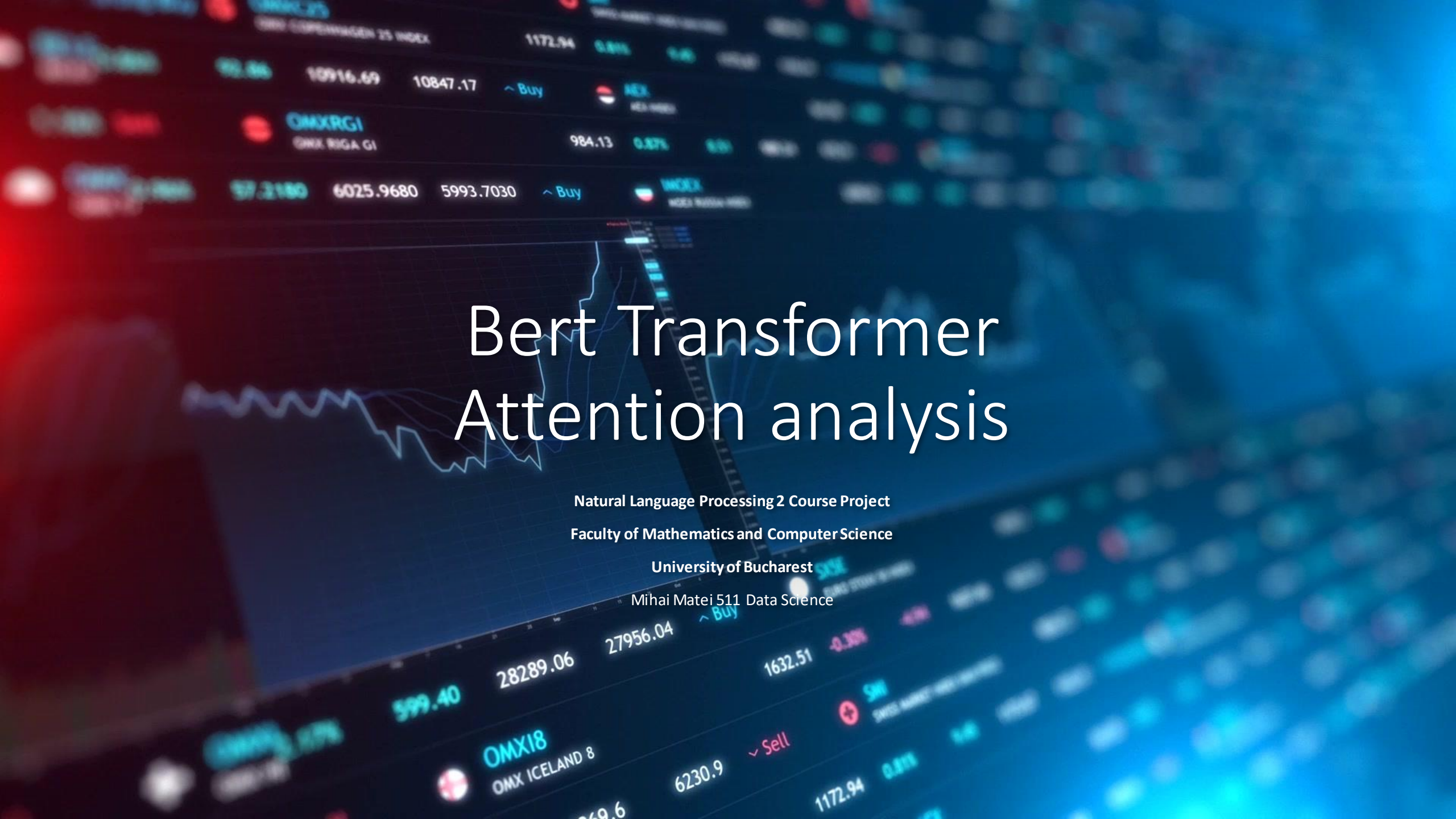


Bert Transformer Attention analysis

Natural Language Processing 2 Course Project

Faculty of Mathematics and Computer Science

University of Bucharest

Mihai Matei 511 Data Science

Attention is All you Need*

Self-Attention Layer**

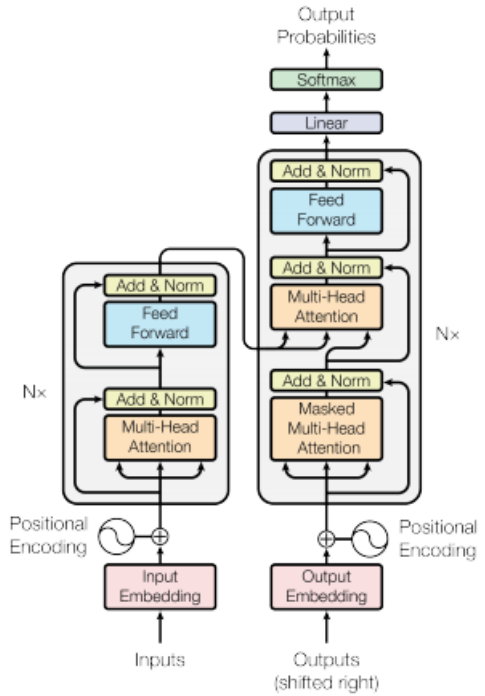
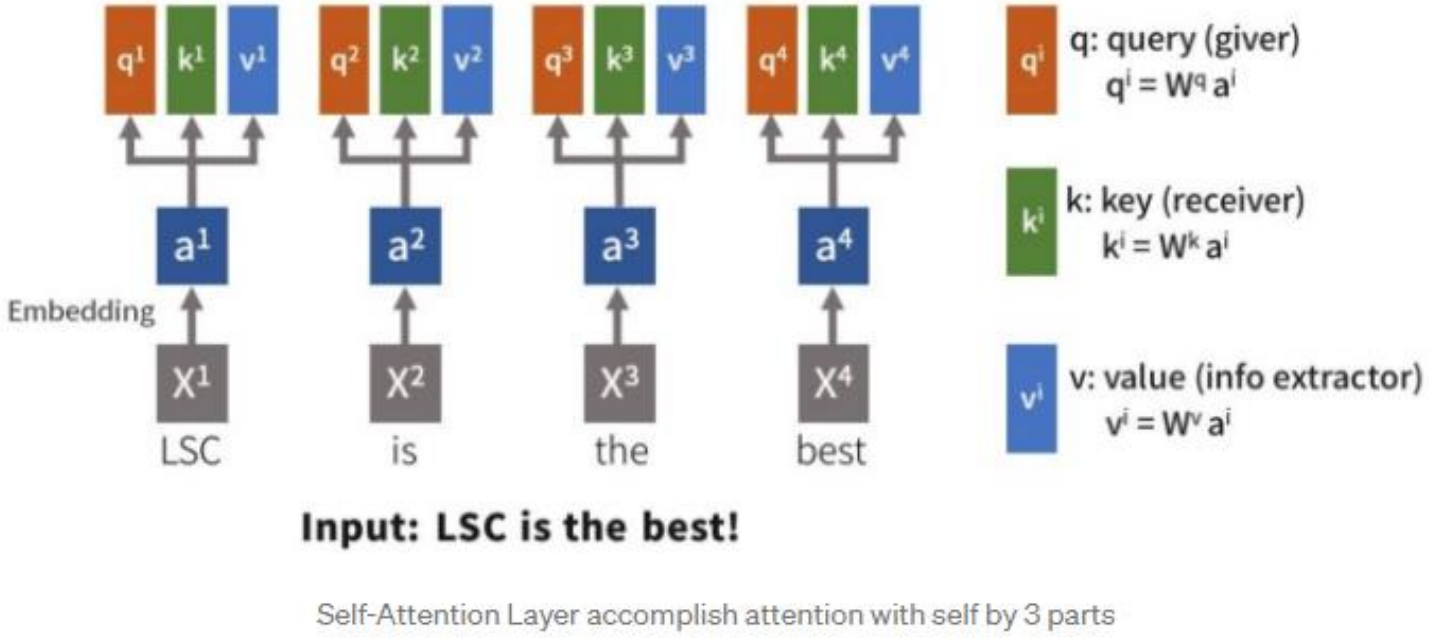
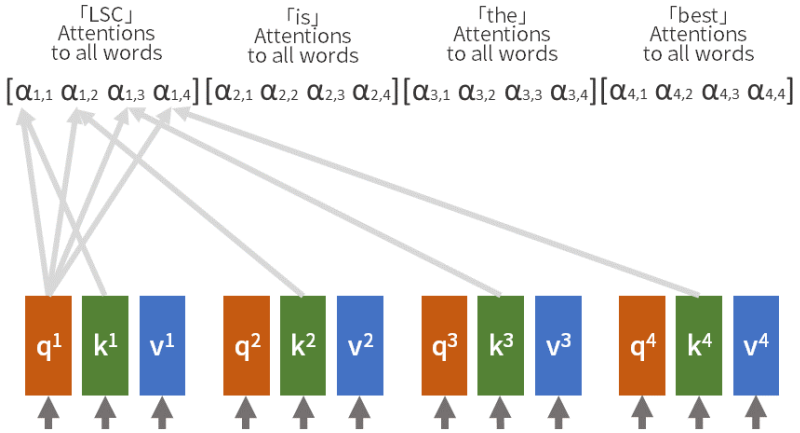


Figure 1: The Transformer - model architecture.



