Concept Drift

28.10.2018

—Indian Institute of Information Technology Allahabad

IIT2016125 - Neil

BIM2016006 - Ritik

IIT2016122 - Shivam

IIT2016127 - Rohit

IIM2016501 - Samanwaaya

Table of Contents

Introduction

Overview

We integrated Deep Learning and Natural Language Processing to extract the Labels and eventually the difference between given two documents. After training and classification we ask our model to predict “How likely is it that the text belongs to a certain Label” which gives a probability and then we calculate “How different are the two documents” which gives a percentage of difference.

Goals

1. Classify the the documents into labels
2. Calculate likelihood of document to belong to certain label
3. Predict the difference between two documents

Specifications

- **Datasets**: BBC Newsgroups
- **Libraries**: Scikit-learn, Numpy, Pandas, Matplotlib
- **Environment**: Jupyter Notebook
- **Package Management**: Pip

Coding Standards

- **Language:** Python 3.7
- **Paradigm:** Object Oriented
- **Version Control**: Git

Dataset Description

The **BBC Newsgroups** data set is a collection of approximately 2000 newsgroup documents, partitioned (nearly) evenly across 5 different newsgroups. To the best of our knowledge, it was originally collected by Ken Lang, probably for his Newsweeder: Learning to filter netnews paper, though he does not explicitly mention this collection. The 5 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering.

Organisation

The data is organized into 5 different newsgroups, each corresponding to a different topic. Some of the newsgroups are very closely related to each other (e.g. comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), while others are highly unrelated (e.g misc.forsale / soc.religion.christian).

Data

The data available here are in **.tar.gz** bundles. You will need **tar** and **gunzip** to open them. Each subdirectory in the bundle represents a newsgroup; each file in a subdirectory is the text of some newsgroup document that was posted to that newsgroup.

Libraries Description

Scikit-Learn

It is an efficient tool for data mining and data analytics.It is built on NumPy, SciPy, and matplotlib.Some of the typical usage include: Classification, Regression, Clustering, Dimensionality Reduction and Preprocessing.

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things like a powerful N-dimensional array object, sophisticated (broadcasting) functions, tools for integrating C/C++ and Fortran code, useful linear algebra, Fourier transform, and random number capabilities

Pandas

Pandas is an open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Software Design

1. We used load_files function which loads text files with categories as subfolder names. Our dataset already has articles organized into different folders. After loading the data, we'll also check how many articles are there per category.
2. **Data preparation:** Now we'll split the data into training and testing set and then print out first 80 chars of some samples.The common NLP pipelines are:
    a. Tokenize i.e. split the text into words
    b. Convert the case of letters to either upper or lower
    c. Remove stopwords. For e.g. "the", "an", "with"
    d. Perform stemming or lemmatization to reduce inflected words to its stem. For e.g. transportation -> transport, transported -> transport
    e. Vectorization (Count, Binary, TF-IDF)
3. We need to convert it to a numerical format. A very common method, among others, is to calculate TF-IDF matrix. TF stands for term frequency in which we calculate how many times a term/word appears in a document. IDF stands for inverse document frequency which measures how important a word is. In simple terms it gives more weight to rare words than common ones. Once we calculate both TF and IDF, we can simply multiply them together to obtain TF-IDF value.
4. We used TfidfVectorizer to calculate TF-IDF. When initializing the vectorizer, we passed stop_words as "english" which tells sklearn to discard commonly occurring words in English. Then we also specified max_features to 1000. The vectorizer will build a vocabulary of top 1000 words (by frequency). This means that each text in our dataset will be converted to a vector of size 1000. Next, we call fit function to "train" the vectorizer and also convert the list of texts into TF-IDF matrix
5. **Build Mode**l: We used Naive Bayes Model. To optimise the accuracy we considered different scikit-learn pipelines and concluded that Support Vector Machine with tf-idf features scored the highest with accuracy of 97%.
6. **Training**: We split the data set to 8:2 and train the aforementioned model
7. **Testing**: Our model performed well with an accuracy of 98% on test data.

Naive Bayes Classifier

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

- Feature matrix contains all the vectors (rows) of dataset in which each vector consists of the value of dependent features.
- Response vector contains the value of class variable (prediction or output) for each row of feature matrix.

