Natural Language Processing and Prediction of Sentiment Labelled Sentences

Assignment Submission for Course CP8305 Instructed by Dr. Cherie Ding

Richard Wen

Ryerson University

April 12, 2017

Contents

1 Objectives

This assignment was provided by Dr. Cherie Ding for the CP8305 Knowledge Discovery course at Ryerson University. The purposes of the assignment was to:

1. Identify a practical dataset from the University of California, Irvine (UCI) Machine Learning Repository (Lichman 2013) or Knowledge Discovery in Datases (KDD) Cup Archives (SIGKDD 2013)

2. Identify a machine learning problem with the chosen dataset in (1)

3. Apply various machine learning algorithms to the problem in (2) to find algorithms that can solve (2) well and provide insight into the data in (1)

2 Data

The data chosen for this assignment was the Sentiment Labelled Sentences (SLS) Dataset donated on May 30, 2015 and downloaded from the UCI Machine Learning Repository (Kotzias et al. 2015). There are 3 text files (amazon_cells_labelled.txt, imdb_labelled.txt, yelp_labelled.txt) with a combined total of 3000 instances, absent of missing values. Each file consists of 2 attributes with the first attribute being sentences from (string type) and the second being a binary class of either 0 for negative sentiment or 1 for positive sentiment (numeric type). The data in each file had attributes separated by a mixture of inconsistent spaces and tabs, and instances separated by rows. An example of the first 5 rows are shown in Table 1[1]. Sentences were extracted by Kotzias et al. (2015) from imdb.com, amazon.com, and yelp.com. These websites represent a movie database, online retailer, and a online business directory with crowd-sourced reviews, respectively. The SLS files are summarized in Table 2[2].

[1]In the table, (tab) indicates that a tab character was present in the data sample.

[2]For clarification, a double indicates type numeric and a character indicates type string.

Table 1: SLS Dataset Example for amazon Data

Line	Sample
1	So there is no way for me to plug it in here in the US unless I go by a converter. (tab) 0
2	Good case, Excellent value. (tab) 1
3	Great for the jawbone. (tab) 1
4	Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!! (tab) 0
5	The mic is great. (tab) 1

Table 2: SLS Dataset Summary

File	Instances	Attributes
amazon_cells_labelled.txt	1000	2
imdb_labelled.txt	1000	2
yelp_labelled.txt	1000	2

3 Problem

The goal of the SLS dataset was to predict the values in attribute 2 (sentiment) which contain 0 or 1 for negative and positive sentiment respectively. The binary values of attribute 2 defined the SLS dataset as a classification problem. In order to predict sentiment, sentences from three different websites (imdb.com, amazon.com, and yelp.com) were given as the explanatory attribute 1. The text values of attribute 1 further defined the problem as a Natural Language Processing (NLP)[3] problem where the attribute values were unstructured, and required pre-processing before the machine could read and learn to model the SLS data. For simplicity, attribute 1 was referred to as the sentence attribute and attribute 2 was referred to as the sentiment attribute. The problem was then known as a classification problem for the sentiment attribute that required NLP of the sentence attribute. The following sections more formally define the problem framework.

[3]NLP seeks to extract meaning from textual data involving language communication with low level tasks such as identification of individual words and high level tasks such as spelling correction (Nadkarni, Ohno-Machado, and Chapman 2011)

3.1 Sentiment Attribute

The sentiment attribute was the second attribute in the SLS dataset. It is the target vector $y^{(n)}$ containing binary values of either 0 or 1 given n instances as seen in Equation (1):

$$y^{(n)} \in \{0, 1\} \tag{1}$$

3.2 Sentence Attribute

The sentence attribute was the first attribute in the SLS dataset. It is the raw text data $x^{(n)}$ given n instances such that it contains k number of words $w^{(k,n)}$ (separated by spaces[4]), where the word lengths[5] l_w of $w^{(k,n)}$ and k are less than the length l_x of the raw texts $x^{(n)}$ as seen in Equation (2):

$$w^{(k,n)} \in x^{(n)} \mid \;\; 0 < l_w^{(k)} \le l_x^{(n)} \;\; and \;\; 0 < k \le l_x^{(n)} \tag{2}$$