*Google 2017: https://arxiv.org/abs/1706.03762

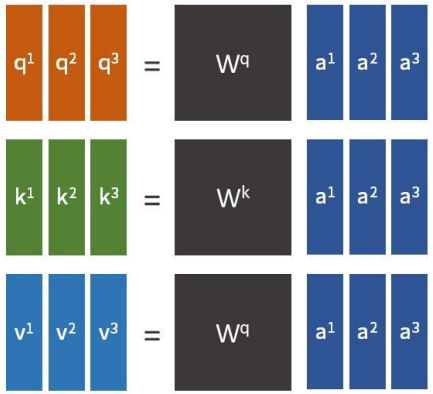
**Taken from: https://medium.com/lsc-psd/introduction-of-self-attention-layer-in-transformer-fc7bff63f3bc

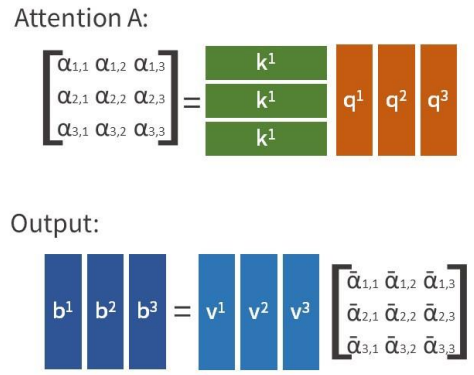


$$\alpha_{i,j} = \frac{q^i \cdot k^j}{\sqrt{d}}$$

d: dimension of q, k

$$A = \text{Attention Matrix} = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} \\ \alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} & \alpha_{4,4} \end{bmatrix}$$



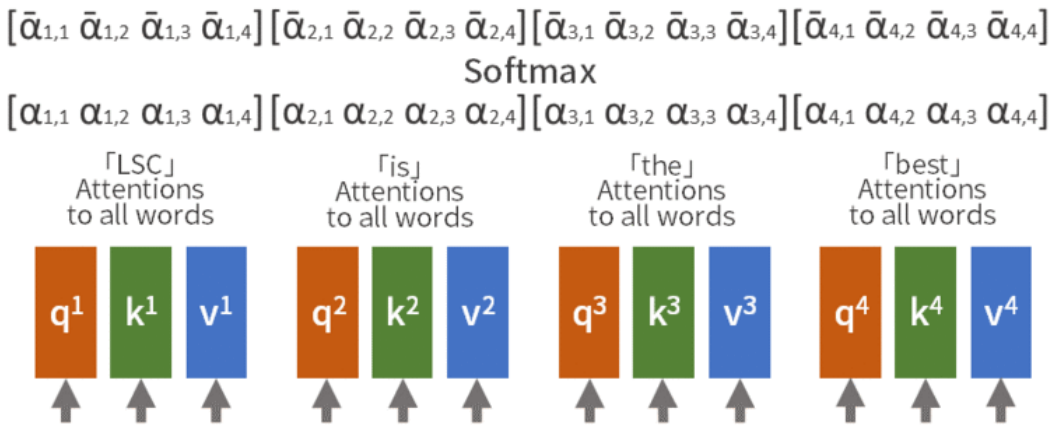


Self-Attention Layer achieved construct relationships and extract information by itself by 3 variance **Query, Key** and **Value.**

$$b^i = \sum_j \bar{\alpha}_{i,j} v^j$$

Query and Key construct the relationships, Value summarize all relations within and concludes an output b which contains relations between input x and all other words.

Matrix form computation – perfect for GPU



Taken from: https://medium.com/lsc-psd/introduction-of-self-attention-layer-in-transformer-fc7bff63f3bc

Google Model Garden

- Vanilla Transformer implementation – Encoder Stack*
  - SelfAttention layer + FeedForward(Dense layers)
  - Configurable number of:
    - Attention heads
    - Embedding size (W matrices)
    - Dense Size
- Bert Base configuration: 12 EncoderStacks (transformers) each:
  - 12 attention heads
  - 768 embedding size
  - 3072 Dense layer size
  - ~ 100K words
- RoBERTA configuration: 4 EncoderStacks

*https://github.com/tensorflow/models/tree/master/official/nlp/transformer

Romanian BIG datasets

Text corpora:

- CC-100 - 16.5GB, webcrawled texts http://data.statmt.org/cc-100/
- Romanian wiki – 0.5GB

https://dumps.wikimedia.org/rowiki/20210101/rowiki-20210101-pages-articles.xml.bz2

Gensim.corpora package:

loading takes 7 minutes, pre-processing 1 hour, **1 Epoch Bert Training 2 hours (NVIDIA 1070TI 8GB)**

Speach corpora*:

Name	Speech Type	Domain	Size [hours]	Availability
RASC [Dumitrescu et al.2014]	Read	Wikipedia Articles	4.8	Public
RoDigits [Georgescu et al.2018]	Read	Spoken Digits	38.0	Public
RO-GRID [Kabir and Giurgiu2011]	Read	General	6.6	Public
IIT [Bibiri et al.2013]	Read	Literature	0.8	Non-Public
N/A [Boldea et al.1998]	Read	Eurom-1 Adapted Translations	10.0	Non-Public
N/A [Popescu et al.]	Spont.	Internet, TV	4.0	Non-Public
RSS [Stan et al.2011]	Spont.	Internet, TV	4.0	Public
SWARA [Stan et al.2017]	Read	Newspapers	21.0	Public
MaSS [Boito et al.2019]	Read	Bible	23.1	Public
N/A [Tarján et al.2012]	Spont.	Broadcast News	31.0	Non-Public
N/A [Suciu et al.2017]	Spont.	Banking	40.0	Non-Public
SSC-train1 [Cucu et al.2014]	Spont.	Radio and TV	27.5	Non-Public
SSC-train2 [Georgescu et al.2017]	Spont.	Radio and TV	103.0	Non-Public
SSC-train3 [Georgescu and Cucu2018]	Spont.	Radio and TV	49.5	Non-Public
SSC-train4 [Georgescu et al.2019]	Spont.	Radio and TV	280.0	Non-Public
CoRoLa [Mititelu et al.2014]	N/A	Various Sources	152.0	Cvasi-Public

Table 1: Romanian speech corpora

*https://www.aclweb.org/anthology/2020.lrec-1.814/

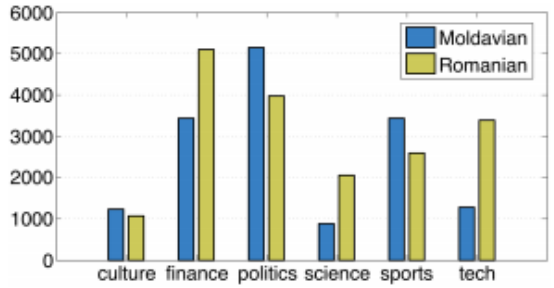
VarDial 2021 - Romanian Dialect Identification (RDI)

https://sites.google.com/view/vardial2021/

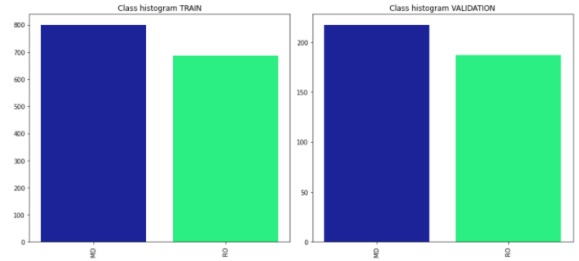
Organizers: Radu Tudor Ionescu and Mihaela Gaman (University of Bucharest, Romania)

*MOROCO dataset: Romanian and Moldavian dialect identification

Texts from 6 categories**:



Loaded 21719 training samples...
Loaded 5921 validation samples...
Loaded 5924 test samples...



