COMP30027 Machine Learning

Project 2 Report

Abstract

This report outlines the methods we used for building blog-authorship classifiers for age prediction by applying a variety of machine learning algorithms. The approaches used for data preprocessing and experimental design are explained in detail, and results are discussed with idea of future improvements.

Introduction

Short text classification is a well-studied topic in the area of machine learning and natural language processing. By researching through different studies and papers, we concluded that we should start the study by analysing training data thoroughly using vectorisation method to extract as much features as possible from the short corpus provided. We also deduced that the frequency of tokens is not necessarily closely related to the age of author; on the other hand, tokens that are age-representative are more related, for example, younger generations tend to use more terms related to school. As a result, we decided to use count vectorisation method on the raw dataset. (Cavnar et al, n.d.)(Nigam et al, n.d.)(Schler et al, 2006)(Sun, 2012)

Preprocessing

- Data Cleaning

Initially, two datasets are provided for training purposes: “train_raw.csv” and “train_top10.csv”. We focused on data cleaning of raw dataset, as our training data is filtered from raw dataset.

Technique	Reason	Before	After
Remove redundant spaces	Improve n-gram representation for vectorisation	“I am”	“I am”
Remove punctuations	Most punctuations are not helpful for text classification purposes in this project; however, question marks and exclamation marks are not removed since initial observation suggests that younger	“yes,”	“yes”

	age groups tend to use them quite frequently in blog posts		
Remove url links	Url links and tags are not related to the identification of corpus and age	"www.google.com yes"	" yes"
Convert all letters to lowercase	Accurate n-gram representation, avoid duplicate recognition of the same word	"YES"	"yes"

Table 1

- Data Filtering/Augmentation

User_id	New_Text	Age

Table 2 - Our training set

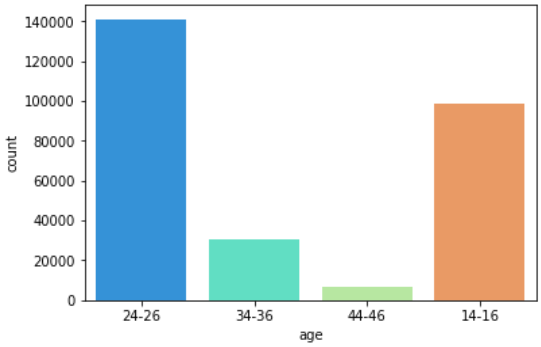
For each unique user id, we concatenated all the blog posts of that particular user into a single String("New_Text") in order to minimise the short length nature of each corpus in the original dataset. By doing this, one user will have only one particular age; although the number of instances used to build the model is significantly decreased and might result in unbalanced or biased models, we think this is the better way to process data. If the instances of a single user are not aggregated together, it is very likely that the same user will have different ages, which is obviously not reasonable and should be avoided.

In addition, we choose to use the precise age of each user as the class instead of using age range as the class, and group the predicted ages into ranges. By predicting the precise age first, we are allowing an error range of the prediction by grouping one user's predicted ages into the corresponding range.

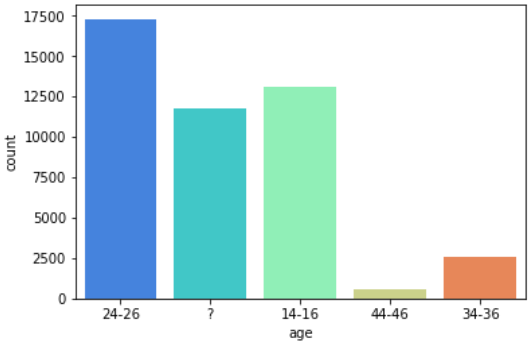
- Data visualisation

Upon visualising the distribution of age groups in both training and development dataset using countplot, we quickly realised that there is no unknown label in development data, but the proportion of unknown class is approximately 25% in development data. In addition, 14-16 and

24-26 groups together constitute more than half of both datasets. On the other hand, the data size of 34-36 and 44-46 groups are significantly small, indicating the dataset is imbalanced. As a result, it is vital to make accurate predictions of 24-26, 14-16, and unknown classes in order to achieve a higher overall accuracy of out prediction.