3.3 Measures of Prediction Quality

The classification problem given $x^{(n)}$ as the explanatory data and $y^{(n)}$ as the target classes was to obtain measurement values that define an algorithm to predict $y^{(n)}$ well. Measurement values were based on whether predicted classes $y_{pred}^{(n)}$ obtained using $x^{(n)}$ seen in Equation (3):

$$\lim y_{pred}^{(n)} = f(x^{(n)}) \tag{3}$$

The classification prediction quality used measurements that were based on the f_{eq} counts of $y_{pred}^{(n)}$ equal

[4]For example, the text instance “hello goodbye now”, contain 3 words “hello”, “goodbye”, and “now” that are separated by spaces

[5]Word lengths and text lengths are measured in the number of characters, excluding spaces and symbols that define punctuation, which are more specifically the number of alphanumeric characters in this case (e.g. the word “apple” has a length of 5 alphanumeric characters and the text “!*–” has a length of 0 characters containing non-alphanumeric characters)

to $y^{(n)}$ given the total number of instances N expressed in Equation (4):

$$f_{eq} = \sum_{n=1}^{N} y_{eq} \mid y_{eq} = \begin{cases} 1: & \text{if } y_{pred}^{(n)} = y^{(n)} \\ 0: & \text{otherwise} \end{cases} \tag{4}$$

Accuracy measurements were defined as a maximization problem, where higher values are better and lower values are worse. Error measurements were defined as a minimization problem, where lower values are better and higher values are worse. An accuracy measurement f_{acc} increases the more times $y_{pred}^{(n)}$ is equal to $y^{(n)}$ given f_{eq} as seen in Equation (5):

$$\lim_{f_{acc} \to +\infty} f_{acc} \;\; as \;\; f_{eq} \to +\infty \quad | \; f_{acc} = f(y_{pred}^{(n)}, y^{(n)}) \tag{5}$$

An error measurement f_{err} decreases the more times $y_{pred}^{(n)}$ is equal to $y^{(n)}$ given f_{eq} as seen in (6):

$$\lim_{f_{err} \to -\infty} f_{acc} \;\; as \;\; f_{eq} \to +\infty \quad | \; f_{err} = f(y_{pred}^{(n)}, y^{(n)}) \tag{6}$$

4 Methods

The methods described in this section attempt to produce a solution (known as the Best Attempt (BA) solution) to the classification problem defined in Section 3. The unstructured nature of the sentence attribute presented required preprocessing to create features which are then further preprocessed to search for adequately useful features (selection). These features were then randomly split into approximately equal number of instances for cross validation training sets. These training sets were then used as input for a selected number of algorithms, which were optimized for algorithm specific parameters and evaluated for prediction quality. A summary of the methods is shown in Figure 1. See Appendix A for the R code.