Class: MD Sentence: La finele acestei luni, NE NE vor lansa în sfârşit mult aşteptatul NE „NE to NE . Deocamdată, albumul va fi lansat în NE iar în NE prezentarea lui va avea loc la sfârşitul lunii iunie . Albumul va include şi noua piesă „NE loves you", care are deja şi o variantă în limba română . Videoclipul la această piesă a fost creat după o nouă tehnologie – stop motion, constituită în 95 la sută din cazuri din fotografii . Pe lângă aceasta, NE şi NE vor începe şi promovarea noului produs în NE în cadrul mai multor concerte . / / NE de NE

Class: RO Sentence: NE este cel mai mare zoolog român al tututor timpurilor . Pasiunea pentru cercetarea naturii o primise de mic, într - un liceu din NE de la profesorul NE NE . Doi dintre elevii săi au fost NE NE şi NE NE . Întors din NE tânărul NE NE ajunge în faţa NE NE NE pe care îl uimeşte cu ideile sale . Este numit la nici 25 de ani în funcţia de director al pescăriilor statului şi revoluţionează acest domeniu . Dincolo de studiiile pe care le realizează, face o serie de propuneri excepţionale care vor dubla cantitatea de peşte, recoltată, fără ca ecosistemul să sufere . Observă însă şi viaţa grea a pescarilor din NE şi îi propune lui NE NE să îl însoţească într - o astfel de călătorie, convingându - l că statul trebuie să realizeze lucrări de decolmatare, să taie noi canale şi să contribuie la bunăstarea pescarilor . Antipa se dovedeşte mai mult decât un cercetător al naturii, apreciind corect oportunităţile economice ale zonei . Ideile lui NE au fost la baza pescuitului românesc, până în 1965, când regimul decide introducerea politicii de desecare, îndiguire şi canalizare, care au distrus sute de mii de hectare de ecosistem .

*https://github.com/butnaruandrei/MOROCO

**https://www.aclweb.org/anthology/P19-1068.pdf

Pre-process text data

Own **matmih** library for model building, evaluation plot and NLP processing

```
all_data = pd.concat([trainDF, valDF, testDF]).reset_index()
preprocessor = mm.PreprocessPipeline(all_data, 'romanian', max_words=128, min_words=5, min_word_count=5).process([
    'lower', 'split_sentences', 'tokenize', 'remove_punctuation',
    'build_vocabulary', 'to_vocabulary_ids', 'filter_rows'])

all_data = preprocessor.DF
VOCAB = preprocessor.VOCAB
VOCAB_R = {v:k for k, v in VOCAB.items()}
VOCAB_SIZE = max(list(VOCAB.values()))

print(f"Vocabulary size: {VOCAB_SIZE} \t\tData Size: {len(all_data)}")

mm.PlotBuilder().create_subplots(1,1, (13, 6)).create_histograms(
    [all_data['text'].apply(len)], ['Number of words after preprocessing']).show()
```

```
Vocabulary size: 8289           Data Size: 29542
```

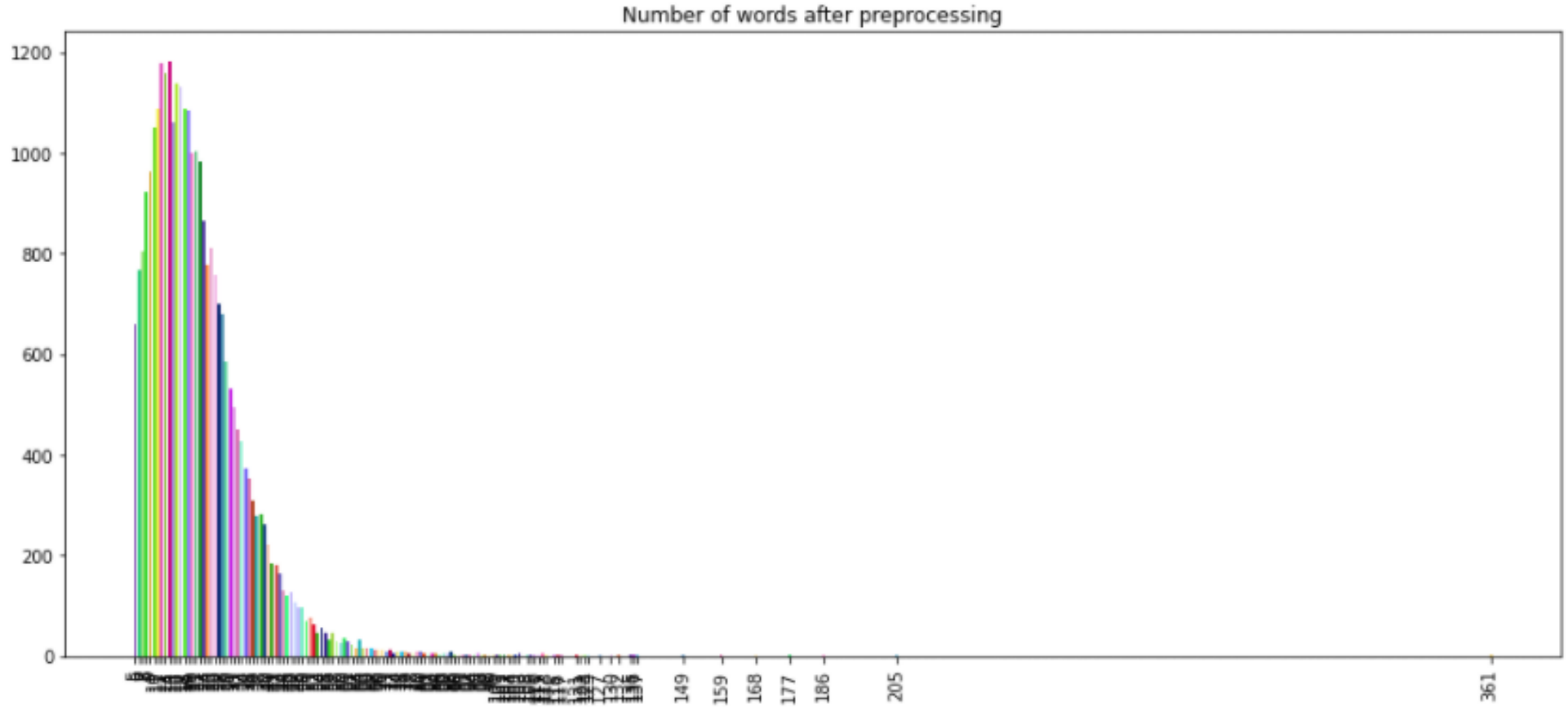
Use only MOROCO **culture** category!!!

