

- Explicit feature and background models built from formatted input data

EM Vectorization

Goal: speed up EM while still conserving memory where possible

Methodology:

- Converted E and M step calculations to matrix operations (calculations detailed in appendix)
- Stored data in scipy sparse matrices where possible to save space
- Ordered matrix calculations to maintain matrix sparseness throughout operations
- Still left one loop over features to avoid 3D arrays (due to time constraints)

Algorithm Output

- Mapping between text sections and explicit feature mentions

- Mapping between text sections and implicit feature mentions
  - One set based on GFLM - Word
  - Second set based on GFLM - Sentence

- EM algorithm hidden parameters

Unit Testing

- Used to verify code operations
- ‘Toy’ datasets used for initial/simple checks
- More complicated and complete tests built by generating data from Santu’s original implementation and comparing results

Project Demo

Uses iPod annotated dataset -
% of explicit feature mentions randomly chosen to be removed to simulate implicit feature mentions

Simulated implicit feature

Review title

Explicit feature mention

```
[t] ipod
##I like the compact ipod and the features it offered
size[@][u]##It is handy to carry around because of the  and easy to store
##The light weight also makes it easy to move with
##It works well and I have had no problems with it
[t] iPod is Awesome
##This is my second iPod
##My first was a "mini" which the nano makes look like a "jumbo"
sound[@]##It's very lightweight, sound quality is typical of these devices
battery[@]##The battery life is outstanding (again, compared to the mini)
battery[@]##I've only had it for a month, but the battery so far is lasting over 8 hours
##I haven't completely run it until it is dead yet, so I don't know how long it will really last
##Awesome!
```

Remainder of project demo in Jupyter notebook

(Can also be found in GitHub project - tutorial.ipynb)

Opportunities for Future Improvement

- Multiple feature section coverage recommendations
- Better explicit feature synonym handling
- Built-in evaluation functions
- Built-in parameter tuning for lamda background and GFLM threshold values
- Fully vectorized EM implementation

Team Member Contributions

- Both team members contributed extensively to both the code base and documentation
- Collaborated to develop and review code
- Primary contributions
  - Norbert Freundlich
    - Vectorized EM implementation
    - Class structure
    - Package construction (for installation)
    - Optimizations for file parse and explicit model construction
    - Code tutorial
  - Hannah Wilder
    - Explicit model construction
    - GFLM word/sentence
    - Optimizations for vectorized EM implementation
    - Presentation slides

Available Documentation

- Jupyter notebook tutorial
- Github README
- Code comments
- HTML code docs
- Slides

Thank you!

Appendix

Vectorization Definitions

m = number of sections
v = number of unique words (vocabulary size, $|V|$)
k = number of explicit features

w_h = word h
f_i = feature i
s_j = section j

ew = element-wise operation with broadcasting if necessary
λ_B = user-determined background model probability

$$\underset{\substack{m\times v\\ sparse}}{R} = \begin{bmatrix} count(s_1, w_1) & ... & count(s1, w_v) \\ count(s_2, w_1) & ... & count(s2, w_v) \\ \vdots & ... & \vdots \\ count(s_m, w_1) & .. & count(s_m, w_v) \end{bmatrix}$$

$$\underset{\substack{m\times v\\ sparse}}{R_B} = \begin{bmatrix} b(s_1, w_1) & ... & b(s1, w_v) \\ b(s_2, w_1) & ... & b(s2, w_v) \\ \vdots & ... & \vdots \\ b(s_m, w_1) > 0 & .. & b(s_m, w_v) \end{bmatrix}$$

$$b(s_j, w_h) = if\ count(s_j, w_h) > 0\ then\ 1\ else\ 0$$

$$\underset{\substack{v\times k\\ dense}}{T} = \begin{bmatrix} p(w_1|\gamma_1) & ... & p(w_1|\gamma_k) \\ p(w_2|\gamma_1) & ... & p(w_2|\gamma_k) \\ \vdots & ... & \vdots \\ p(w_v|\gamma_1) & .. & p(w_v|\gamma_k) \end{bmatrix}$$

