Practice IV

Clickbait detection
First stage

Introduction

- Clickbait is a widespread phenomenon in online news
- It is a way of creating headlines and teasers aimed at capturing readers' attention in order to increase traffic
- The function of informing is relegating to a secondary role

Introduction

- The definition of clickbait for this practice is as follow
- “*Clickbait is a method for generating teasers, especially online, that deliberately omits part of the information with the goal of generating curiosity by creating an information gap, thereby attracting the readers' attention and making them click*”

Objective

- Create a machine learning model for detecting whether a text is a clickbait or not

Specification

In teams of 3-4 members do the following:

1) Load the corpus *TA1C_dataset_detection_train.csv*

   - *Teaser Text* column will be used as feature
   - *Tag Value* colum will be used as target (class)

2) Apply text normalization to the feature value

3) Split the corpus in *train and dev* sets using 75% for training and 25% for development

4) Create different text representations. You could try different n-gram ranges

5) Use different machine learning methods to train a model and predict *dev* set instances

6) Make the necesary adjustemnts to the model to improve perfomance over the development *set*

7) Use the best adjusted model to predict instances of *test* set available in the *TA1C_dataset_detection_dev.csv* file

Text normalization

- It is recommended to apply different normalization techniques to the feature value
- The normalization process could include some of the following steps: tokenization, text cleaning, stopwords and lemmatization
- It is important to try different combination of the above steps

Corpus split

- Use *train_test_split* function of scikit-learn
- Instance of corpus should be shuffled (*shuffled* = *True*)
- Set the the fixed random seed equals to 0 (*random_state=0*)
- Activate the stratify function (*stratify=y*)

Cross validation

- You must use a cross validation method over the train set
- Set a stratified 5-fold cross validation (*StratifiedkFold(n_splits=5) or cross_val_score(estimator, X, y, cv=5, scoring='f1_macro'*)

Features and text representations

- You could try unigram, bigram, trigrams and combinations of them to extract features
- You could use Binary, Frequency and TF-IDF text representations
- It is also recommended to explore the use of TruncatedSVD to reduce dimensionality and enrich features

Machine learning methods

- Try different traditional machine learning (ML) algorithms
- You could use the same methods implemented in the previous practice
- In addition you could try the ensemble methods as:
  - Random forest (RandomForestClassifier)
  - Gradient Boosting (GradientBoostingClassifier)
- It would help if you tuned the algorithm hyperparameters to improve the results

Evaluation metrics

- A classification report (*classification_report*) should be applied to the predictions on the dev set for each variation of:
  - Normalization process
  - Text representation
  - Machine learning method
  - Hyperparameters adjustments
- The main metric is f1_macro, this one will be used for selecting the best model

Predictions on the test set

- The best model must be retrained using the full corpus of the file *TA1C_dataset_detection_train.csv* with the selected configurations (normalization, text representation, ML model, hyperparameters, class balance, etc.)
- The retrained model must be used to predict the *Tag Value* of instances in the *TA1C_dataset_detection_dev.csv* file

Predictions on the test set

- The output CSV file must have the following features:
  - Name: *detection.csv*
  - Columns: *Tweet ID* and *Tag Value*
  - Separator character: Comma “,”

Class imbalance

- The distribution of classes in the corpus shows a significant imbalance (71% No clickbait and 29% clickbait)
- Class imbalance often leads to bias in learning models
- It would be helpful to use some methods to balance class distribution
- Undersampling and Oversampling are two of the most common balance methods (https://imbalanced-learn.org/stable/)

Evidence

- Source code
- A report in PDF format describing the following:
  - Task to be solved
  - Selected machine learning methods
  - Adjusted hyperparameters
  - Classification report of each experiment

Evidence

A table describing the experiments performed showing the best configuration of each ML method on the *dev* set

ML method	ML hyperparameters	Text normalization	Text representation	Balance methods	Average f-score macro
Logistic regression	max_iter = 200	Tokenization, lemmatization	Unigram, binarized	None	0.85
Naïve Bayes	default	Tokenization, stopwords	Bigram, frequency	Undersampling	0.88
...	...		...		...
Multilayer perceptron	hidden_layer_sizes = (200, 100)	Tokenization, text cleaning, lemmatization	Unigram + bigram, Tf-idf	Oversampling	0.9

Evidence

- The file *detection.csv*