Sequence size limits

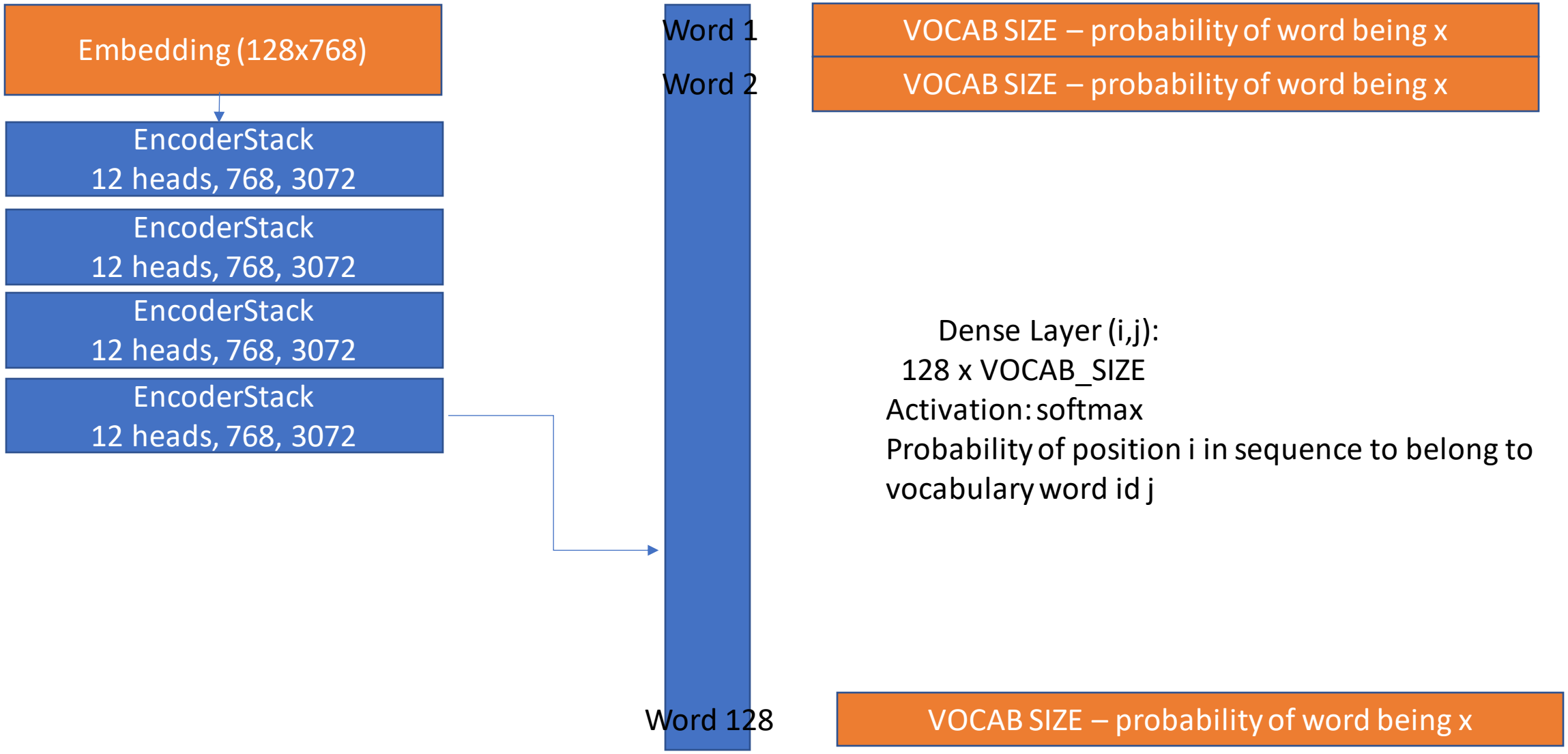
Transformer limitation: sequence size has to be fixed – set to **128 tokens**
Tensorflow2 limitation: using batch requires data to be same size

===> split each text sample into sentences:

===> pad data to 128 tokens with 0



Transformer Language Model



Language Model Training

Model learns to predict missing words in a sentence

Randomly mask 15% of the input sentence
at each epoch

Compute loss only on the masked data!!!

Using tensorflow sample_weight to care about
Only masked words prediction...

```
class LanguageModel(tf.keras.Model):
    def __init__(self, inputs, output):
        super(LanguageModel, self).__init__(inputs, output)
        self._loss = tf.keras.losses.SparseCategoricalCrossentropy(reduction=tf.keras.losses.Reduction.NONE)
        self._metrics = [tf.keras.metrics.Mean(name="loss"), tf.keras.metrics.Accuracy()]

    def train_step(self, inputs):
        features, labels, sample_weight = inputs

        with tf.GradientTape() as tape:
            predictions = self(features, training=True)
            loss = self._loss(labels, predictions, sample_weight=sample_weight)

        trainable_vars = self.trainable_variables
        gradients = tape.gradient(loss, self.trainable_variables)
        self.optimizer.apply_gradients(zip(gradients, self.trainable_variables))

        self._metrics[0].update_state(loss, sample_weight=sample_weight)
        self._metrics[1].update_state(labels, tf.math.argmax(predictions, axis=2), sample_weight=sample_weight)

        return {"loss": self._metrics[0].result(), "accuracy": self._metrics[1].result()}
```

masking data...
<p> aceeasi <p> seara de seara <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p>
<p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p>
<p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p> <p>
<p> <p> <p> <p> <p> <p>

<m> <m> <m> seara de seara <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m>
<m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m>
<m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m>
<m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m> <m>
Masked words: aceeasi

[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 00 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

What should we mask???

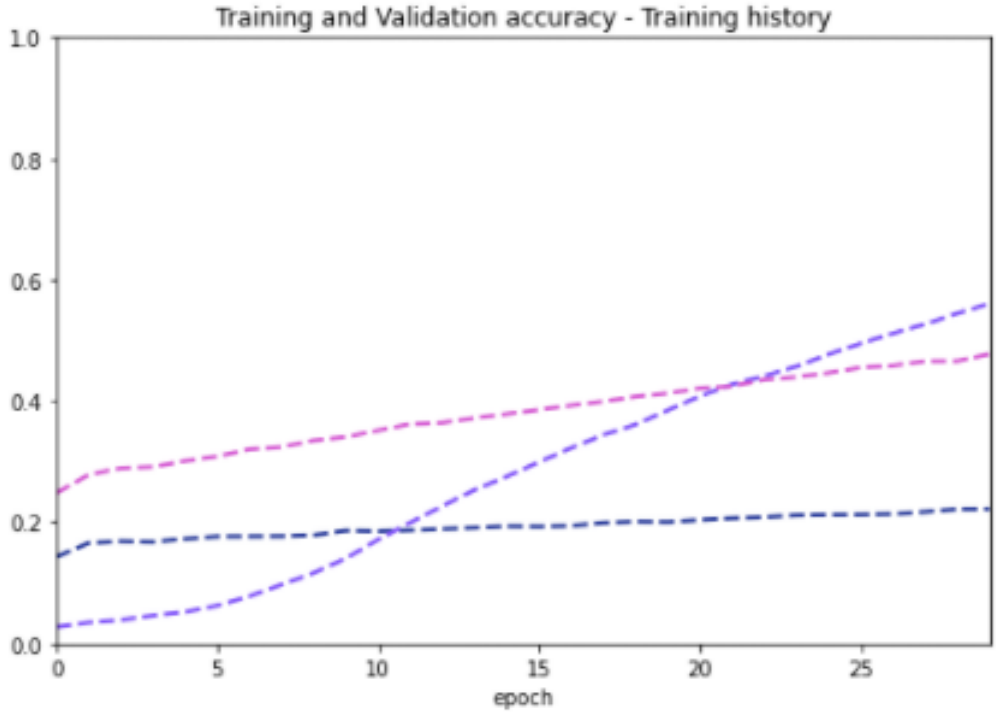
3 different choices tried:

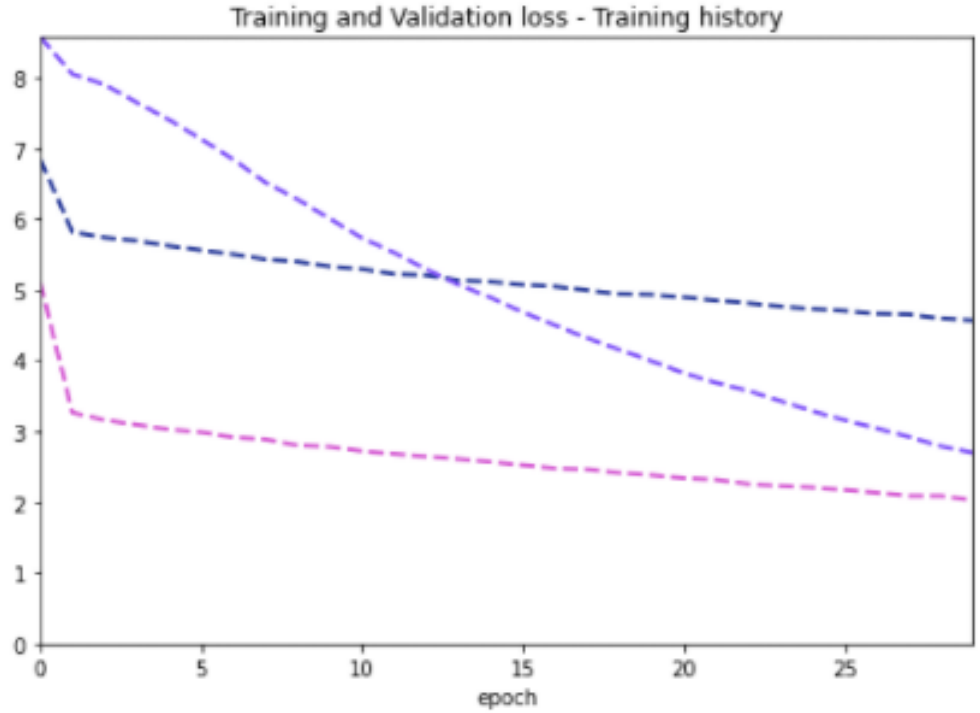
- Random masked 15% of sentence words 0.22 accuracy
- Random masked words with length greater than 5 0.56 acuracy
- Random masked words with length less than 4 0.48 accuracy

```
TRAIN_EPOCHS=30

lm_hyper_lookup = mm.HyperParamsLookup(lambda hyper_params: TransformerModel(**hyper_params),
                                       lambda hist: np.min(hist.history('loss', mm.DataType.TRAIN)))

tf.keras.backend.clear_session()
lm_hyper_lookup.grid_search(all_data, log=True, save_checkpoints=True,
                            train_epochs=[TRAIN_EPOCHS],
                            mask_words=[ # mask random words
                                       lambda wids: range(len(wids)),
                                         # mask random words with length greater than 5
                                       lambda wids: [min(i, 127) for i,w in enumerate(wids) if len(VOCAB_R.get(w, "")) > 5],
                                         # mask random words with length lower than 4
                                       lambda wids: [min(i, 127) for i,w in enumerate(wids) if len(VOCAB_R.get(w, "")) < 4]
                                       ],
                            vocab_size=[VOCAB_SIZE],
                            encoders=[4],
                            attention_heads=[12],
                            embedding_size=[768],
                            filter_size=[3072],
                            optimizer=[partial(create_optimizer, len(all_data), 3e-4)])
```

