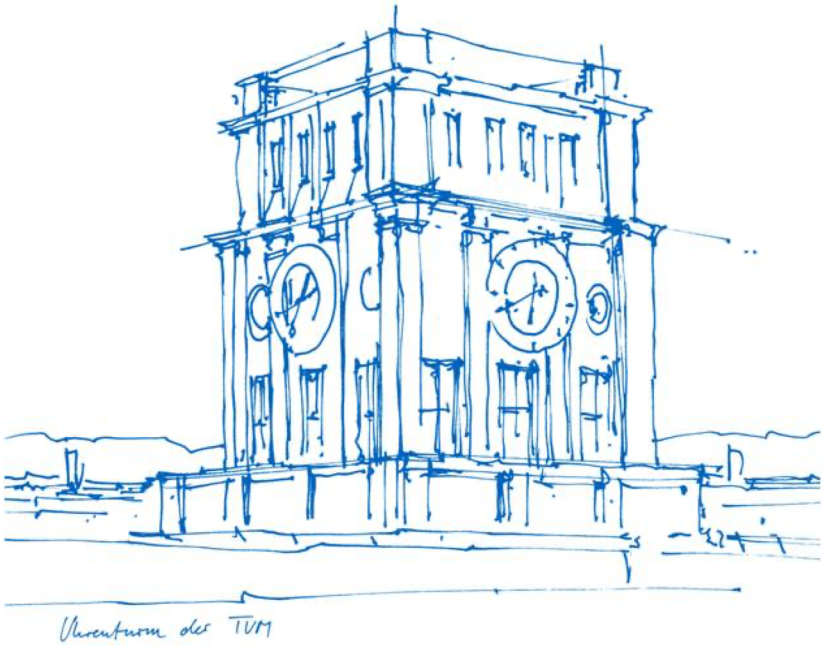
Exercises for
Social Gaming and
Social Computing
(IN2241 + IN0040)

Exercise Sheet 6
Climate and NLP



Exercise Sheet x: Climate and NLP

Goals:

- familiarize with neural topic modeling
- gather experience with neural networs / Tensorflow
- learn about text classification

Data

- climate_tweets.csv
  - tweets related to climate change
  - labeled either as activist, neutral or denier

- toxicity_train.csv, toxicity_test.csv
  - comments from wikipedia
  - labeled by severeness of toxicity

BERT

- Bidirectional Encoder Representations from Transformers (BERT) model : multi-layer bidirectional transformer encoder
- transformer architecture: widely used for various NLP tasks, such as text summarization, Q&A, or translation tasks
- Transformers: identify relations and context in input sequences by using the attention mechanism
- We will use BERT for embedding the input data for our models
- Further reading for interested students: [1], [2], or [2b]

Task 5.0: The Data

5.0.1 Climate Data

Before we can begin with our analysis, we have to load and preprocess the data.

a) Cleaning the data: Use regular expressions in `clean` to remove links ("https" and "//t.co"), usernames and punctuation. In `remove_stopwords` remove tokens of with length < 2, if they are a digit or if they are contained in the stop words.

Hint: Use the library `re` for the regular expressions

5.0.2 Toxicity Data

For prediciting the toxicity score of a tweet, we will use the Toxic Comment Classification dataset , containing comments from wikipedia, labeled by their level of toxicity. Before we can use it for training, we have to preprocess it again.

5.0.3 Final DataFrame

This DataFrame will be used to store the information retrieved from our models.

```
final_df = pd.DataFrame(columns=["text", "topic", "label", "toxicity"])
final_df["text"] = tweets
```

Task 5.1: The Topic Model

- For the topic model we will use BERTopic [3]
- BERTopic uses BERT for embedding the input and clusters the embeddings using hdbscan [4]
- We will use various visualization tools provided by the library for analysing the extracted topics

