Language Identification of Tweets

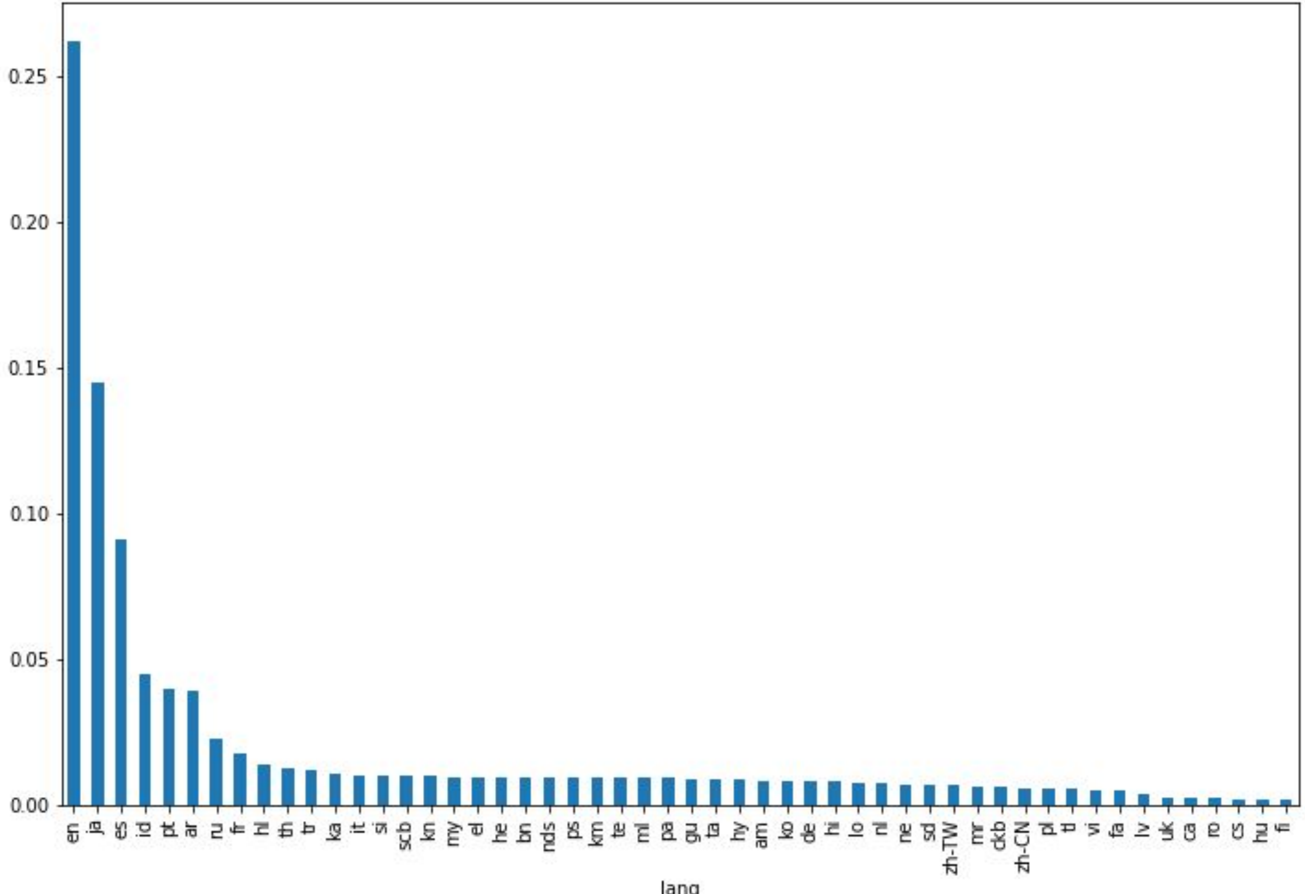
Group 3

Introduction

- Task: Classification of a sequence of characters
- 57 different languages
- 8536 different characters
- 189520 tweet ids for training/validation, 97582 tweet ids or testing
- actually: 80738 tweets for training, 47885 tweets for testing

```
493576585144840193  en  Web OS enables you to quickly & easily manage your web site, social media a
495248136588120064  en  Webbie & Trill Fam InDmixxx @ #ClubBoss Tomorrow Night!! TEXT OR CALL FOR T
490177847059689472  en  Webcomunicati is out! http://t.co/MOSZdPGrl6
489528439284580353  en  Weeds smoking hypocrites smh lol
490021432270024705  en  Weekend hugs to @H50fan @jlopie1 @Tigger0714 @jessSPNH50 @polzlinger @Terry
494814744390668288  en  Weekly Survey:The world’s standard for getting married is “falling in love”
487397434126237696  en  Weekly cut 👌✂
485430801589235712  en  Weeping Japanese politician goes viral | Dunya News http://t.co/NxDqMysAnN
484199418883616768  en  Welcome #YGBOSS (: @Hanbin_Bi96
486695024654942208  en  Welcome :) SQMUSLIM @STALOVEZ @QHye18 @bomixapink @kyungroro_
```

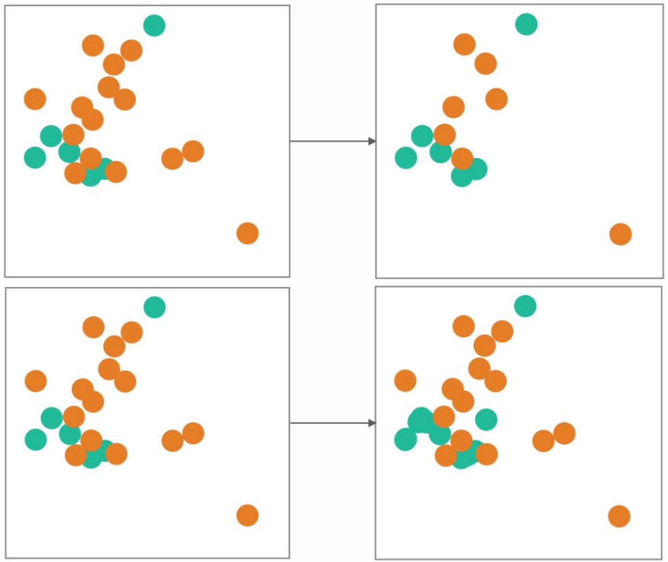
Dataset: Class distribution



Methods to combat class imbalance

- Sampling methods:

  - undersampling of majority class

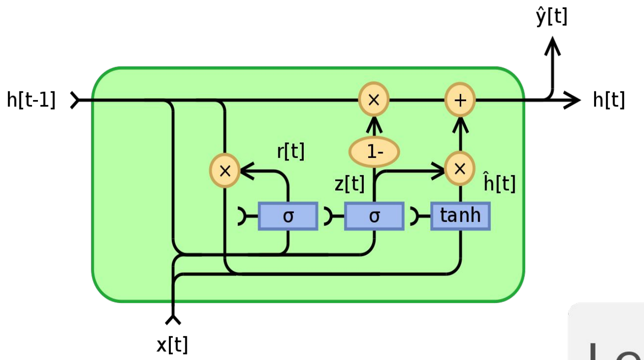


  - oversampling of minority class

- weighted loss functions (larger penalties for misclassifications of the minority class)

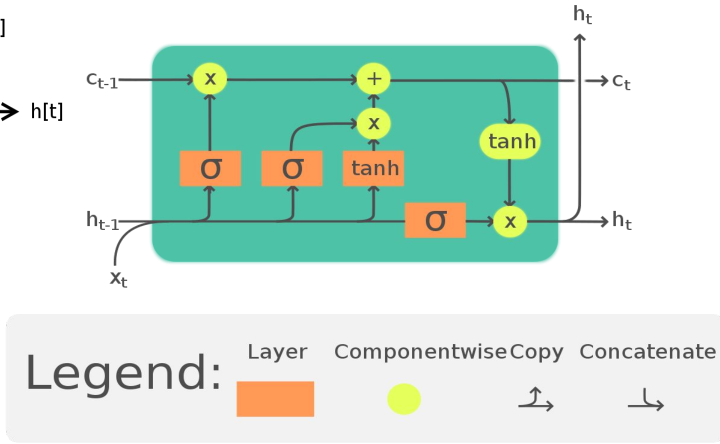
Model Choices

GRU:



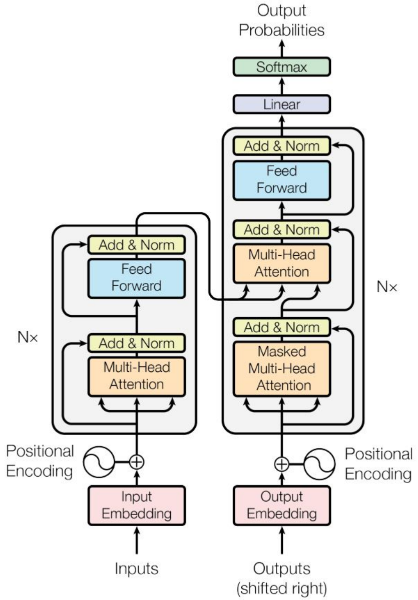
[1]

LSTM:



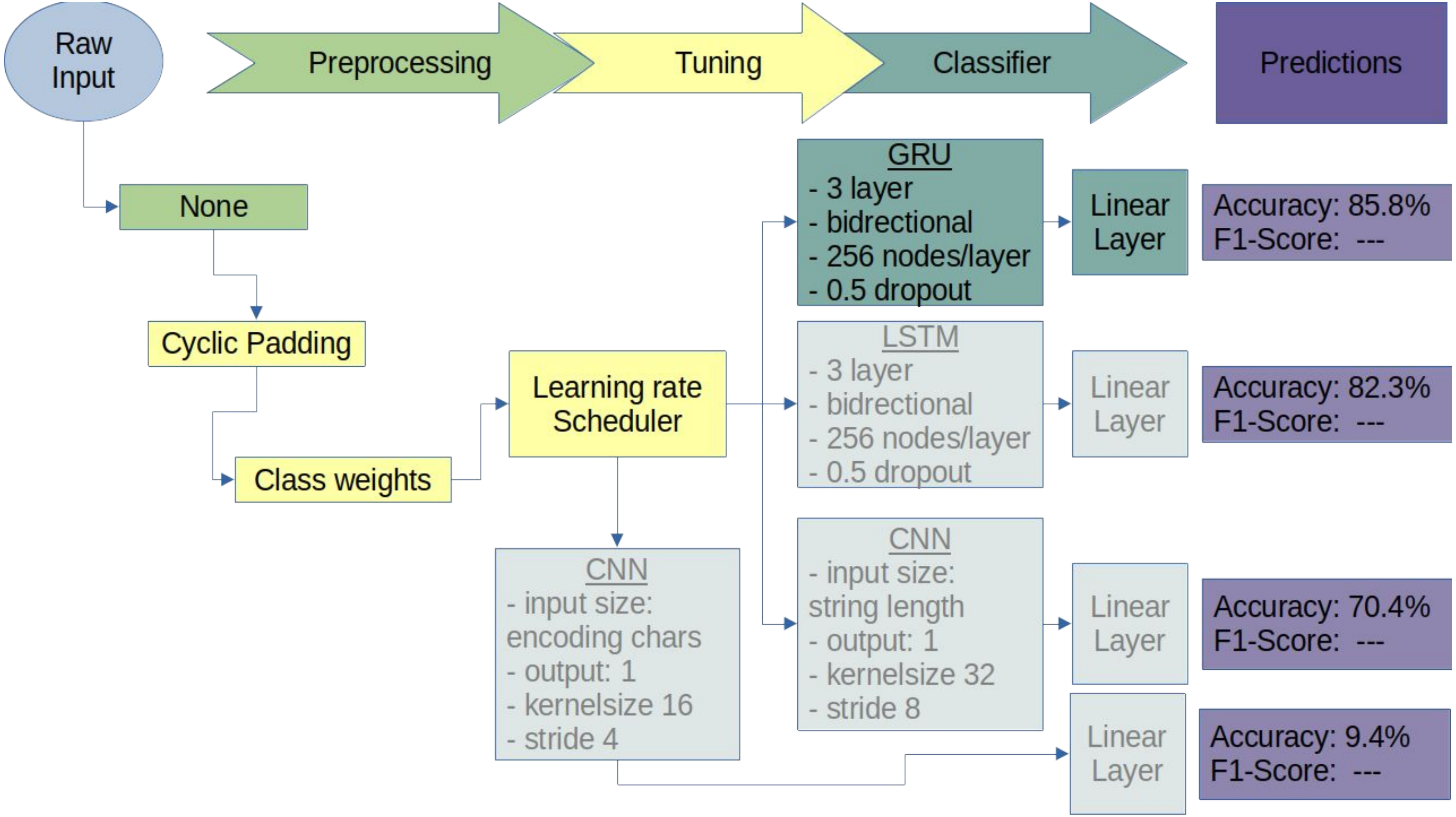
[2]