train_language_model took 6:13:24.159259

Language Model training results






```
Loading initial weights ./best_model_32dc45c8-f036-4e47-8aab-102afadea9cb.save
masking data...
------------ORIGINAL------------
actorul a fost condus pe ultimul drum în aplauzele celor prezenți
-------------MASKED-------------
actorul a fost condus <m> ultimul drum în aplauzele celor prezenți
----------PREDICTED MASK--------
pe
```

EncoderStack Attention

Override Encoder Stack to save self-attention outputs

```python
from official.nlp.transformer import attention_layer, transformer

class MyEncoderStack(transformer.EncoderStack):
    """Subclass transformer to be able to save self-attention output"""
    def __init__(self, params, save_attention=False):
        super(MyEncoderStack, self).__init__(params)
        self._save_attention = save_attention
        self._attention = []

    @property
    def attention(self):
        return self._attention


    def build(self, input_shape):
        super(MyEncoderStack, self).build(input_shape)
        if self._save_attention:
            for layer in self.layers:
                self._attention.append(tf.Variable(tf.zeros(input_shape[1:]), dtype=tf.float32))

    def call(self, encoder_inputs, attention_bias, inputs_padding, training):
        """Override the call method to save the attention if needed"""
        for n, layer in enumerate(self.layers):
            self_attention_layer = layer[0]
            feed_forward_network = layer[1]

            with tf.name_scope("layer_%d" % n):
                with tf.name_scope("self_attention"):
                    encoder_inputs = self_attention_layer(
                        encoder_inputs, attention_bias, training=training)
                    if self._save_attention:
                        self._attention[n].assign(encoder_inputs[0])
                with tf.name_scope("ffn"):
                    encoder_inputs = feed_forward_network(
                        encoder_inputs, training=training)

        return self.output_normalization(encoder_inputs)
```

Plot attention output as heatmaps

```python
# get all attention output data from the encoder
try:
    encoder = model._model.layers[-3]
    encoder_layers = encoder.layers
except:
    encoder = model._model.layers[-2]
    encoder_layers = encoder.layers
attention = [np.mean(encoder.attention[layer].numpy(), axis=1) for layer in range(len(encoder_layers))]
size = len(df_text['text'][0])
attention = [a[:size] for a in attention]

d = pd.DataFrame(data=attention, columns=df_text['text'][0])
mm.PlotBuilder().create_subplots(1,1, (18,4)).create_heatmap(
    d, "Encoder attention per layer", vmin=0, vmax=1.0).show()
```

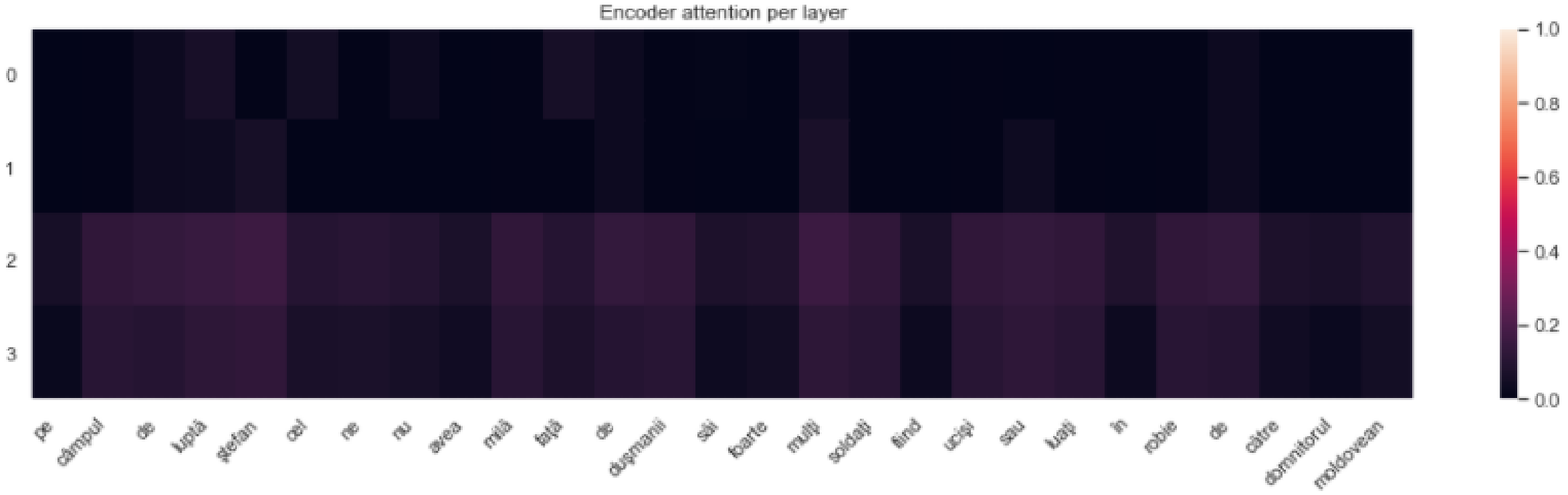
Mean attention 128x768 to 128

Use 4 tf.Variables to save the output at each of the 4 EncoderStacks self-attention

Attention heatmaps

Pe câmpul de luptă, Ştefan cel NE nu avea milă faţă de duşmanii săi, foarte mulţi soldaţi fiind ucişi sau luaţi în robie de către domnitorul moldovean

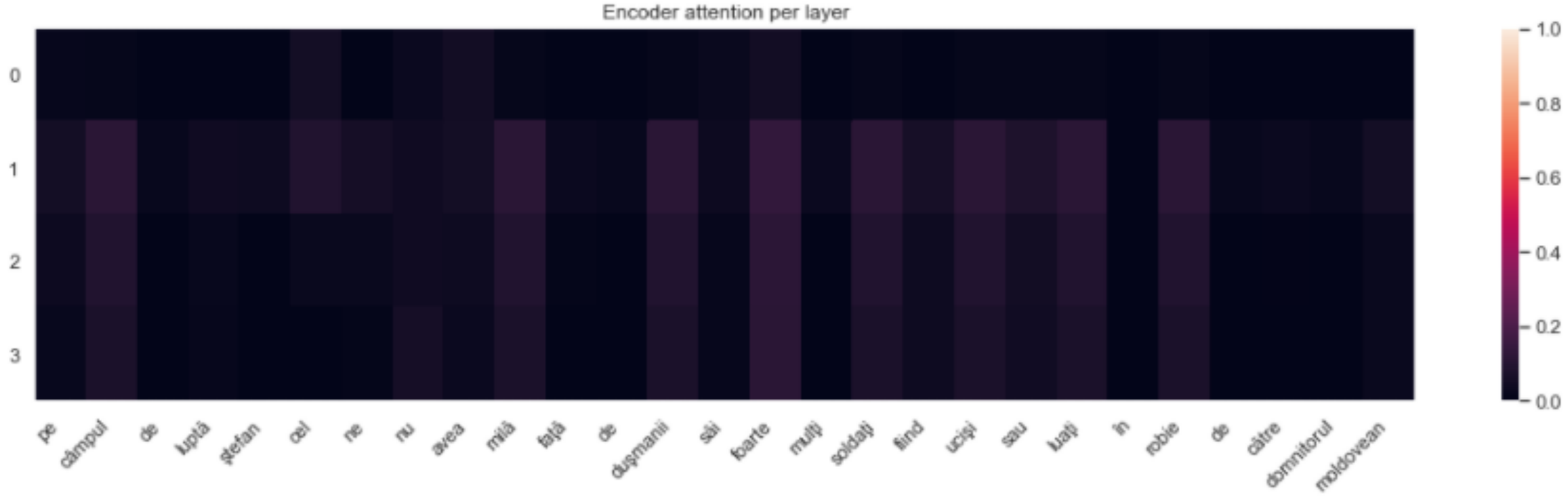
Random masked 15% of sentence words



Attention heatmaps

Pe câmpul de luptă, Ştefan cel NE nu avea milă faţă de duşmanii săi, foarte mulţi soldaţi fiind ucişi sau luaţi în robie de către domnitorul moldovean

