Problem Statement: News Classification

Large volumes of news articles are received every day from various sources. To simplify downstream processing tasks, one of the first steps is to properly categorize news into well-defined categories. In this problem, I have been given a set of news articles and a set of categories. My task was to develop a model which should be able to predict the category of unseen news.

Preprocessing steps performed on the dataset

1. Loaded data from all the three files given and mapped news ids with category names
2. Checked title, snippet and description from the dataset, and had following observations:
    a. For few of the data points, news description was missing
    b. For few of the data points, snippets are also missing but number of snippets missing is smaller than the number of missing new description
    c. Title is available for all the observations, and title is often more descriptive for the category of the news
3. To treat missing values and going with the intuition for title, following steps I took for getting final text ready:
    a. Replace missing news description with the available snippets, as snippets are also part of the news description, so it will add up to the data instead of removing them
    b. Join title with news description, and keep title first while joining
4. Pre-Process Steps for the final text:
    a. Convert to lowercase
    b. Remove punctuations and special characters
    c. Remove Stop words
    d. Remove URLs
    e. Remove new line symbols and other unnecessary symbols
    f. Remove Digits
    g. Lemmatize words
    h. Remove single letter word which got generated during above pre-processing steps as they don't make any sense
5. To understand data better, did EDA on it:
    a. Plotted a bar plot for the count of data points for each category and observed:
        i. Most of the news were tagged as 'Executive Movement' followed by 'Partnerships'
        ii. 'Mass Hiring' tagged news were least in the given category followed by 'Deals' at the second lowest number
    b. Plotted word cloud for each of the categories
    c. Plotted bar plot for number of characters available in each observation for each category
    d. Plotted bar plot for number of words available in each observation for each category
    e. Plotted KDE plot for number of characters and number of words available in each observation for each category
    f. From all the above three plots, I observed the distribution is same for both number of characters and words
    g. Plotted distplot to understand the average number of words available for each category
6. Created corpus of the words available in dataset
7. Extracted top 10 words appeared most in the corpus
8. Created a function to get top n grams given corpus, number of words and value of g is passed as the input parameter. Plotted bar plot for Unigram, Bigram, and Trigram analysis and most of the words/combination of words appeared makes sense

Feature selection and Helper Functions

1. Created train and test data by keeping 20% of the data for testing the model performance and remaining for training
2. Tokenized and encoded text of train and test data and kept maximum length of 256 vectors, padded sequence for the texts if they are smaller than 256 (For BERT Classifier)
3. For target variable, did One hot encoding
4. Extracted features using TF-IDF techniques with number of maximum features to be 10000 for other type of classifiers
5. Created two helper function to evaluate the performance of the model trained:
    a. Created function to return precision, recall, f1-score and accuracy of the model
    b. Created another function to plot confusion matrix

Different types of models tested

1. I did configuration to train BERT models on TPUs if available (I was using Kaggle so was helpful to fasten the process)
2. Trained Different Models:

Model Type	Accuracy (%)	Precision	Recall	F1-Score
Multinomial Naïve Bayes	67.9	0.686	0.679	0.679
Decision Tree Classifier	65	0.652	0.65	0.65
Gaussian Naïve Bayes	50.8	0.494	0.508	0.508
Stochastic Gradient Descent Classifier	82.56	0.827	0.826	0.826
Light Gradient Boosting Classifier	84.22	0.843	0.842	0.842
BERT Classifier	93.57	0.937	0.936	0.936

3. Out of all the models trained, BERT looks more promising in comparison to other models

Model Tuning techniques used to give best results

Right now, all the models are trained with default parameters, except the BERT classifier, which is trained by designing a custom TextCNN model.

Any other comments/experiment details

`Note`: Things we can do more to improvise the work or to be assured about the best model:

- Experimented with different embeddings (`TF-IDF` or `BERT Embeddings` or `Glove Embeddings`) and different models (SVM, CatBoost, LSTM with Attention, Bidirectional GRU) combinations
- Parameter Tuning, like:
    - Grid Search for TF-IDF Parameters
    - Feature weights
- Evaluate on different combination of metrics (weighted/micro)
- Cross validation and hyper parameter tuning of the models trained
- Use of mlflow and papermill to do multiple experiments and track all of them from one UI
- Building pipeline and code scheduling if planning for putting the code in production