Transformer/Attention:



[3]

Base Model


Raw Input
Preprocessing
Tuning
Classifier
Predictions
None
Cyclic Padding
Class weights
Learning rate Scheduler
GRU
- 3 layer
- bidirectional
- 256 nodes/layer
- 0.5 dropout
LSTM
- 3 layer
- bidirectional
- 256 nodes/layer
- 0.5 dropout
CNN
- input size: string length
- output: 1
- kernelsize 32
- stride 8
CNN
- input size: encoding chars
- output: 1
- kernelsize 16
- stride 4
Linear Layer
Linear Layer
Linear Layer
Linear Layer
Accuracy: 85.8%
F1-Score: ---
Accuracy: 82.3%
F1-Score: ---
Accuracy: 70.4%
F1-Score: ---
Accuracy: 9.4%
F1-Score: ---

Preprocessing

Eliminate languages when

- not in both train / test set
- <= 100 samples

Impact of Preprocessing

description	example	accuracy	f1 score
baseline		50.8%	46.8%
remove user refs	@BBCWorld	50.5%	45.9%
remove emojis	😃	49.7%	46.0%
lowercase letters	Hello Twitter! -> hello twitter!	48.8%	44.3%

- BiGRU
  - 2 layers,
  - 128 hidden size
  - 0.5 dropout
- Linear layer for classification
- 16 epochs
- Batch size: 64
- Learning rate: 1e-4
- Fixed seed

Impact of Preprocessing

description	example	accuracy	f1 score
baseline		50.8%	46.8%
remove urls	http://t.co/JPmVA03coJ	52.2%	48.0%
remove hashtags	#news	51.1%	46.8%
reduce repetitive characters and trailing whitespaces	GOALLLLLL!!!!!!! -> GOALLL!!!	51.3%	47.4%
reduce characters specific to single language	ስላም ነህ ወዳጄ -> ኧኧኧ ኧኧ ኧኧኧ	52.5%	48.0%
manually reduce chinese, korean, japanese letters	我要飞去米兰T_T -> 这这这这这这T_T	53.1%	48.1%
merge similar languages		61.4%	58.3%
cyclic padding	hello ->hellohellohello	61.3%	57.9%
all improving preprocessing		**70.7%**	**68.9%**

- BiGRU
  - 2 layers,
  - 128 hidden size
  - 0.5 dropout
- Linear layer for classification
- 16 epochs
- Batch size: 64
- Learning rate: 1e-4
- Fixed seed

serbian - croatian - bosnian,
norwegian - danish - swedish
latinized hindu - urdu
“Declaration on the Common Language”

8500 -> 1500 characters

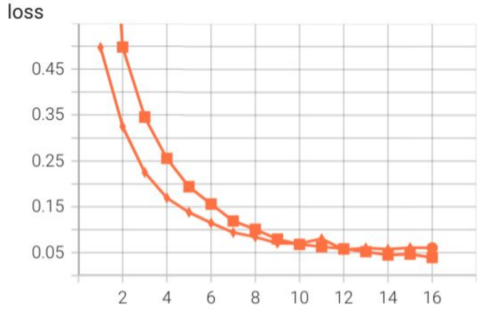
preprocessing reduces training time: 48min -> 8min

Impact of training techniques

description	accuracy	f1 score
all improving preprocessing	70.7%	68.9%
class weights	77.1%	76.5%
cyclic learning rate	71.9%	70.5%
batch size 32	74.6%	74.0%
batch size 16	79.7%	79.9%
learning rate 1e-3	87.5%	88.8%
learning rate 1e-2	58.9%	58.2%
0.6 dropout	70.5%	68.7%
0.4 dropout	71.9%	70.5%
all improving preprocessing + training	89.2%	89.3%
all improving preprocessing + training + reduce lr on plateau	**89.3%**	**89.4%**

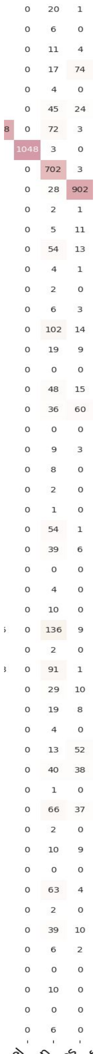
$$\frac{1}{\text{n\_classes}} \cdot \frac{1}{\frac{\text{num\_in\_class}}{\text{num\_gesamt}}}$$

- BiGRU
  - 2 layers,
  - 128 hidden size
  - 0.5 dropout
- Linear layer for classification
- 16 epochs
- Batch size: 64
- Learning rate: 1e-4
- Fixed seed



Impact of training techniques

description	accuracy	f1 score
all improving preprocessing + training + reduce lr on plateau	89.3%	89.4%
trained until convergence	88.9%	89.2%
¼ * english_class_weight	89.4%	**89.6%**
1/10 * english_class_weight	88.8%	89.0%

