Sentiment Classification using Naive Bayes Classifier

Koç University COMP541 Self-Assessment Programming Exercise*

October 7, 2022

1 Introduction

In this project you will implement a sentiment analysis system based on the "Large Movie Review Dataset". The input to your system is a piece of text and the output is a binary classification of the text into "positive" or "negative". Here are some examples:

> **Input:** *But perhaps you have to have grown up in the 80's to truly appreciate this movie. If you love the early 80's this is definitely a must see. Also, one of the best soundtracks ever!*
>
> **Output:** *Positive*
>
> **Input:** *Widow hires a psychopath as a handyman. Sloppy film noir thriller which doesn't make much of its tension promising set-up.*
>
> **Output:** *Negative*

The core dataset contains 50,000 reviews split evenly into 25k train and 25k test sets. The overall distribution of labels is balanced (25k pos and 25k neg). Feel free to use any preprocessing and prediction method you like. Train on the training data and report your accuracy on the test data.

Please start by downloading and reviewing the data from

> `https://ai.stanford.edu/~amaas/data/sentiment`

2 Preprocessing

You can explore the following ideas for preprocessing, they are all optional and not all may be helpful:

- Tokenize the text into individual words and punctuation.
- Lowercase the words.
- Map each unique token to an integer to ease further processing.
- Use an "unknown" symbol for tokens below a frequency threshold.
- Use only a subset of the words you deem informative as features.
- Crop longer texts and/or pad shorter texts to a fixed length.

3 Model

For prediction, you will implement a Naive Bayes Classifier. This classifier assumes the following generative process: To generate each (x, y) pair where $y \in \mathcal{Y}$ is a class and $x = [x_1, x_2, \ldots, x_n]$ is a document with features (e.g. words) $x_i \in \mathcal{X}$:

- First pick $y \in \mathcal{Y}$ with probability q_y (categorical distribution).
- Then pick each word $x_i \in$ with probability $q_{x_i|y}$ (more categorical distributions).

*Initially prepared by Deniz Yuret.

Given this generative process, we can write the probability of generating a particular (x, y) pair as:

$$P(x, y) = q_y \, q_{x_1|y} \, q_{x_2|y} \ldots = q_y \prod q_{x_i|y}$$

The maximum likelihood estimates for the Naive Bayes parameters would be:

$$\hat{q}_y = \frac{\text{number of documents with class } y}{\text{number of documents}}$$

$$\hat{q}_{x_i|y} = \frac{\text{number of words} = x_i \text{ in class } y}{\text{number of words in class } y}$$

Here are some potential issues with the Naive Bayes classifier and possible ways to deal with them:

- Maximum likelihood overfits: some words will have zero probability if never before observed with a class.
  Possible solution: Use add-one (or other) smoothing (i.e. add a fixed amount to each count).
- Independence assumptions (bag-of-words view) too strong.
  Possible solution: Use ngram models for each class (i.e. take n word sequences as features).
- If a word appears once it is more likely to appear again in a document.
  Possible solution: Dirichlet compound multinomial (DCM) model (Murphy 2012, Sec. 3.5.5: `http://www.cs.ubc.ca/~murphyk/MLbook`; Madsen et al. 2005: `http://eprints.pascal-network.org/archive/00001454/01/dcmnbV10.1.pdf`) takes into account the burstiness of word usage.
- Joint distribution $P(x, y)$ may be difficult to learn and unnecessary if all we need is $P(y|x)$.
  Possible solution: Conditional model, learn $P(y|x)$ directly.

These suggestions are all optional. However, in order to complete this assignment, you need to achieve minimum 80% accuracy on the test set.

4 Requirements

In order to complete this assignment, you MUST fulfill all the following requirements,

- You MUST implement Naive Bayes classifier: you cannot select another model.
- You MUST implement this assignment using Python.
- To complete this assignment, you CANNOT use the 3rd party packages except NumPy.
- You CANNOT use the preprocessed data (imdb.vocab, labeledBow.feat, unsupBow.feat etc.).
- Your implementation MUST run less than 1 minute on regular workstations.
- You MUST achieve minimum 80% accuracy on the test set.
- You MUST NOT use hard-coded paths for the data directory. You can achieve this in two different ways: (i) define a DATA_DIR variable at the beginning of your code or (ii) you can have the data at the same directory with your code.
- Your code MUST run without any errors.
- You MUST upload a single file (either .py or .ipynb).

Warning

If you think this assignment is too difficult for you, or you spend more than 10 hours completing it, please consider taking this course the next year.