Binary Classification of Populist Discourse: Outline

June 2024

Phase 0: BRIEFING AND DEFINITIONS
[Alessandro Pala, Greta Grelli]
The group is introduced to populism and chooses the two most cited recent academic papers for definitions. The populist label is assigned to discourses showing at least two of these traits: mentions "the people" as a unity, contrasts "the people" with an "elite", evokes feelings of injustice, or is radically anti-establishment. A list of probable dataset locations and figures showcasing populist speech is provided. Non-English samples are translated for better accuracy.

Phase 1: WEB SCRAPING AND LABELLING
[All members, equally divided]
Manual web scraping extracts speeches for the dataset, labeling each as populist [1] or not [0]. Only politically-related examples are kept, avoiding non-political speeches in order not to train the model to merge political speech with populist speech, making a better model while possibly decreasing accuracy. The dataset is composed by 500 balanced samples, half populist and half non-populist.

Phase 2: IMPLEMENT MODELS
[All members, equally divided]
Phase 2.0: Implement database structure and plotting. [Alessandro Pala]
A database structuring class is implemented in Python to set the foundations for easy tokenization code for each subsequent model. The datasets are merged and split into training and testing sets, stratified by the labels.
For each model a custom sequence classifier is defined using its own sequence classifier that applies dropout and forwards the model. Being pre-trained models, the tokenizer is simply imported. **Phase 2.1**: BERT-tiny. [Alberto Calabrese]
Phase 2.2: BERT-large. [Lorenzo Cino]
Phase 2.3: GPT-2. [Giacomo Filippin]
Phase 2.4: RoBERTa-large. [Greta Grelli]

Phase 3: MACHINE LEARNING
[Alessandro Pala]
A one-epoch training function iterates over batches, computing loss based on predictions and true labels. An evaluation function computes accuracy and loss without updating model parameters. A prediction function labels single speeches using the trained model.
Learning rate and epochs are initialized, and training begins, with real-time plotting of loss and accuracy.

Phase 4: OPTIMIZATION AND REGULARIZATION
[All members, equally divided]
An AdamW optimizer with weight decay is used, with a learning rate scheduler for gradual adjustment. Once the final code was assembled, members competed to fine-tune hyperparameters* for optimal model performance. *Learning rate and number of epochs (diff. for small vs. large models), gradient clipping rate, dropout rate, weight decay starting value and update function.