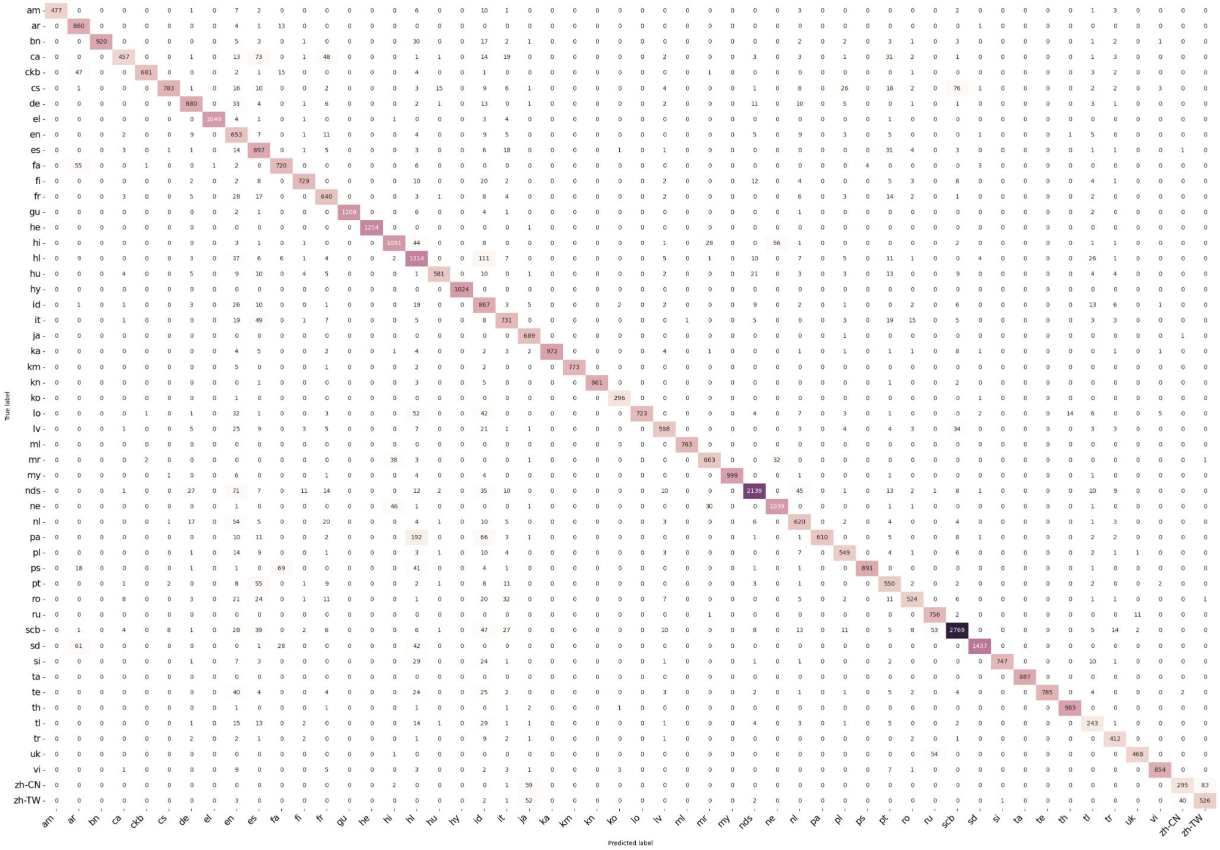


- BiGRU
  - 2 layers,
  - 128 hidden size
  - 0.5 dropout
- Linear layer for classification
- 16 epochs
- Batch size: 64
- Learning rate: 1e-4
- Fixed seed

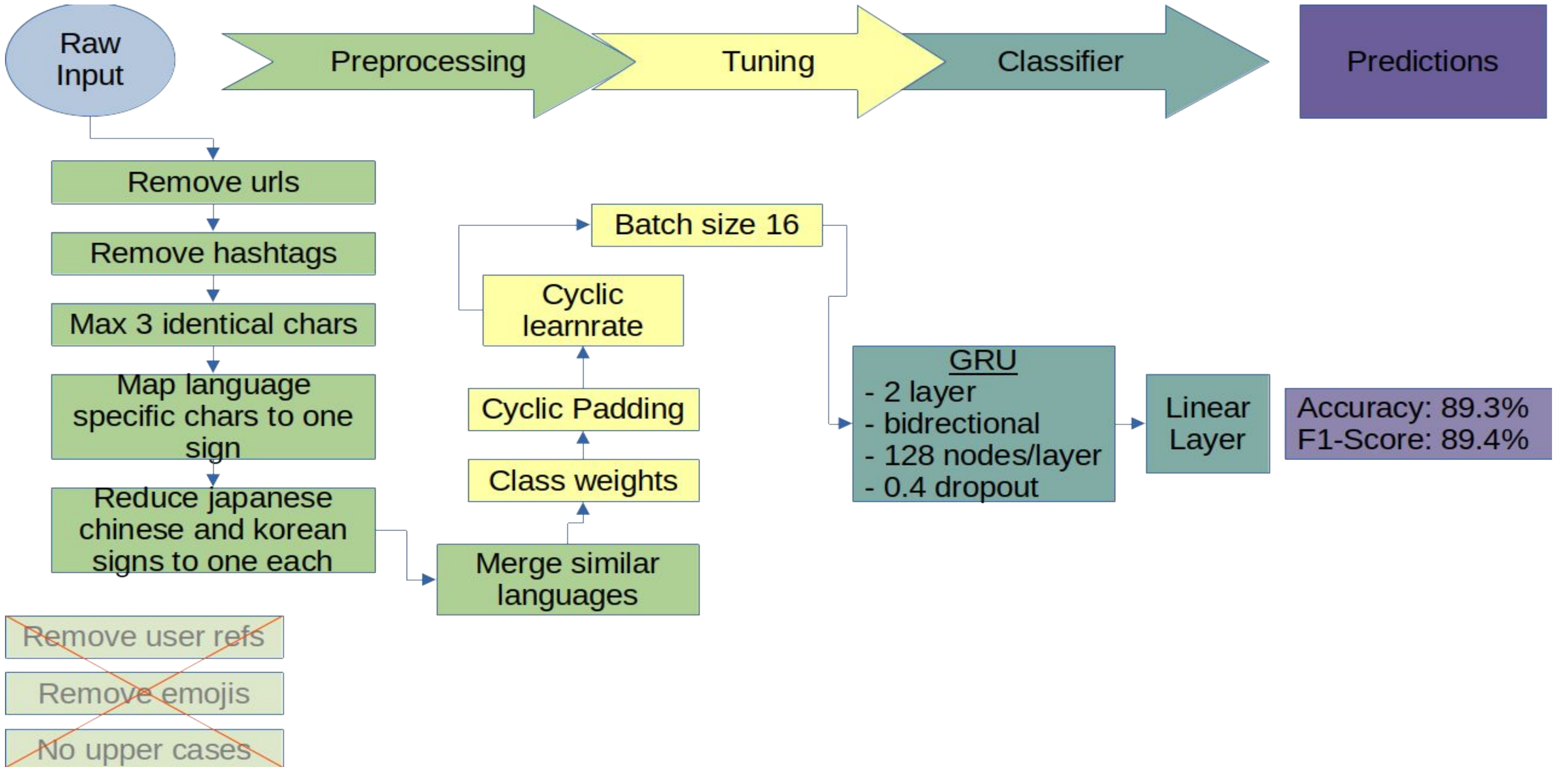
Final BiGRU-only model

- all improving preprocessing
- 3 layers
- 256 hidden states
- dropout 0.4
- cyclic learning rate with initial rate 1e-3 and reduce_lr_on_plateau
- batch size 16
- class weights (with ¼ of initial english weight)