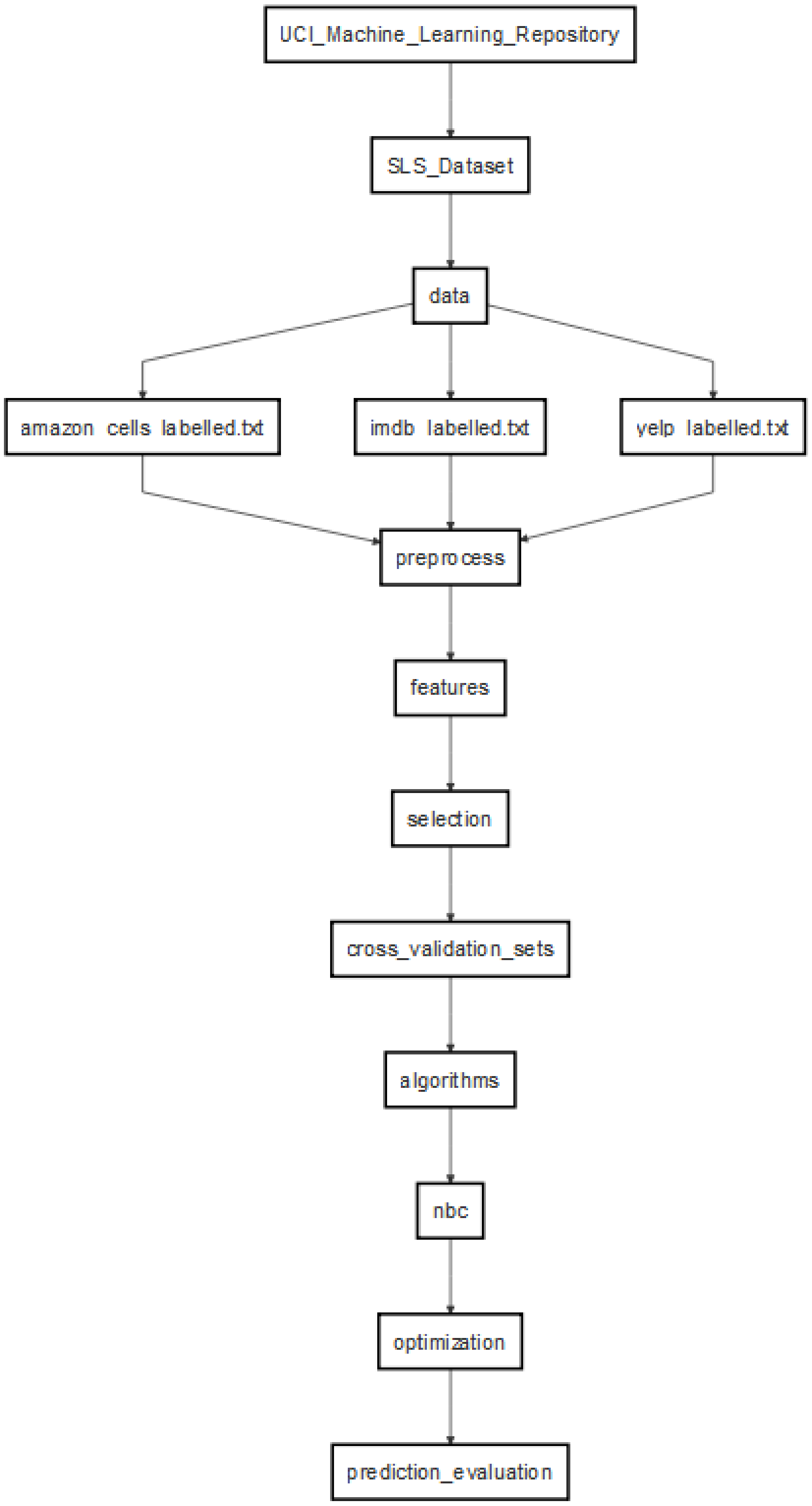

Figure 1: Flowchart of Methods

4.1 Preprocessing

The preprocessing steps involved getting the data ready for the machine learning algorithms to train on. This involved creating features from the instances in the sentence attribute and selecting only the features that are measured to be useful in terms of predicting the target sentiment attribute. Furthermore, the data was also split into cross validation training sets and used as the resulting training data for the machine learning algorithms.

4.1.1 Parsing

The data was defined as a tab delimited text file that held string based text as the first sentence attribute and numeric binary numbers as the second sentiment attribute. The unstructured nature of the sentence attribute made parsing the file less straight-forward as instances in the sentence attribute were not quoted and could contain any number of tab characters. The data was: (i) parsed line by line (ii) cleaned by removing the last occurence of tab characters, and (iii) extracted for sentiment and sentence instances where the last character was defined as the sentiment, and the rest of the text was defined as the sentence.

```r
# 1. Read the data on a line by line basis
ds <- sapply(dsFiles, readLines)

# 2. Extract sentence and sentiment attributes
x <- c() # sentence attribute
y <- c() # sentiment attribute
for (n in ds) { # each instance n
  lx <- nchar(n) # sentence length
  xn <- substr(n, 1, lx - 2) # sentence instance
  yn <- as.numeric(substr(n, lx - 1, lx)) # sentiment instance
  x <- c(x, xn)
  y <- c(y, yn)
```

Table 3: Example of Word Features

Word-1	Word-2	Word-k
2	5	10
5	2	5
10	10	2

```r
}
dsParsed <- data.frame(sentence=x, sentiment=y, stringsAsFactors=FALSE)
```

4.1.2 Features

The features[6] were created using a simple bag of words model that was not order-dependent. The bag of words model constructs a feature for each unique word in the sentence attribute and counts the occurence per sentence instance in the SLS dataset (Nadkarni, Ohno-Machado, and Chapman 2011). Each k word feature $w^{(n,k)}$ is a count of the occurence of k unique words in the sentence attribute $x^{(n)}$ given n instances. For clarification, all words were considered, including stop words[7], as these would be removed in the feature selection process if they were measured as being not very useful - thus, the possibility of them being useful was considered. An example of the word features is given in Table 3.

```r
library(text2vec)
xn <- dsParsed$sentence

# 1. Obtain words wn for each xn using a tokenizer
wn <- itoken(xn, tolower, word_tokenizer)

# 2. Vocabulary of wk words for wn
wk <- vocab_vectorizer(create_vocabulary(wn))
```