Graph 1



Graph 2

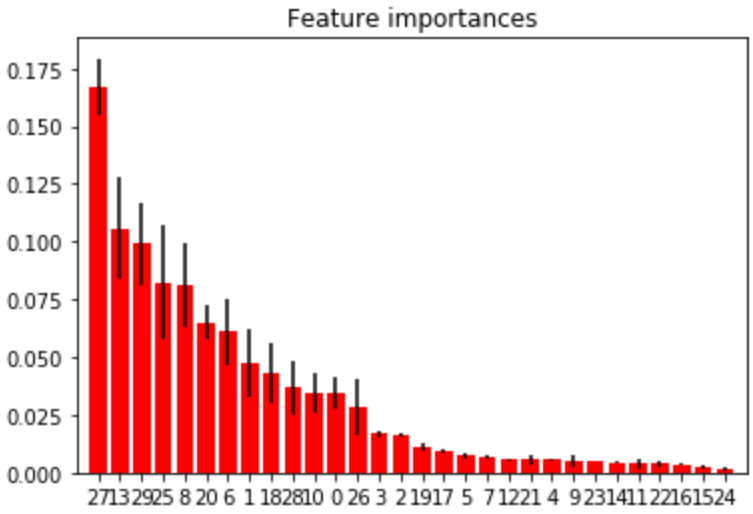
- Text Preprocessing/Data Representation

Our major issue with the dataset is that it is all in text(String) format. As a result, it is necessary to represent our text data in ways that are suitable for text classification. By researching a number of text identification algorithms, we found it necessary to have numerical feature vectors to perform text classification (Sun, 2012)(Schler et al, 2006); that is, to convert each corpus into a vector format. While vectorisation of the dataset outputs a sparse matrix, which leads to large data volume and may affect grid search and ensemble system in later stage, we think vectoristion is significant, so the pros outweigh the cons.

Literature points out there are two commonly used approaches that stand out in natural language processing: “bag of words” and “n-grams”. The “bag of words” approach is the simplest; this technique breaks up the text into words, and each unique word is represented by its frequency. The “n-grams” approach is more complex, subsequence of the text and its frequency and other features are considered”.

As an initial experiment, we took advantage of the “top10_train.csv” dataset, as top tokens are already extracted with their frequencies calculated, which is very similar to the “bag of words” technique. By aggregating the total frequency of each token of each user, the number of instances changed from 65,535 to 8950; we selected features with feature importances higher than 0.05. As the graph(Graph 3) shows, the 30 tokens are label encoded, with feature importance represented as bar charts.However, the accuracy produced was not ideal (around 40 percent), which is not ideal. In addition, with comparison to the “bag of words” representation,

we prefered the “n-gram” approach, as it is more likely to generate more features from each corpus than simply splitting the text into tokens.



Graph 3

In order to select the best length of n-gram, literature suggests that the best way to select the parameter is to use 10-fold cross validation on the training data for a range of value of n, then select the value of n that produces the minimum validation error rate (Cavnar et al, n.d.). Yet, given the limit of our machines, we were unable to complete n-gram representation with size greater than 3. In particular, trigram took over a night to complete, which clearly is not efficient under time-limited circumstances. As a result, in the later part of this project, we only focused on word based unigram representation of the text data.

Subsequently, unigram of the dataset is generated and token counts are stored as a matrix by vectorisation. In addition to the counting, we also used term frequency-inverse document frequency (TF-IDF) for weighting and normalisation.