Confusion matrix of Final BiGRU-only model



Model with preprocessing



Improving the Model

1. Freeze the weights of the GRU
2. Replace the linear layer (for SVM) or add a layer between GRU and linear layer (CNN)

- Use a (gaussian kernalized) SVM as final layer instead if a linear Layer

  → Failed to converge to reasonable weights; accuracy ~ 2%

- CNN over output-layer of GRU

  → Failed, also accuracy <10%

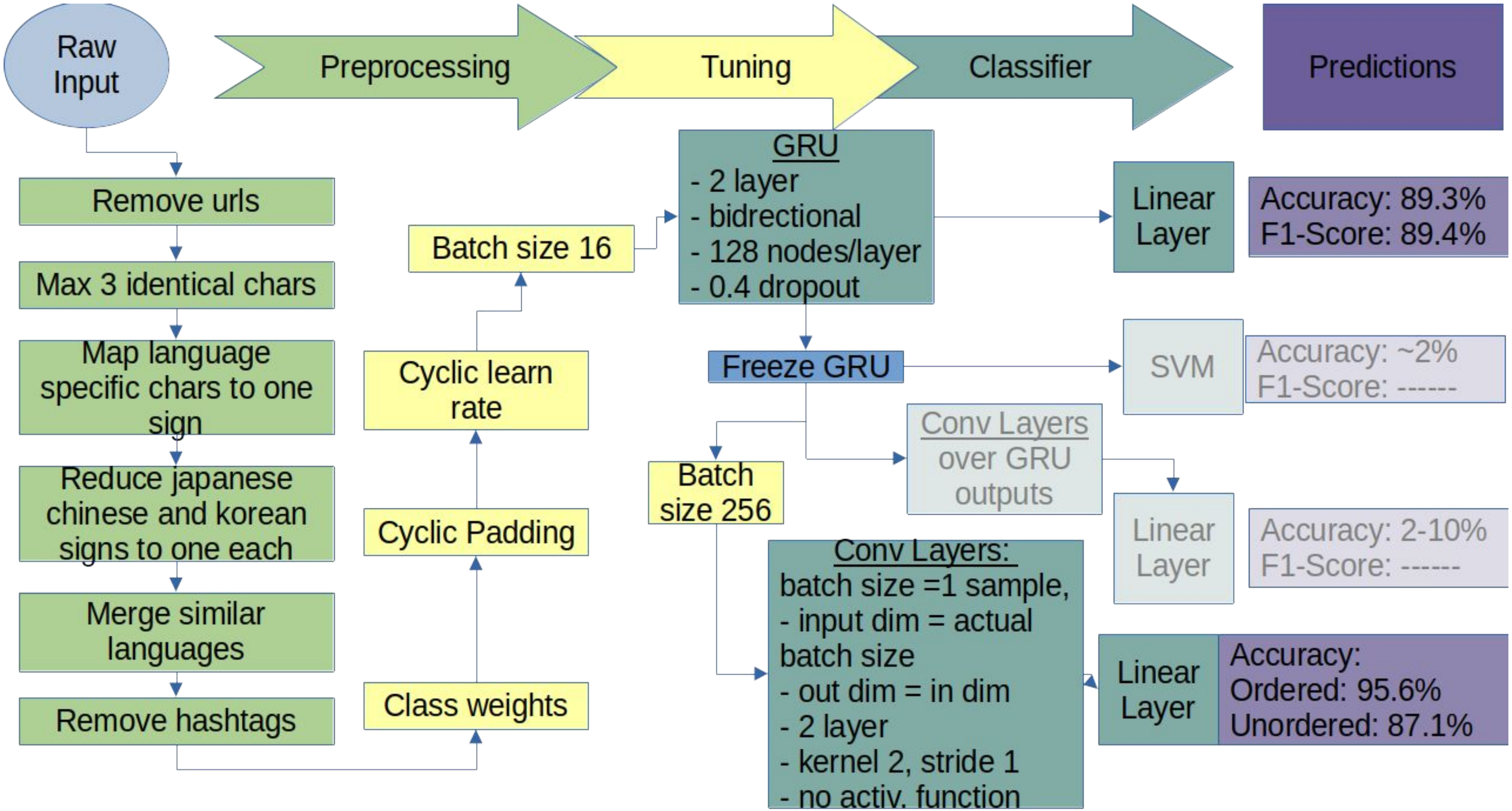
- CNN over samples

  → Success!