Task 5.1: The Topic Model

5.1 The Topic Model

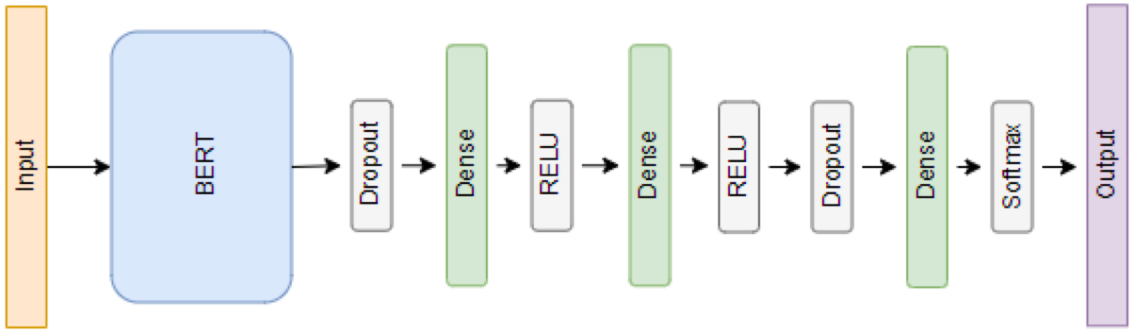
For this task we will utilze the BERTopic library. This library embeds our pre-processed tweets using the BERT model and calculates the topics using hdbscan, a hierarchical version of DBSCAN, and a version of TF-IDF. Further information about BERTopic can be found here.

a) Train the topic model: Use `.fit_transform` on `topic_model` with our pre-processed tweets as input and save the returned topics and probablities for later use.

b) Analyse and adjust the topics: You can find the pre-defined functions here. You can reduce the amount of topics with `.reduce_topics`, try to find a suitable number of topics for our data.

Task 5.2: Text Classification

- In order to identify the class of a tweet we will extend BERT by three feed-forward and dropout layers using Tensorflow
- Use dropout of 0.2
- Feed-forward layers: 256, 64, 3 neurons
- The general architecture for both classifiers can be found below



Task 5.2: Text Classification

5.2 Text Classification

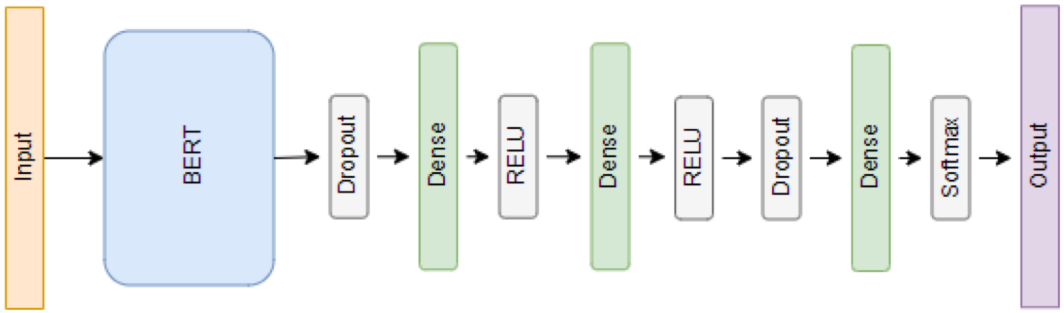
In this task we will make use of the BERT model and append feed-forward layers for classification.

5.2.1

In this task we will implement the function that builds our classifier.

Below you can see the architecture of our model. Append the missing dense and dropout layers with `keras.layers.Dense` and `keras.layers.Dropout` respectively. For the dropout we will use a value of 0.2 and the dense layers will have 256, 64, 3 neurons.

Note: Output corresponds to the last layer and is only present for visualization purposes.



Task 5.2: Text Classification

5.2.2

Since BERT is a rather large model we will use the GPU to speed up training.

a) Build the model: Before we can train the model we have to build it using the previous defined `build_classifier_bert` function.

b) Prepare for trianing: Since we want to predict one-hot encodings of our labels, we will use the `CategoricalCrossentropy` as our loss and the `CategoricalAccuracy` for our metric. You can call them directly from `keras.losses` and `keras.metrics` respectively. Additionally, we want to take a look at the precision and recall during the evaluation. Add the corresponding metrics.

Next we will have to specify the number of epochs and training steps. You can experiment with the number of epochs until you get a satisfying result.

Before we can compile the model, we need to initialize the optimizer. Here we will use a variation of the Adam optimization algorithm that was also used during the training of the BERT model.

Now we can put everything together and compile our model.

c) Train the model: Call the `.fit` method on the classifier, use the training set as input and set the correct amount of epochs. If you want to receive information of the training process set `verbose=1`. After the fine-tuning of our model, test it using `.evaluate` and print out the loss and metrics.

d) Predict the labels: Use the model to predict the labels of our complete dataset.