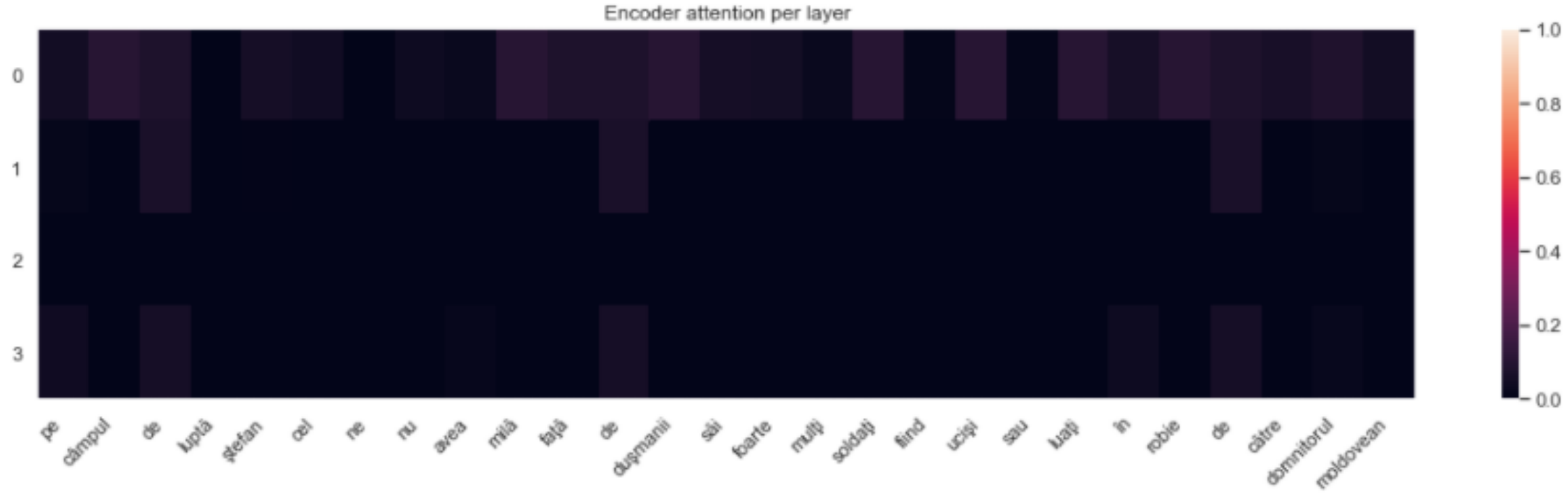
Random masked words with length greater than 5



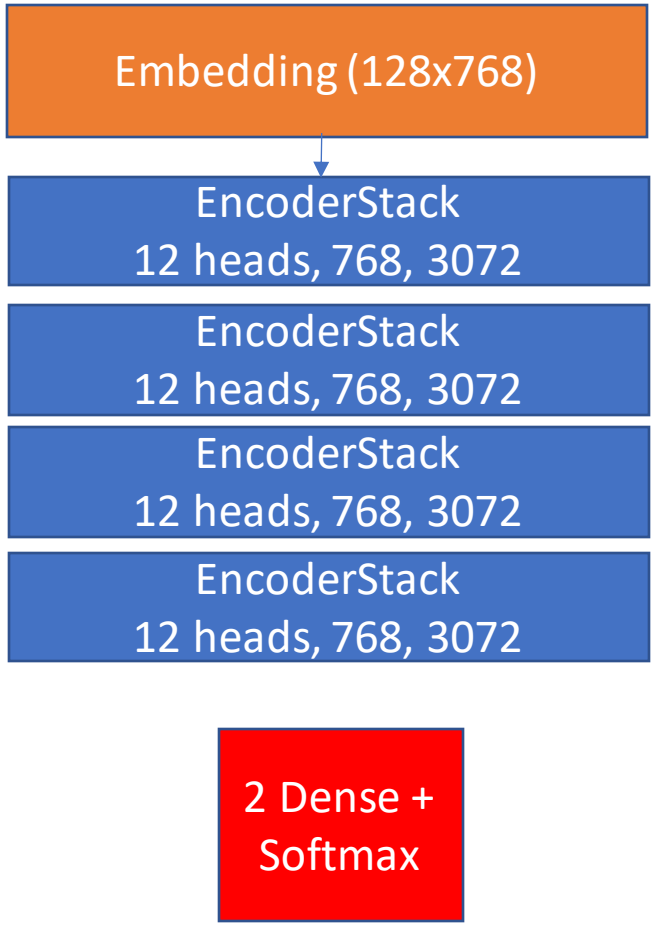
Attention heatmaps

Pe câmpul de luptă, Ştefan cel NE nu avea milă faţă de duşmanii săi, foarte mulţi soldaţi fiind ucişi sau luaţi în robie de către domnitorul moldovean

Random masked words with length less than 4



Transfer learning – Classifier model for dialect prediction



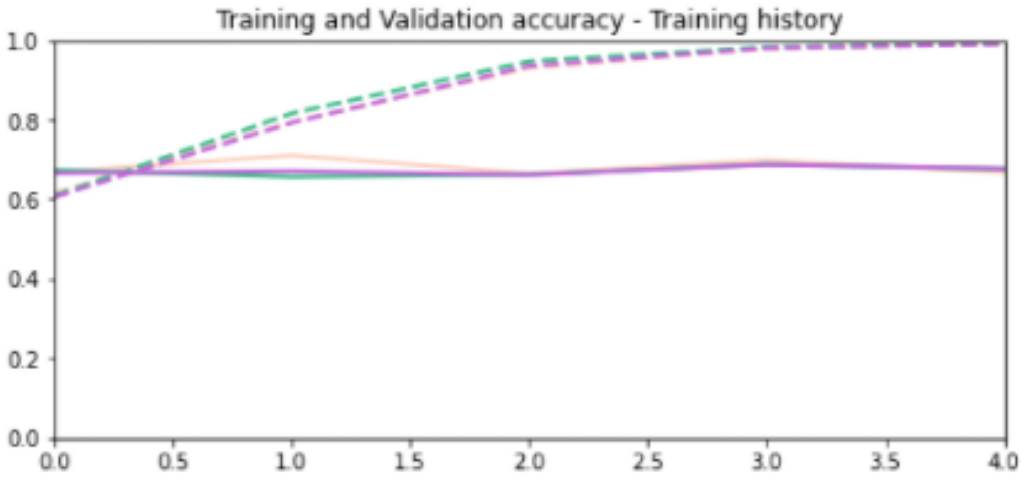
Load weights from the 3 learned language models
Do transfer learning for each of them...