Experiment Design

With a cleaned training dataset processed as above, an experiment is designed to identify the performance of each classifier in this task. In overall, classifiers are tuned with grid search, and are ensemble into a stacking classifier. Moreover, approaches involving threshold tuning are taken to address the prediction of unknown classes.

- Base Classifiers

Base classifiers are trained with the consideration to produce only probabilistic outputs for each class in order to perform threshold tuning in the second stage. As illustrated below, we list all base classifiers that were used along with their hyperparameters that were tuned.

1. Logistic Regression

Logistic regression uses a linear method, classes are predicted by transforming the linear regression model using logistic function.

As for this project, the predicted class is likely to have a linear relationship with feature values; for example, some words could be used more and more frequently with the growing of age. As a result, by transferring the linear regression model fitted with class labels and feature values to a logistic regression model, we can predict the possibility of each age range easily.

Penalty: L1, L2 C: [1, 10] Fit intercept: True, False

2. Multinomial Naive Bayes

Multinomial naive bayes classifiers are often used as baseline classifier for text classification; it is able to "capture word frequency information in documents" according to literatures(Nigam et al, n.d.).

Alpha = [0,1.0], fit_prior = True, False

3. K-Nearest Neighbour

KNN makes predictions using the training dataset directly. A new instance is predicted by searching through the entire training set for the K most similar instances. We found that the performance of KNN is the lowest among all the classifiers, the reason may be data are of sparse matrix type, which is characterized by its high dimensionality.

K neighbour = [1,10] Weights: uniform, distance

4. Stochastic Gradient Descent Classifier

SGD classifier implements regularised linear models using SGD learning method. This method is particularly useful for training data involving sparse matrix.

Penalty: L1, L2 Loss: hinge, log, squared hinge, perceptron

5. Stacking

As we have vectorised the data and used tf-idf in data preprocessing phase, the processed data is of sparse matrix type, which is hard to manipulate; we need to transfer it to a data frame. However, since the amount of the processed data is too large, the attempt of converting the processed data to data frame failed as memory error is raised up. Such a large amount of data can be easily fit into the form of a sparse matrix; however, a data frame is denser, and takes up much more memory with the same amount of data. So, we have to select only part of the processed data to data frame to avoid memory error.

There are in total more than 390,000 unique words, and we chose 10,000 of them (top 25% ranking by frequency). Thus, memory error is avoided and noises are eliminated; dimension is reduced. The performance of the Stacking Ensemble model was the best among all the base classifiers, which is expected, since the predicted classes of base classifiers are appended into the training data, which is used to train the meta classifier, in this case, logistic regression model.

Since the 3 base classifiers we implemented are of totally different properties, each of them may have different specialties on predicting instances with some special features. For example, we found that the KNN is good at predicting the age group of 34-36, which is bad in the performance of other classifiers.

By training the meta-classifier with the predictions generated from the base classifiers, the meta-classifier can learn which classifier has the best performance for each class of instances, so it can combine all the advantages of these base classifiers, and choose the predictions generated from these base classifiers to get a better performance.

- Parameter Tuning

1. Threshold Tuning For “Unknown” Classes

We observed that there exists unknown class labels in the develop dataset and test dataset; however, these unknown class labels do not exist in the training dataset. The instances of the unknown class are those whose age does not fall into the 4 age ranges (14-16, 24-26, 34-36,44-46).

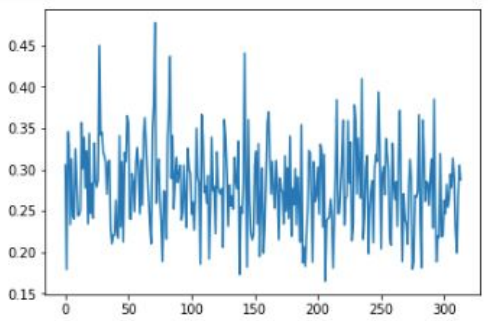
After doing some research on unknown label handling, we found that there are two approaches. First is to generate instances of the unknown class, label them as unknown, and append those artificially generated instances to the training dataset to fit the model. However, we found that this approach is not applicable, as it is too hard to artificially generate those instances. We could randomly adapt the development instance of unknown class to generate new instances of

unknown class; however, it is not allowed to use the development instances in the training phase. So, the only possible solution left to us is to implement the second approach, to tune classification model with thresholds.