Naive Bayes can be modeled in several different ways including normal, lognormal, gamma and Poisson density functions:

$$p(x_k \mid C_j) = \begin{cases} \dfrac{1}{\sigma_{kj}\sqrt{2\pi}} \exp\left(\dfrac{-(x-\mu_{kj})^2}{2\sigma_{kj}}\right), \quad -\infty < x < \infty, -\infty < \mu_{kj} <, \sigma_{kj} > 0 & \text{Normal} \\ \mu_{kj} : \text{mean}, \ \sigma_{kj} : \text{standard deviation} & \\ \dfrac{1}{x\sigma_{kj}(2\pi)^{1/2}} \exp\left\{\dfrac{-[\log(x/m_{kj})]^2}{2\sigma_{kj}^{\,2}}\right\}, \quad 0 < x < \infty, m_{kj} > 0, \sigma_{kj} > 0 & \text{Lognormal} \\ m_{kj} : \text{scale parameter}, \ \sigma_{kj} : \text{shape parameter} & \\ \dfrac{\left(\dfrac{x}{b_{kj}}\right)^{c_{kj}-1}}{b_{kj}\Gamma(c_{kj})} \exp\left(\dfrac{-x}{b_{kj}}\right), \quad 0 \le x < \infty, b_{kj} > 0, c_{kj} > 0 & \text{Gamma} \\ b_{kj} : \text{scale parameter}, \ c_{kj} : \text{shape parameter} & \\ \dfrac{\lambda_{kj}\exp(-\lambda_{kj})}{x!}, \quad 0 \le x < \infty, \lambda_{kj} > 0, x = 0,1,2,\ldots & \text{Poisson} \\ \lambda_{kj} : \text{mean} & \end{cases}$$

Note. Poisson variables are regarded here as continuous since they are ordinal rather than truly categorical. For categorical variables, a discrete probability is used with values of the categorical level being proportional to their conditional frequency in the training data.

Vectoriser

TF (Term Frequency)

- **tf(T,D)**: Frequency of term for N-gram 'T' in a document 'D'.
- The weight of a term that occurs in a document is simply proportional to the term frequency.

Variants of term frequency (tf) weight

weighting scheme	tf weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} \Big/ \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

IDF (Inverse Document Frequency)

- N = |D| = total number of docs in our corpus
- |\{d \in D: t \in d\}| : number of documents where the term {\displaystyle t} t appears (i.e., {\displaystyle \mathrm {tf} (t,d)\neq 0} \mathrm{tf}(t,d) \neq 0). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to {\displaystyle 1+|\{d\in D:t\in d\}|} 1 + |\{d \in D: t \in d\}|.

$$\operatorname{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

-
- The inverse document frequency is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient)

Variants of inverse document frequency (idf) weight

weighting scheme	idf weight ($n_t = \|\{d \in D : t \in d\}\|$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log\left(1 + \frac{N}{n_t}\right)$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1 + n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

-

TF-IDF

- tf–idf is calculated as:

$$\mathbf{tfidf}(t, d, D) = \mathbf{tf}(t, d) \cdot \mathbf{idf}(t, D)$$

- A high weight in tf–idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights.hence tend to filter out common terms. Since the ratio inside the idf's log function is always greater than or equal to 1, the value of idf (and tf–idf) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and tf–idf closer to 0.

Recommended tf–idf weighting schemes

weighting scheme	document term weight	query term weight
1	$f_{t,d} \cdot \log \frac{N}{n_t}$	$\left(0.5 + 0.5 \frac{f_{t,q}}{\max_t f_{t,q}}\right) \cdot \log \frac{N}{n_t}$
2	$1 + \log f_{t,d}$	$\log\left(1 + \frac{N}{n_t}\right)$
3	$(1 + \log f_{t,d}) \cdot \log \frac{N}{n_t}$	$(1 + \log f_{t,q}) \cdot \log \frac{N}{n_t}$

-

Equations To Calculate Percentage Difference:-

First we find the absolute difference of probabilities for both text with respect to every tag.

We take the maximum of these absolute differences and multiply it by 100 to convert it into percent.

Here the probability represent how much the text belongs to a particular topic.

Higher the probability of a topic ,greater part of the text belongs to the topic.

So, maximum of these absolute differences will give us the difference in concept of two texts.