  Requires larger batch size for optimal performance,

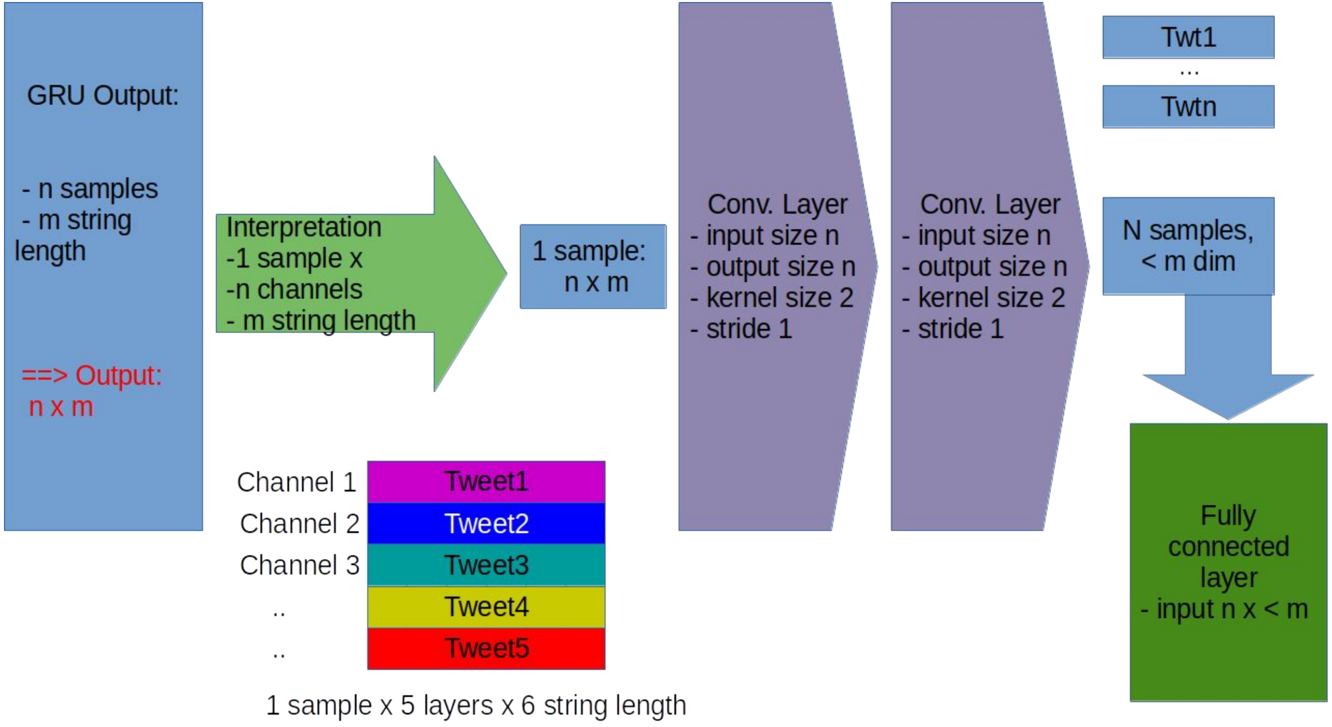
  Accuracy >95%, up to 97.7%

  – Problem: Only achieves such good accuracies for ordered datasets! (else approx. as good as a good GRU)


Raw Input
Preprocessing
Tuning
Classifier
Predictions
Remove urls
Max 3 identical chars
Map language specific chars to one sign
Reduce japanese chinese and korean signs to one each
Merge similar languages
Remove hashtags
Class weights
Cyclic Padding
Cyclic learn rate
Batch size 16
GRU
- 2 layer
- bidirectional
- 128 nodes/layer
- 0.4 dropout
Linear Layer
Accuracy: 89.3%
F1-Score: 89.4%
Freeze GRU
SVM
Accuracy: ~2%
F1-Score: ------
Batch size 256
Conv Layers over GRU outputs
Linear Layer
Accuracy: 2-10%
F1-Score: ------
Conv Layers:
batch size =1 sample,
- input dim = actual batch size
- out dim = in dim
- 2 layer
- kernel 2, stride 1
- no activ. function
Linear Layer
Accuracy:
Ordered: 95.6%
Unordered: 87.1%

What exactly does the Convolution do?

- Using correlations between tweets! →order dependent performance



Further improving the model?

- Convolutional layer, similar to the CNN base model

  – Convolute the sign encodings down

  – Map full tweet sequence to shorter sequence (input layers > output layers)

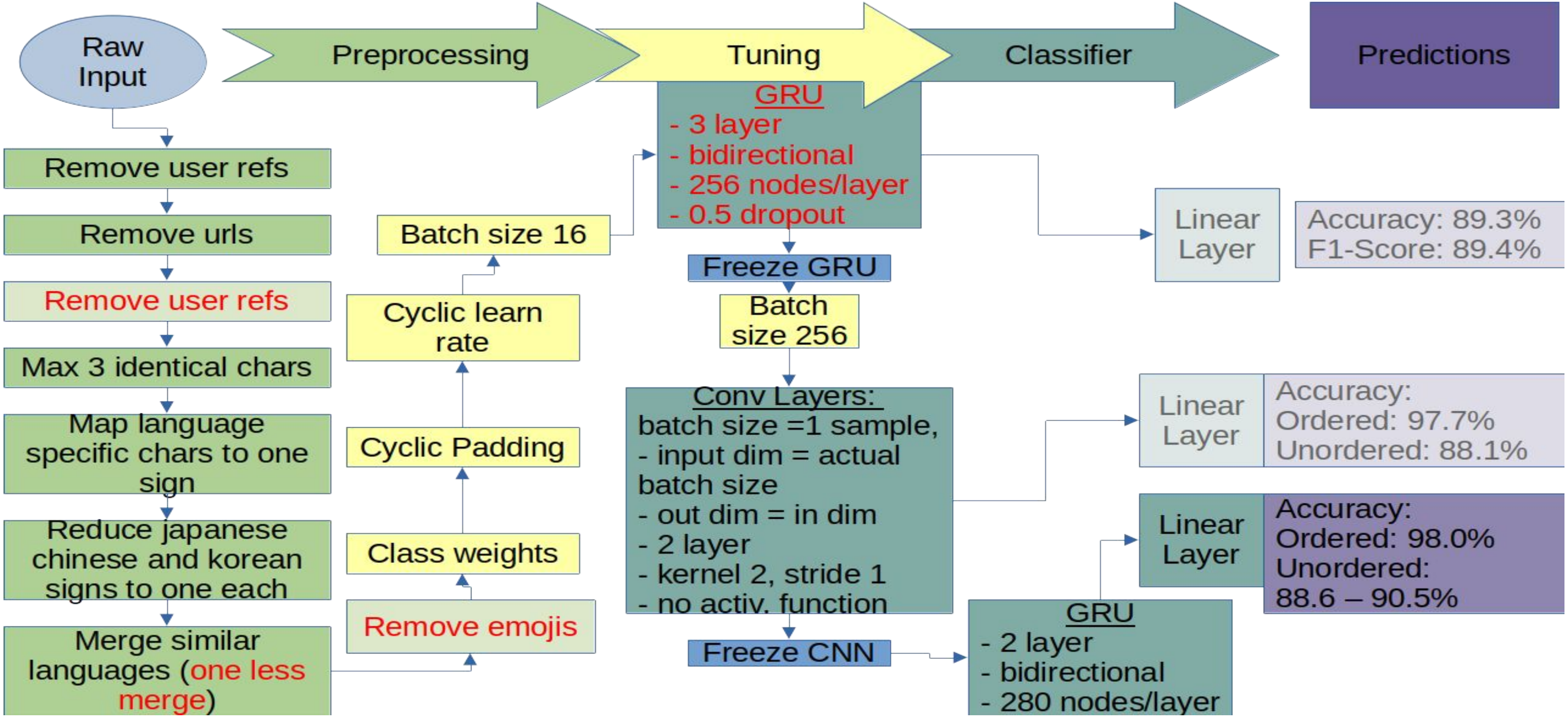
  → Not useful! – slightly reduces the performance

- GRU-layer after Convolutional layer,

  → gives a little boost to performance (but the problem with order remains)

  (only for an older Version, yet)

Best model (so far)



Solutions for not ordered datasets?