We came up with two ways for threshold tuning - tune by the minimum thresholds and the maximum thresholds, meaning that we predict the class to be unknown if the predicted probabilities of other classes are all below this threshold or are all above this threshold. In order to get the appropriate value of the threshold, which results in the highest prediction accuracy, we turned the accuracy from 0.01 to 0.99 with increment of 0.01in each iteration. However, we found that the outcome is not as good as we expected.

	precision	recall	f1-score	support
14-16	0.81	0.18	0.30	13100
24-26	0.72	0.02	0.04	17298
34-36	0.00	0.00	0.00	2584
44-46	0.00	0.00	0.00	551
?	0.27	0.95	0.42	11799
avg / total	0.58	0.31	0.21	45332

Table 3



Graph 4

The accuracy of the predictions of the unknown class is very high (0.95 as shown in Table 3), but the performances of other classes are deteriorated as threshold increases. The reason behind that is the characteristic of instances of the unknown class are actually hard to be defined into one class, as the unknown class consists of different age ranges (0-13, 17-23, 27-33, 37-43, >47). Hence, predictions of the unknown cass could be very close to those of the normal 4-classes, like the instance of age 17 which belongs to unknown class could be really close to those instances in the class 14-16. As a result, by assigning a single threshold may potentially confuse the classifier regarding the difference between unknown class and other classes. The model predicted too many instances which are actually not unknown class to be unknown class.

Inspired by the idea that the unknown class consists of different age ranges, we came up with the tuning of threshold ranges. Each threshold range intend to represent a range of possibility of each age group. We derived these ranges by calculating the possibility range of each age group from the raw dataset instances(e.g. 0.155 - 0.518 for 14-16 as shown in Graph 4)

We labeled the instance as unknown if the predicted possibility of each age group does not reside in its corresponding possibility range. Thus, the issue of classifying instances wrongfully into unknown class is solved. This threshold handling method is proved by improvement in accuracy.

Experiment Result and Improvement

Graph 5

In overall, stacking produced the highest accuracy, and KNN produced the lowest accuracy. When looking at the results of this experiment, it is important to note again that vectorisation of the dataset using n-gram representation leads to a very sparse matrix/vector with high dimensions. One improvement in the future could be using principal component analysis(PCA) to reduce dimension of the matrix. In addition, the size of training data is relatively large. Another improvement is to check for overfitting of our model and investigate methods for checking overfitting; for example, the accuracy of the model when evaluating develop data is 71%, but Kaggle yields the result to be 64%.

Summary

In overall, this report outlines our approaches to build four machine learning systems for short text classification using n-gram representation for age prediction. With choices of data processing and experimental design as discussed in this report, the maximum accuracy we achieved on the development data was 71%.

Reference

1.

StackingCVClassifier. (n.d.). Retrieved from https://rasbt.github.io/mlxtend/user_guide/classifier/StackingCVClassifier/

2.

Cavnar, W. B., & Trenkle, J. M. (n.d.). *N-Gram-Based Text Categorization*.

3.

Nigam, K., & McCallum, A. (n.d.). *A Comparison of Event Models for Naive Bayes Text Classification*.

4.

Schler, Jonathan, Moshe Koppel, Shlomo Argamon, and James Pennebaker (2006) Effects of Age and Gender on Blogging. In Proceedings of 2006 AAAI Spring Symposium on Computational Approaches for Analyzing Weblogs. Stanford, USA.

5.

Sun, A. (2012). Short text classification using very few words. *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR 12*. doi:10.1145/2348283.2348511