$$\underset{\substack{v\times 1\\ dense}}{B} = \begin{bmatrix} p(w_1|\gamma_B) \\ p(w_2|\gamma_B) \\ \vdots \\ p(w_v|\gamma_B) \end{bmatrix}$$

$$\underset{\substack{m\times k\\ dense}}{\pi} = \begin{bmatrix} \pi_{s_1,f_1} & ... & \pi_{s_1,f_k} \\ \pi_{s_2,f_1} & ... & \pi_{s2,f_k} \\ \vdots & ... & \vdots \\ \pi_{s_m,f_1} & .. & \pi_{s_m,f_k} \end{bmatrix}$$

$$\underset{\substack{m\times v\\ sparse}}{H_{f_{[i]}}} = \begin{bmatrix} p(z_{s_1,w_1} = f_i) & ... & p(z_{s_1,w_v} = f_i) \\ p(z_{s_2,w_1} = f_i) & ... & p(z_{s_2,w_v} = f_i) \\ \vdots & ... & \vdots \\ p(z_{s_m,w_1} = f_i) & .. & p(z_{s_m,w_v} = f_i) \end{bmatrix}$$

$$\underset{\substack{m\times v\\ sparse}}{H_B} = \begin{bmatrix} p(z_{s_1,w_1} = B) & ... & p(z_{s_1,w_v} = B) \\ p(z_{s_2,w_1} = B) & ... & p(z_{s_2,w_v} = B) \\ \vdots & ... & \vdots \\ p(z_{s_m,w_1} = B) & .. & p(z_{s_m,w_v} = B) \end{bmatrix}$$

E-step vectorization

1. $$P(z_{S,w} = f) = \frac{\boxed{\pi_{S,f}^{(n)} P(w|\gamma_f)}}{\boxed{\sum_{f'=1}^{k} \pi_{S,f'}^{(n)} P(w|\gamma_{f'})}}$$

1.1

1.2

2. $$P(z_{S,w} = B) = \frac{\boxed{\lambda_B P(w|\gamma_B)}}{\boxed{\lambda_B P(w|\gamma_B) + (1-\lambda_B)\sum_{f'=1}^{k} \pi_{S,f'}^{(n)} P(w|\gamma_{f'})}}$$

2.1

E-step

 # Calculates 2.1, only needs to be calculated once

 # (performed during EM initialization but still part of E-Step)

$H'_B = \lambda_B \; B \times_{ew} R_B$

For each f_i in 1 to k

 # Calculates 1.1

$H_{f_{[i]}} = (\pi_{[,i]} \; T^T_{[i,]}) \times_{ew} R_B$

 # Calculates 1.2

if i =1

$H_{f_{sum}} = H_{f_{[i]}}$

else

$H_{f_{sum}} \mathrel{+}= H_{f_{[i]}}$

For each f_i in 1 to k

 # Calculates 1

$H_{f_{[i]}} = H_{f_{[i]}} \times_{ew} H_{f_{sum}}{}^{-1_{ew}}$

 # Calculates 2

$H_B = H'_B \times_{ew} \left(H'_B + (1-\lambda_B) H_{f_{sum}} \right)^{-1_{ew}}$

M-step vectorization

$$3.\quad \pi_{S,f}^{(n+1)} = \frac{\boxed{\sum_{w \in V} c(w,S)(1 - P(z_{S,w} = B))P(z_{S,w} = f)}}{\boxed{\sum_{f'=1}^{k} \sum_{w \in V} c(w,S)(1 - P(z_{S,w} = B))P(z_{S,w} = f')}} \quad \begin{matrix} 3.1 \\ 3.2 \end{matrix}$$

M-step

For each f_i in 1 to k

 # Calculates 3.1

$$\pi_{[,i]} = \left(R \times_{ew} (1 -_{ew} H_B) \times_{ew} H_{f_{[i]}}\right) \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}_{v \times 1}$$

 # Calculates 3.2

if i =1

$\pi_{sum} = \pi_{[,i]}$

else

$\pi_{sum} += \pi_{[,i]}$

 # Calculates 3

$\pi = \pi \div_{ew} \pi_{sum}$