- Classify the data (with a different) classifier before and sort the sample according to their given predictions

  – longer evaluation times

  \+ does not need any new data

  – other classifiers might have problems with the same samples → no gain

  → other (not gru based) classifier may be necessary

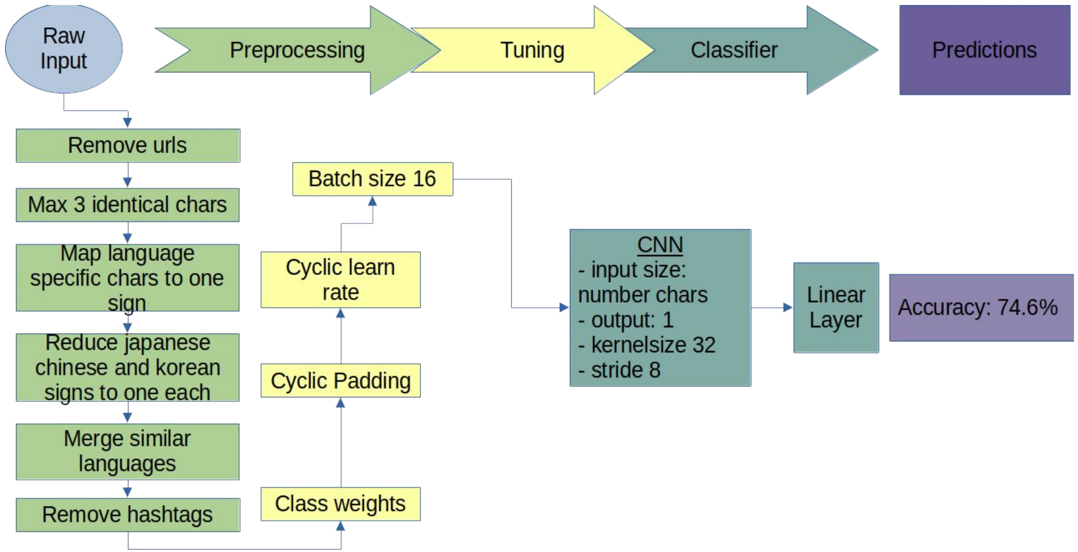
  \+ CNN or Transformer Based Models

Sorting classifiers

1. A simple CNN classifier

   → 90.5% accuracy in GRU-CNN (if CNN used for sorting)

