Discrete Mathematics

Programming Assignment 3

Naive Bayes Text Classifier

Shin Hong

27 Nov, 2019

Team

ITP 20002-01 (1 PM Class)

Team 31	김주현	김평강	송수근	이인석	최새암
Team 32	김영빈	손한나	윤수빈	이민재	정수산나
Team 33	김민희	김은종	오상진	안병웅	최진아
Team 34	김규림	박하윤	박지희	방선기	정예은
Team 35	김준형	서예지	서준표	신희주	이지훈
Team 36	김서예	김예진	이기현	유수정	최하현
Team 37	강나린	박예인	박은우	이혜림	한광영
Team 38	권지현	김지수	신지영	이현규	임성빈
Team 39	김기웅	남현지	최승아	황보연	

ITP 20002-02 (4 PM Class)

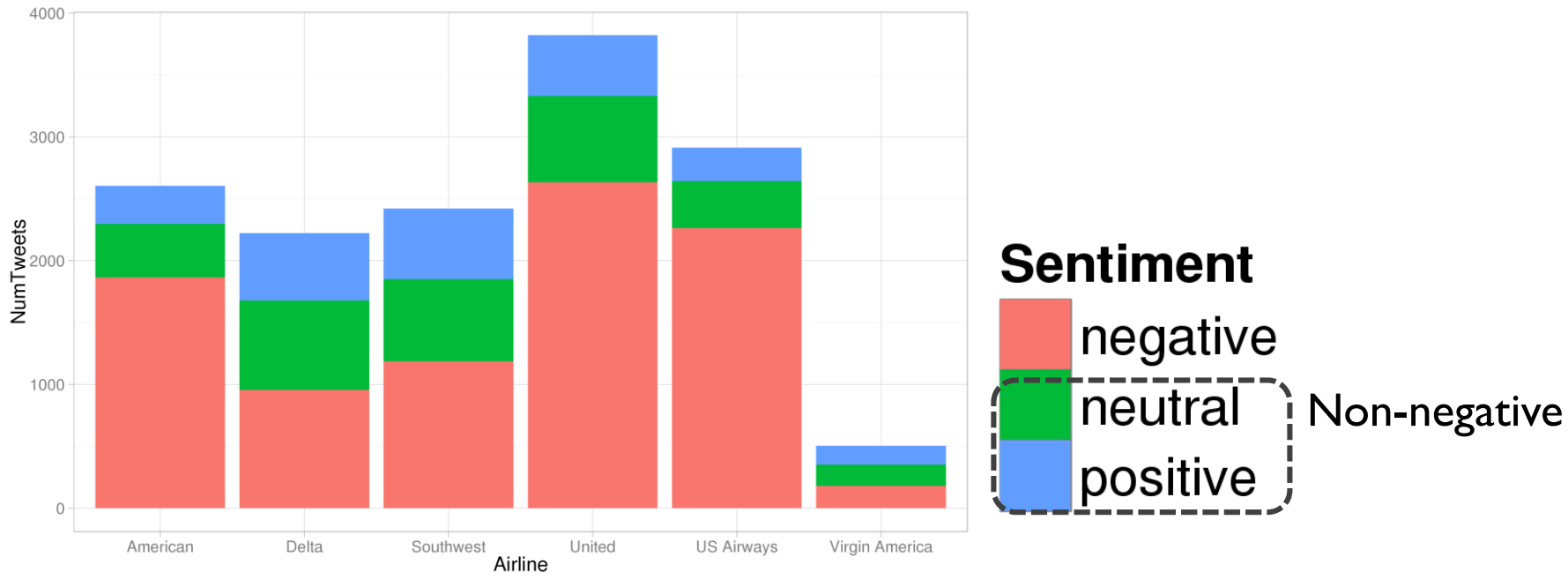
Team 31	김해린	서상원	양은영	이민정	임예찬
Team 32	금락운	김기훈	김미소	여대엽	임예원
Team 33	김예찬	김혜영	박주원	송하림	안제인
Team 34	문영은	서인아	전병운	지성민	홍순규
Team 35	Daniel	윤대영	장류미	진영인	최건영
Team 36	김유영	김영제	박상범	박수아	

Assignment Overview

- Write a C program for predicting whether the sentiment of a given sentence is negative or not
  - trainer: build a Naive Bayes model from the Twitter Airline dataset
  - predictor: read a sentence and returns either 'Negative' or 'Non-negative'
- Submission
  - deadline: 14 Dec (Sat), 11:59 PM
  - deliverables (via Hisnet)
    - source code files
    - write-up: up to 3 pages in A4
- Team: work with your team members to make one submission
- Starter code: https://github.com/hongshin/DiscreteMath/tree/pa3

Twitter US Airline Sentiment Dataset

- Kaggle URL: https://www.kaggle.com/crowdflower/twitter-airline-sentiment
- A sentiment analysis job about the problems of each major U.S. airline.
- Twitter data was scraped from February of 2015 and contributors were asked to first classify positive, negative, and neutral tweets, followed by categorizing negative reasons (such as "late flight" or "rude service").