Result:-

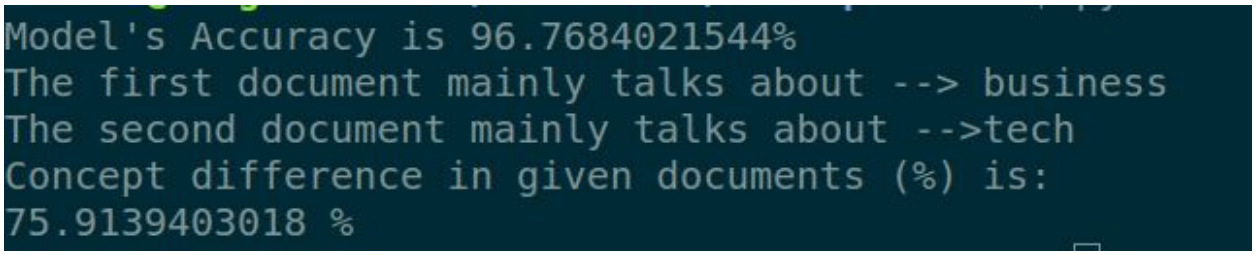
Input1-

```
150,000 cars a year. This will boost the annual production capacity of the
company - India's second-largest car manufacturer - to 400,000 units.
Hyundai expects its sales in India to grow 16% to 250,000 in 2005.
By 2010, it expects to nearly double sales to 400,000 cars.
The new plant will be built close to the existing one in Chennai, in the southern province of Tamil Nadu.
South Korea's top car maker estimates that the Indian market will grow 15% this year, to 920,000 vehicles, reaching 1.
```

Input2-

```
Lasers help bridge network gaps
An Indian telecommunications firm has turned to lasers to help it overcome the problems of setting
up voice and data networks in the country.
Tata Teleservices is using the lasers to make the link between customers' offices and its own core network.
The laser bridges work across distances up to 4km and can be set up much faster than cable connections. In 12 months t
"In this particular geography getting permission to dig the ground and lay the pipes is a bit of a task,
" said Mr R. Sridharan, vice president of networks at Tata. "Heavy traffic and the layout under the ground
mean that digging is uniquely difficult," he said. In some locations, he said, permission to dig up roads
and lay cables was impossible to get. He said it was far easier to secure permission for putting networking
hardware on roofs. This has led Chennai-based Tata to turn to equipment that uses lasers to make the final mile
leap between Tata's core network and the premises of customers. The Lightpointe laser bridges work over distances
of up to 4km and are being used to route both voice and data from businesses on to the backbone of the network.
The hardware works in pairs and beam data through the air in the form of laser pulses.
The laser bridges can route data at speeds up to 1.25gbps (2,000 times faster than a 512kbps broadband connection) but
```

Output-



Discussion

Here we observe that using simple Naive Bayes did not attain the expected accuracy and it was limited to 75% which is impressive but not worthy of practical usage, Thus to optimise the model we decided to add feature of TF-IDF which on training dataset obtained an accuracy of 97%, there are other models which can be tested for the same task like SVM and Deep Neural Nets but considering the constraint on size of the dataset and a cap on number of topics that can be classified, our Naive Bayes model fits the best.

The 20 Newsgroup dataset has 46 topics classified so our model tries to guess the best fit within these topics only, A larger and more precise training data could help us enhance the accuracy even more and Deep Neural Nets as it is shown in results of other experiments has the highest accuracy given ample data and computing power

References:-

1. https://hackernoon.com/how-to-build-a-simple-spam-detecting-machine-learning-classifier-4471fe6b816e?fbclid=IwAR0anmG_AV2_7Merk7PUcqk9wKPyvy8urj2rNVJY7VdF-Bseg1CjTp6Wz4A
2. https://machinelearningmastery.com/crash-course-deep-learning-natural-language-processing/?fbclid=IwAR2x7V-O4SFolkNQuVWAibqTBjf7OSW_gk9sSvJ8AltXjzG0TVFZ1vW7YZQ
3. https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/?fbclid=IwAR1qktvE3F6aKuLdMUoC-P_tgQAFNDnZOIBgWAOkDy2FCD-cOKujZc8HoGA
4. http://scikit-learn.org/stable/documentation.html
5. https://en.wikipedia.org/wiki/Tf%E2%80%93idf
6. http://mlg.ucd.ie/files/datasets/bbc-fulltext.zip