[6]Features in this report are similar to attributes, except that features refer to the machine-constructed columns to differ from the original sentence and sentiment attributes

[7]Stop words are commonly used words in the language that are often, but not always, removed as they do may not hold useful information

```r
# 3. Obtain word features matrix wnk given n instances and k words
wnk <- as.matrix(get_dtm(create_corpus(wn, wk)))
```

4.1.3 Feature Selection

4.1.4 Cross Validation

The data from the feature selection was split into a standard 10-fold cross validation scenario. In the 10-fold cross validation scenario, the data was randomly split into 10 equal parts, trained on a 9 parts, and tested on the 1 part not in the 9 training parts until all testing parts have been tested for (Borra and Di Ciaccio 2010).

4.2 Algorithms

4.2.1 Naive Bayes Classifier

5 Appendix A: R Code

```r
# Dependencies ----

# (package_install) Required packages and loading
packages <- c("text2vec")
for (pkg in packages) {
  if (!requireNamespace(pkg, quietly=TRUE)) {
    install.packages(pkg,
                     dependencies=TRUE,
                     repos="http://cran.rstudio.com/")
  }
  library(pkg, character.only=TRUE)
}

# Data (See Section 2) ----

# 1. Download the zipped SLS data from UCI into a temp dir
temp <- tempdir()
src <- paste0("http://archive.ics.uci.edu/ml/machine-learning-databases/",
              "00331/sentiment%20labelled%20sentences.zip")
zipped <- file.path(temp, basename(src))
download.file(src, zipped)
unzip(zipped, exdir=temp)

# 2. Obtain SLS data paths from unzipped folder
dsFolder <- file.path(temp, "sentiment labelled sentences")
```

```r
dsIgnore <- file.path(dsFolder, c(".DS_Store", "readme.txt"))
dsFiles <- list.files(dsFolder, full.names=TRUE)
dsFiles <- dsFiles[!dsFiles %in% dsIgnore]

# Parsing (See Section 4.1.1) ----

# 1. Read the data on a line by line basis
ds <- sapply(dsFiles, readLines)

# 2. Extract sentence and sentiment attributes
x <- c() # sentence attribute
y <- c() # sentiment attribute
for (n in ds) { # each instance n
  lx <- nchar(n) # sentence length
  xn <- substr(n, 1, lx - 2) # sentence instance
  yn <- as.numeric(substr(n, lx - 1, lx)) # sentiment instance
  x <- c(x, xn)
  y <- c(y, yn)
}
dsParsed <- data.frame(sentence=x, sentiment=y, stringsAsFactors=FALSE)

# Features (See Section 4.1.2) ----

# 1. Obtain words wn for each xn using a tokenizer
xn <- dsParsed$sentence
wn <- itoken(xn, tolower, word_tokenizer)
```

```r
# 2. Vocabulary of wk words for wn

wk <- vocab_vectorizer(create_vocabulary(wn))


# 3. Obtain word features matrix wnk given n instances and k words

wnk <- as.matrix(get_dtm(create_corpus(wn, wk)))
```

References

Borra, Simone, and Agostino Di Ciaccio. 2010. "Measuring the Prediction Error. A Comparison of Cross-Validation, Bootstrap and Covariance Penalty Methods." *Computational Statistics and Data Analysis* 54 (12). Elsevier: 2876–2989. doi:10.1016/j.csda.2010.03.004.

Kotzias, Dimitrios, Misha Denil, De Freitas Nando, and Padhraic Smyth. 2015. "From Group to Individual Labels Using Deep Features." *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 597–606. doi:10.1145/2783258.2783380.

Lichman, Moshe. 2013. "(UCI) Machine Learning Repository." http://archive.ics.uci.edu/ml.

Nadkarni, Prakash M, Lucila Ohno-Machado, and Wendy W Chapman. 2011. "Natural Language Processing: An Introduction." *Journal of the American Medical Informatics Association* 18 (5). PubMed Central: 544–51. doi:10.1136/amiajnl-2011-000464.

SIGKDD. 2013. "KDD Cup Archives." www.kdd.org/kdd-cup.