Preprocessed Data

- Twitt messages are classified into Negative and Non-negative
  - total 14843 messages
  - airline names had been removed to reduce noise

- Train data
  - files
    - `test.negative.csv`: 9078 messages labelled as Negative
    - `test.non-negative.csv`: 5565 messages labelled as Non-Negative

- Test data
  - files
    - `test.negative.csv`: 100 messages labelled as Negative
    - `test.non-negative.csv`: 100 messages labelled as Non-Negative
  - these files must not be used for training

Naive Bayes Text Classifier

- Naïve Bayes is well known for its good performance in text classification, although it's based on super simple computation

- S: the message is Negative
- V: the message contains ‘poor'
- The probability that the message is Negative conditional on containing ‘poor' is:

$$P(S \mid V) = \frac{P(S,V)}{P(V)} = \frac{P(V \mid S)P(S)}{P(V \mid S)P(S)+P(V \mid \neg S)P(\neg S)}$$

- Once we assume $P(S) = P(\neg S)$,

$$P(S \mid V) = \frac{P(V \mid S)}{P(V \mid S) + P(V \mid \neg S)}$$

Bag of Word Model

- Words: $w_1, w_2, \ldots, w_n$
- X_i : the event that a message contains word w_i
- $P(X_i \mid S)$: the probability that a Negative message contains w_i
- $P(X_i \mid \neg S)$: the probability that a Non-negative message contains w_i
- $M = \{w_{m_1}, w_{m_2}, \ldots, w_{m_L}\}$: a message as a set of its containing words
- Assume that appearance of each word in a message is independent to each other:

$$P(S \mid M) = \frac{P(M \mid S)}{P(M \mid S) + P(M \mid \neg S)}$$

$$P(M \mid S) = P(X_{m_1} \mid S) \times P(X_{m_2} \mid S) \times \cdots \times P(X_{m_L} \mid S)$$

$$P(M \mid \neg S) = P(X_{m_1} \mid \neg S) \times P(X_{m_2} \mid \neg S) \times \cdots \times P(X_{m_L} \mid \neg S)$$

Rescaling and Smooting

- Rescaling with log function
  - $x > y$ iff $\log(x) > \log(y)$ for positive x and y
  - $\log(ab) = \log(a) + \log(b)$
  - $\log(P(M \mid S)) = \log(P(X_{m_1} \mid S)) + \cdots + \log(P(X_{m_L} \mid S))$

- Smoothing
  - what if a word does not appear in any message in the training set?
  - assume we also saw *k* Negative messages and *k* Non-negative messages containing the word, in addition to the training set
  - $P(X_i \mid S) = (k$ + *#-negatives-with-*w_i *)* / $(2k$ + *#-negatives)*

Overall Structure

- Trainer
  - read the given data files
  - generate `model.csv` which is a Naive Bayes model for the sentiment classification learnt from the given dataset
    - each line is a triple of w_i , $P(X_i \mid S)$ and $P(X_i \mid \neg S)$

- Predictor
  - read `model.csv`
  - receive a sentence (one line of text) from the standard input
  - print out the prediction result as either 'Negative' or 'Non-negative'

Trainer Implementation

- Tokenization
- Normalization
- Probability estimation

Tokenization

- Convert a line of text to a set of words
  - convert each message to lowercase
  - exclude numbers, apostrpohes, etc.

- Reduce dictionary
  - remove useless words for classification (i.e., stopword)
    - E.g., “is”, “a”, “the”, “she”, “he”, “it”
  - remove redundant words
  - remove infrequent words

Normalization

- Stemming
  - reduce a word to its root form by cutting off prefix or postfix
    e.g., cheap, cheapest, cheaper → cheap
  - use the Snowball stemmer

Original description	`crashes when I Manage Bookmarks with a Personal Toolbar Folder link`
After stop-words removal	`crashes manage bookmarks personal toolbar folder link`
After stemming	`crash manag bookmark person toolbar folder link`

Counting

- Count the dictionary words in labeled training messages
  - both for Negative messages and for Non-Negative messages

- Turn the counts into estimated probabilities
  - logarithmic rescaling
  - smoothing

- Store the estimated probabilities of the dictionary to `model.csv`

Predictor

- Convert the given sentence to a set of words in the same way at the training
- Use the generated model to compute the estimatied probability of the sentence for being Negative
- Determine the sentence as Negative if the computed probability for being Negative/Non-negative exeeds certain thresholds

Programming Requirement

- The model file `model.csv` must be generated by a triainer program and you cannot directly update this
- Use the GTK/GNOME collection libary
  - useful data structure
    - Hash Table: https://developer.gnome.org/glib/stable/glib-Hash-Tables.html#g-hash-table-foreach
    - Balanced Tree (Map): https://developer.gnome.org/glib/stable/glib-Balanced-Binary-Trees.html
  - use GLib-2.0
    - already installed in Peace
    - e.g., `sudo apt-get install libglib2.0-dev libgtk2.0-dev`
- Use the Snowball stemmer
  - given with the starter code

Write-up Requirements

- Describe how your trainer module constructs a model
  - Explain and suppot your decisions at implementing tokenization, normalization and probability estimation
- Describe how your predictor module determines the sentiment of a given sentence
  - Explain and support your decisions
- Evaluate your program with the given test data with 200 messages
  - Count false positives, false negatives
- Discuss the limitation of your program and the ideas of improving this sentiment analyis

Evalution Criteria

- Write up (70 points)
  - Description on your solution design (45 points)
    - check whether your design is reasonable and sound
    - check whether your description is clear, consistent and informative
    - check whether your presentation is systematic and well organized
  - Discussion (25 points)
    - detailed analysis of results, interesting observations, lessons learned, suggestions, new ideas, etc.

- Tests (30 points)
  - Run each program with several inputs to see whether it works correctly