COMP90051 Statistical Machine Learning
Project 1 - Group 14

1. Introduction

It was proposed to develop a machine learning model capable of classifying whether given text instances were generated by a human or a machine. We were given pre-processed data, that is tokenized and mapped to indices from 0 to 83,582. The training data consists of two distinct datasets from different domains. Domain 1 comprises 5,000 samples evenly distributed between human-generated and machine-generated text. Domain 2 dataset consists of 13,000 samples with a significant class imbalance: 1,500 human-generated samples and 11,500 machine-generated samples. The test dataset includes texts from both domains, evenly split, with labels unknown.

The primary objective of the project is to build a model capable of handling samples from different domains and addressing the challenge of imbalanced data distribution presented in Domain 2. To solve this problem, a methodological approach where feature engineering and machine learning capabilities are mixed up is presented.

2. Domain impact

It is clear from the image below that the difference in the domain distribution was mainly due to the unbalance in the second dataset and the probability distribution of the features.

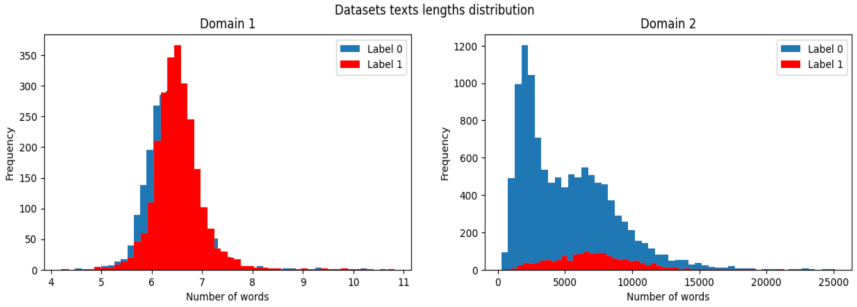

Image 1. Features token frequency by Domain

To address such problems the intended solution tried to eliminate these 2 differences in the datasets by **resampling** the second dataset in order to even the label observations, as well as **encoding the features** of both datasets in a way that it was possible to make general predictions over the imputed data regardless of the domain. In the next section both problems are addressed and described.

3. Methodological approach

This section the final solution approach will be methodologically described trying to decompose the problem into the main aspects considered during the project.

Data sampling and data Augmentation

To mitigate the effects of undersampling and disparate domain distributions, data from both domains were combined. This approach augmented the training data size and increased the samples for each label, thereby enhancing the predictive generalization capabilities of the tested models.Regarding sampling techniques, directly incorporating a weighting strategy into the models yielded better outcomes. For instance, with models developed using the sklearn library, the *'class_weight="balanced"'* method was employed, while for PyTorch, a custom weighting function was formulated and integrated into the model via the *dataloaders* method *sampler*.

Contextual representation (BoW and N-gram)

The contextual representation served as an excellent feature extractor for the datasets, successfully addressing the disparities in probability distributions across domains. After many iterations, the best training results obtained were a combination of BoW and N-gram models. In our case a *CountVectorizer* representation with *ngram_range* set to *7-8*, *up to 8-11* had a great feature extraction power. This came at the cost of a very large number of features to be fed into the models with resulting datasets with over 1.000.000 features for the training models.

Model selection

After discharge some dimensionality reduction models tested seeking the highest Accuracy scores, the most promising results were achieved by employing sparse matrices for both ML and DL models, in the case of ML the '*sklearn'* modules used accepted sparse matrices as an input, and in the case of DL the sparse matrix was passed to the model through the dataset class definition for the model dataloaders.

The final selected models for comparison are described below:

- Logistic Regression: Default parameter weighted logistic regression classifier.
- LinearSVC: Weighted linear support vector classifier with a strong regularization of C=0.08 and a dual optimization.
- RidgeRegression: Default parameter Ridge classifier with *sparse_cg* solver for sparse matrices.
- BoW-NN: Contextual representation of the datasets passed through a feed forward neural network with 2 hidden layers and 64 hidden nodes.
- DANN-BiLSTM: Bidirectional Long Short Term Memory (BiLSTM) used as a feature extractor over a Domain Adversarial Neural Network (DANN) as a class classifier.
  The model had 3 sections:

I. Feature encoder: BiLSTM with 2 hidden layers, 256 hidden nodes and 128 encoding dimensions for the first layer.
II. Classifier: 1 layer sequence for label classification.
III. Discriminator: 2 layer sequence for domain classification.

- Ensembling:Majority Voting ensemble of the 3 highest scoring models in the kaggle public score.

The best obtained model positioned us in the **2nd position in the public leaderboard** at the time of the close of the competition. The main reason for such a result is related to the contextual representation of the features, where the combination of the CountVectorizer with the n-grams of each text retains the most important features of each text class making them easily differentiable. Bellow there is a detail of the obtained results:

Table 1. Models performance

Model	Accuracy	F1 Score	Roc AUC	Kaggle Score
Weighted Logistic Regression	0.933	0.933	0.933	0.925
Support Vector Machine	0.933	0.933	0.933	0.926
Ridge Classifier	0.928	0.929	0.897	0.922
BoW + FFNN	0.928	0.929	0.928	0.914
BiLSTM + DANN	0.730	0.674	0.730	0.730
Ensemble (Top 3 models)	**-**	**-**	**-**	**0.934**

As evidenced in the table the worst performing model was the BiLSTM DANN, the main reason for this is the model input, due that, while the other models were receiving a contextual representation of the tokens, this model was using the BiLSTM as a feature encoder.

In the case of the BoW-FFNN, although this model presented good results, the time consumption of this model made it difficult to tune its results to make it more accurate in the final predictions.

As of the other models, they were chosen because there was not much difference in results among these linear separation models, and models with higher complexity in terms of accuracy, and the time execution of such complex models were significantly superior to these models. The best performing models both in terms of accuracy and time were Linear SVM, logistic Regression, and Ridge Classifier.

Finally, a majority ensemble was done on these models predicting over the test labels and joining the predictions with a majority voting decision rule.

Further developments may explore other encoder techniques that generalize the extracted features from both domains. Also, deeper NN or Cross validation techniques could further improve the obtained results.

4. Other approaches

There were also some different approaches that we tested before getting to the depicted solution, some of them are:

- **Sampling techniques** to address data imbalance before passing the data through the models.
- **Dimensionality reduction** over the BoW sparse matrices to catch the most important relations.
- Run **cross validation** over the selected models to tune models hyperparameters.
- N-gram models over the test set to unmask the unknown tokens.
- Different Neural Network methods for **Domain Generalization** and **feature encoding**.