Categorical Cross Entropy

Multiclass - classification

* Training data: $\{(x^{(n)}, t^{(n)})\}_{n=1}^{N}$

  $x^{(n)}$: D, $t^{(n)}$: K

  with $t^{(n)}$: one of K encoding

$$t^{(n)} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

* Activation function on K output neurons:

$$y_k(x^{(n)}, w) = \frac{\exp(a_k(x^{(n)}, w))}{\sum_{k'=1}^{K} \exp(a_{k'}(x^{(n)}, w))}$$

Softmax

Categorical Cross Entropy

* $p(\underset{K}{z^{(n)}} \mid \underset{D}{x^{(n)}}, w) = \prod_{k=1}^{K} y_k(x^{(n)}, w)^{z_k^{(n)}}$ (Multinoulli = categorial)
  (one of K)

* likelihood $\underset{N\times K}{p(z} \mid \underset{N\times D}{X}, w) = \prod_{n=1}^{N} \prod_{k=1}^{K} y_k(x^{(n)}, w)^{z_k^{(n)}}$

* $-\log(\,) \Rightarrow E(w) = -\sum_{n=1}^{N} \sum_{k=1}^{K} z_k^{(n)} \ln y_k(x^{(n)}, w)$

$$:= -\sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \ln y_{nk}$$

(cross entropy error fct)

Task 5.3: Toxicity Prediction

5.3 Toxicity predictor

For predicting the toxicity of a tweet we will use the same architecture for our classifier as in the previous task.

a) Build the classifier, train and evaluate it

Hint: Due to the high amount of training data, less epochs are needed to achieve a good result.

b) Use the trained classifier for predicting the toxicity labels

Task 5.4: Analysis of our data

- Use the final dataframe for our analysis
- Compute average toxicity by topic by class and store it in a DataFrame
- Compute and print total toxicity and average toxicity by class
- Make final observations backed by the found evidence

Task 5.4: Analysis of our data

5.4 Analysis of our data

Now that we have all the models we need we can start our analysis.

a) Calculate the average toxicity per topic: Iterate over all topics and calculate their average toxicity, the average toxicity per label and the share of the labels.

b) Visualize the average toxicity by topic: Try to find good visualizations. Examples:

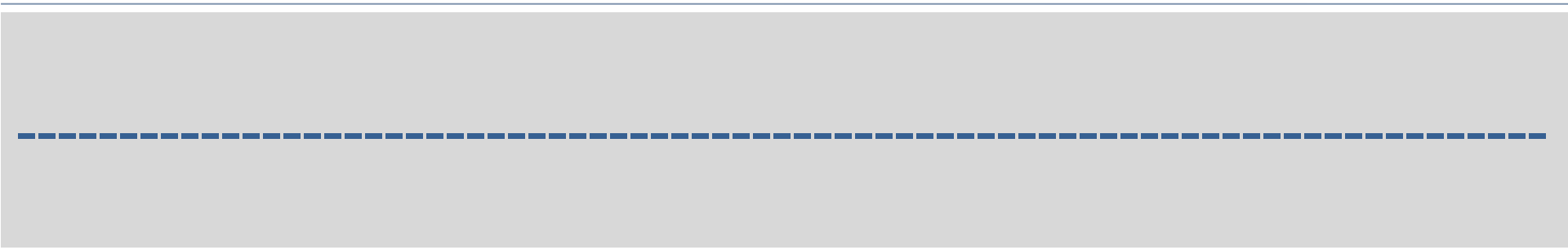
- Use a bar plot for `by_topic`, you can also drop the share of labels per topic, to get a better overview
- Sort topics by their toxicity and plot them

c) Visualize the overall toxicity and by label

d) Discuss the visualizations and try to make assumptions based on these. Which group is more toxic and why? What are the distributions among the data? Are there topics dominated by certain groups?

Submitting your solution

- work by expanding the .ipynb iPython notebook for the exercise that you downloaded from Moodle
- save your expanded .ipynb iPython notebook in your working directory
- submit your .ipynb iPython notebook via Moodle (nothing else)
- remember: working in groups is not permitted. Each student must submit their own .ipynb notebook!
- we check for plagiarism. Each detected case will be graded with 5.0 for the whole exercise
- deadline: check Moodle



Citations

[1] Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

[2] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

[2b] The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning): https://jalammar.github.io/illustrated-bert/ (URL 2022)

[3] Grootendorst, Maarten. "BERTopic: Neural topic modeling with a class-based TF-IDF procedure." *arXiv preprint arXiv:2203.05794* (2022).

[4] Campello, R.J.G.B., Moulavi, D., Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2013. Lecture Notes in Computer Science(), vol 7819. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37456-2_14