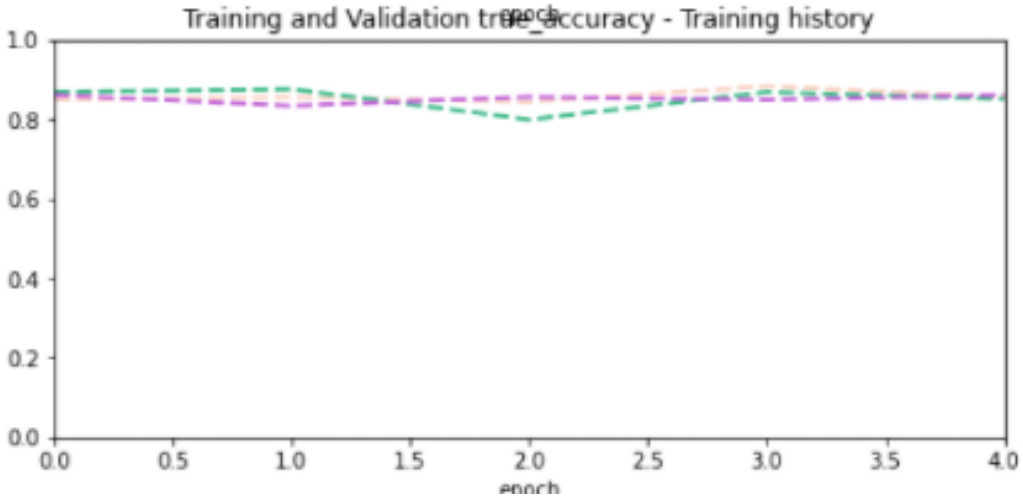
Custom prediction

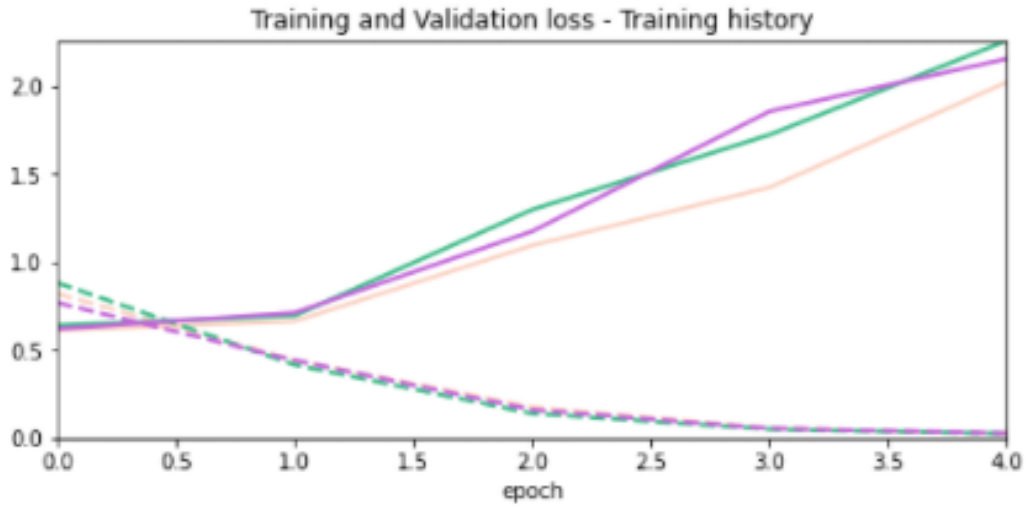
Each text sample is split in n 128 padded sentences – use majority voting on all sentence predictions

```
def predict(self, testDF):
    split_scores = self._model.predict(Dataset(testDF, test=True))

    # reduce targets and scores per sid
    sid_scores = {}
    for i, (sid, text) in testDF[['sid', 'text']].iterrows():
        lst = sid_scores.get(sid, [])
        lst.append(split_scores[i])
        sid_scores[sid] = lst

    scores = []
    targets = []
    last_sid = None
    for sid in testDF['sid']:
        if sid == last_sid:
            continue
        last_sid = sid
        sample_score = np.sum(np.array(sid_scores[sid]), axis=0)
        sample_score = scipy.special.softmax(sample_score)

        targets.append(np.argmax(sample_score, axis=-1))
        scores.append(sample_score)

    return np.array(targets), np.array(scores)
```

Transfer learning results





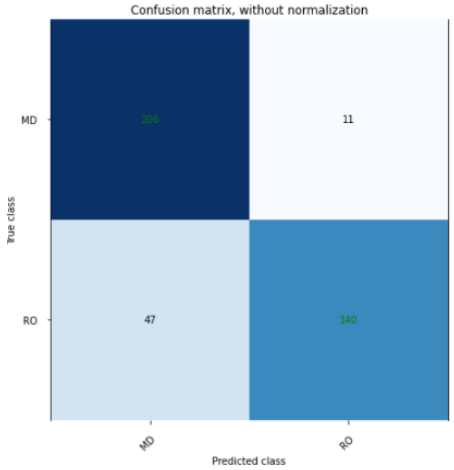


Models overfit very fast

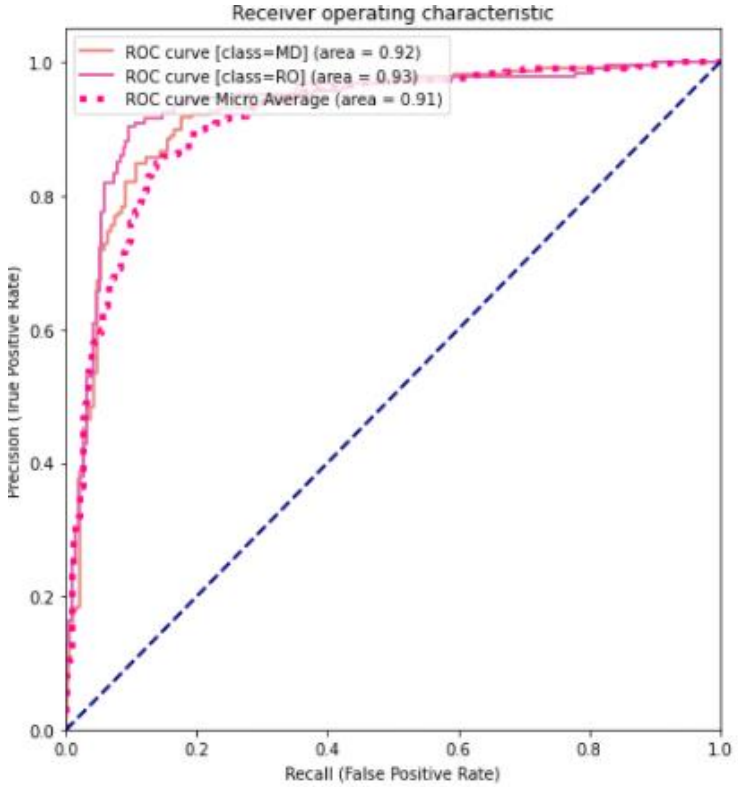
- too much capacity
- **no difference between language model pre-training**

Perhaps each of the selected features from each pre-training have each good discrimative values...

