Seminar 2: Mining sequential data

Intelligent Systems (IS)

December 6, 2021

1 Introduction and Overview

In real life, sequential data is common, appearing in text sequences, time series, nucleic acid sequences etc. This seminar tests your ability to model and derive conclusions from sequential data. The assignment has to be submitted in the form of two files: a markdown file and a PDF file and created from the R Studio markdown file (in RStudio → file - new file - R Markdown), where you write both the code, as well as the text of answers (echo = T option must be enabled for each code block). Markdown files can easily be exported to PDF using ("Knit") button in R Studio.

2 Task overview

Documents are collections of ordered strings – string sequences. In the assignment, you shall inspect, transform and learn from this type of input. You are provided a pre-split collection of labeled documents (*train.tsv*, *test.tsv*) related to detection of *fake news*[1]. The task is to i) pre-process the documents, ii) do feature construction, iii) model of the documents, and iv) evaluate. The documents represent snippets of social media text and are labelled as either fake (0) or true (1).

2.1 Pre-processing

The provided data is realistic in the sense that it can be noisy. Your first task is to *clean* the documents of possible noise, including, e.g., URL links, strange symbols etc. Compute some basic statistics (e.g., term frequencies) and visualize them. What do you observe?

(15%)

2.2 Feature construction

Raw texts are not the best form of input to many machine learning algorithms. Your task is to *convert* the documents into feature matrices suitable for learning. Devise a feature construction procedure that outputs a real-valued matrix (rows = documents, columns = features), suitable for learning[2]. Note that you need to transform both the train and the test instances. Discuss your choices and describe the final space you will use for learning.

(40%)

2.3 Modeling

Use at least three machine learning classifiers and one ensemble method. Train the models on the training data (*train.tsv*) and produce the predictions for the test instances (*test.tsv*). Be very careful not to use any of the test set instances during learning (no data leaks allowed!).

(15%)

[1] Read more at `https://arxiv.org/abs/2101.03717`

[2] If you are using an end-to-end learner, ignore this sentence

2.4 Evaluation

Implement the classification accuracy and F1 score, and compute them for individual models from the previous section. Compare the performance with the baselines (same train-test splits) offered below. Plot/visualize the results in an interpretable manner of your choice (e.g, a barplot of classifier-performance tuples) alongside the table of results. Comment on which methods prevail and why? The baselines include a full spectrum of models, from light-weight ones to pre-trained language models[3]. Place your best solution within the context of the baselines (Table 1). Did you consider different hyperparameter configurations? Why (not)? The minimal criterion for you to score any points in this section is that **your model beats the majority classifier**.

- HINT: Feature ranking is potentially very useful when considering high-dimensional spaces.

Table 1: Pre-computed baselines (test set performance).

Baseline	Accuracy	F1
autoML (1h)	0.957	0.956
MPNet + LR	0.939	0.939
char + LR	0.929	0.929
word + LR	0.912	0.911
doc2vec + LR	0.812	812
majority	0.523	0.344
your apprach	?	?

(30%)

2.5 Bonus: Beat the top baseline

If you manage to beat the top-performing baseline (1h of automated model search on 60 threads), or demonstrate that you did your best trying, you deserve the additional bonus.

- HINT: You can combine different representations.

- HINT: You can combine different models.

(10%)

[3]To learn more about the stronger baselines, visit `https://huggingface.co/microsoft/mpnet-base`. Note that LR stands for Logistic Regression.