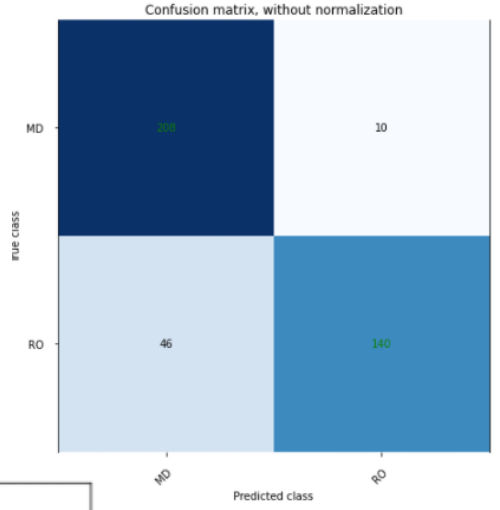
Best model metrics

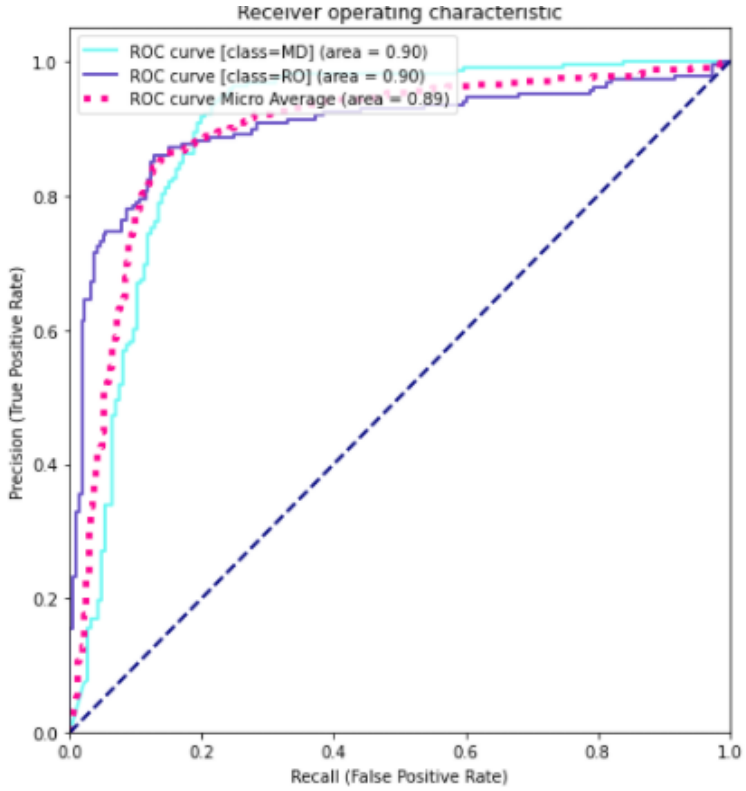


VALIDATION accuracy
0.8564



TEST accuracy
0.8613

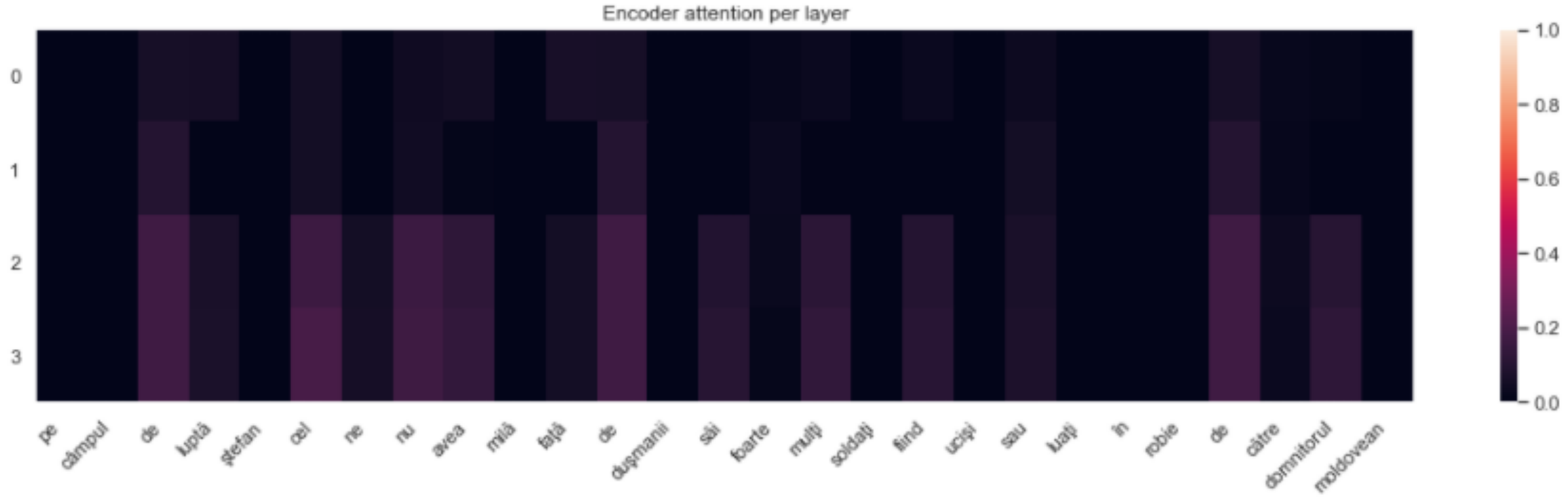




Attention heatmaps - change

Pe câmpul de luptă, Ştefan cel NE nu avea milă faţă de duşmanii săi, foarte mulţi soldaţi fiind ucişi sau luaţi în robie de către domnitorul moldovean

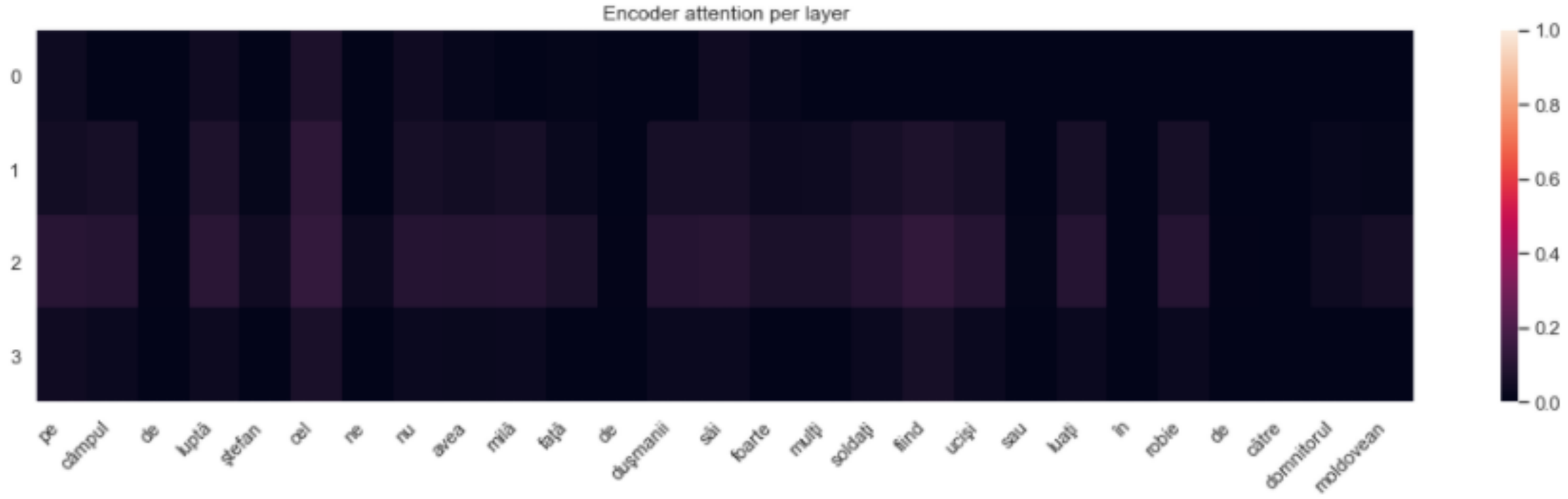
Random masked 15% of sentence words



Attention heatmaps - change

Pe câmpul de luptă, Ştefan cel NE nu avea milă faţă de duşmanii săi, foarte mulţi soldaţi fiind ucişi sau luaţi în robie de către domnitorul moldovean

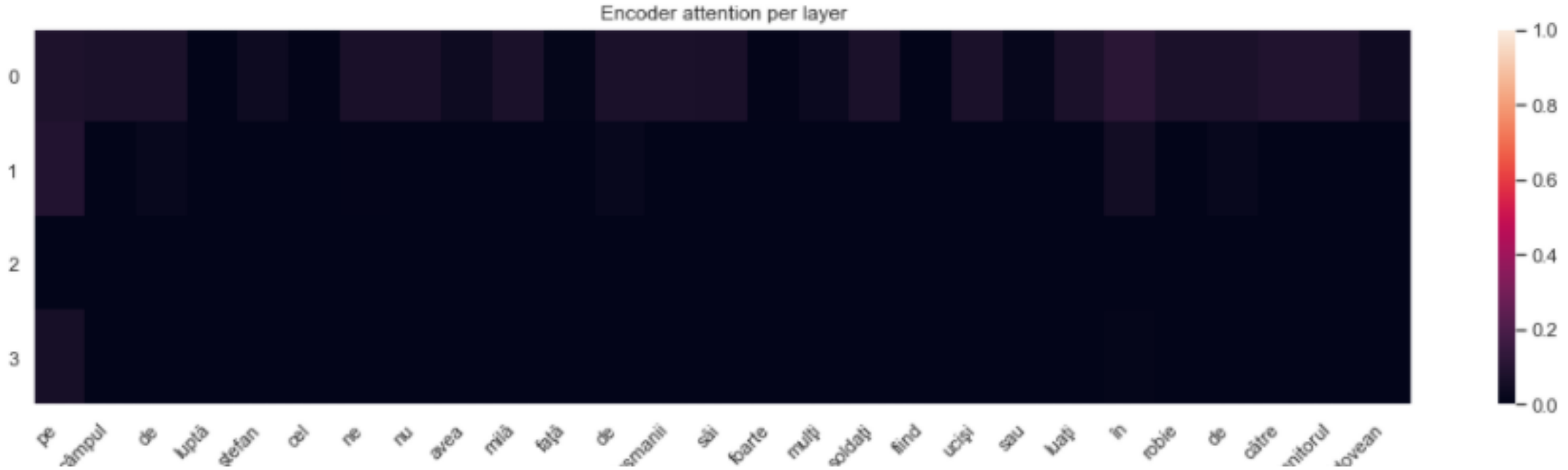
Random masked words with length greater than 5



Attention heatmaps - change

Pe câmpul de luptă, Ştefan cel NE nu avea milă faţă de duşmanii săi, foarte mulţi soldaţi fiind ucişi sau luaţi în robie de către domnitorul moldovean

Random masked words with length less than 4



- Thank you
- TODO – replicate results on a much larger data...