2. Transformer based models (BERT, …)

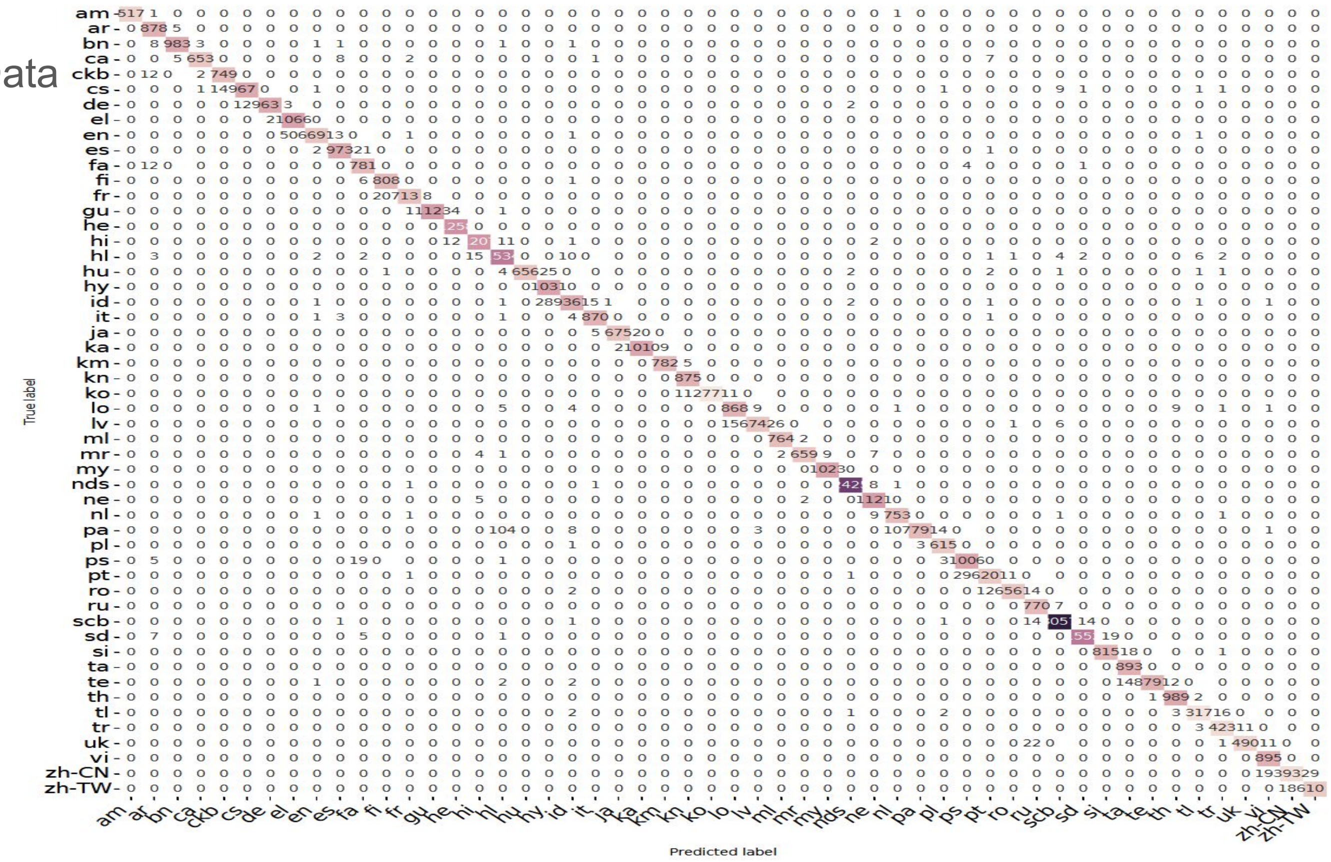


Results of our Model

1. Sorted:
   - Accuracy: 97.8%
   - Precision: 97.7%
   - Recall: 97.3%
   - F1-score 97.5%
2. Not perfectly sorted – Ordered by the additional ‘preprocessing’ CNN:
   - Accuracy: 90.2%
   - Precision: 90.4%
   - Recall: 90.0%
   - F1-score 90.0%

Results of our model: Confusion Matrix

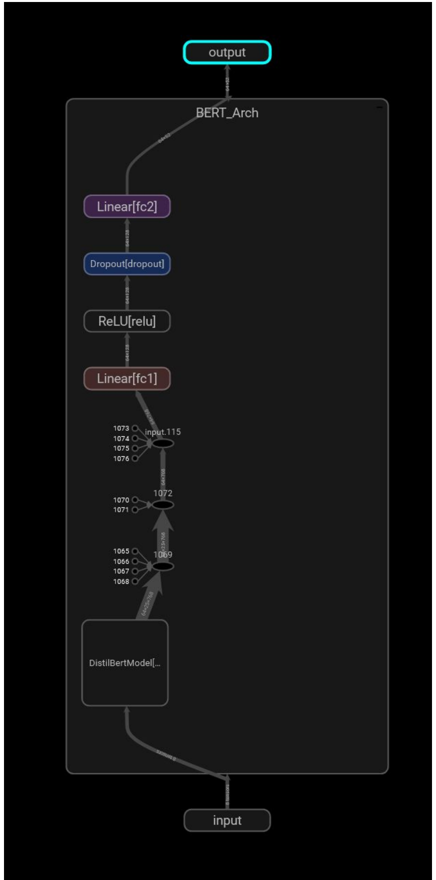
- For sorted Data



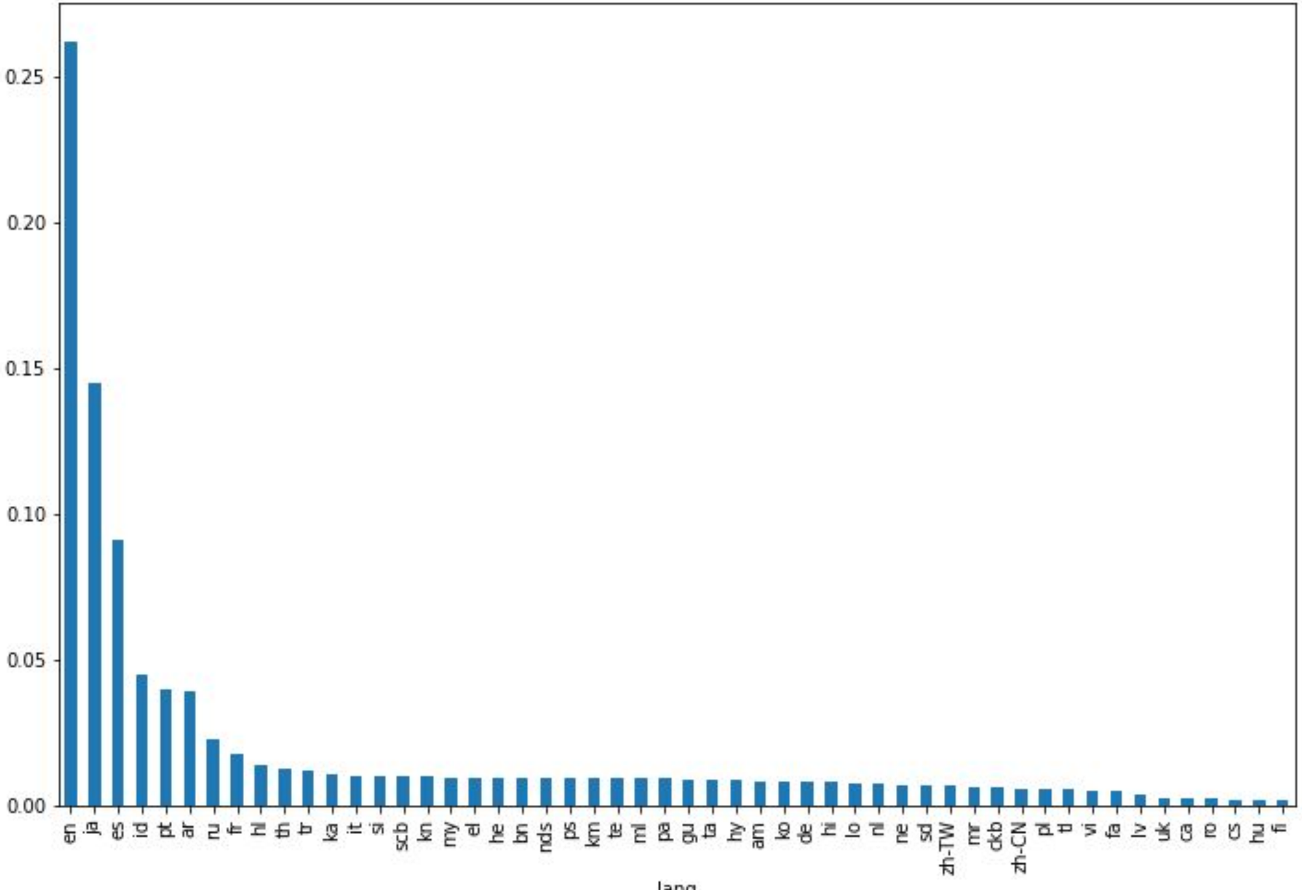
Transfer Learning

- Extraction of features
- Features are fed into a simple Feed-forward classifier
- necessary since tweets are short and use informal language
- Freeze previous layers to reduce amount of trainable parameters

Visualization of the new model with tensorboard

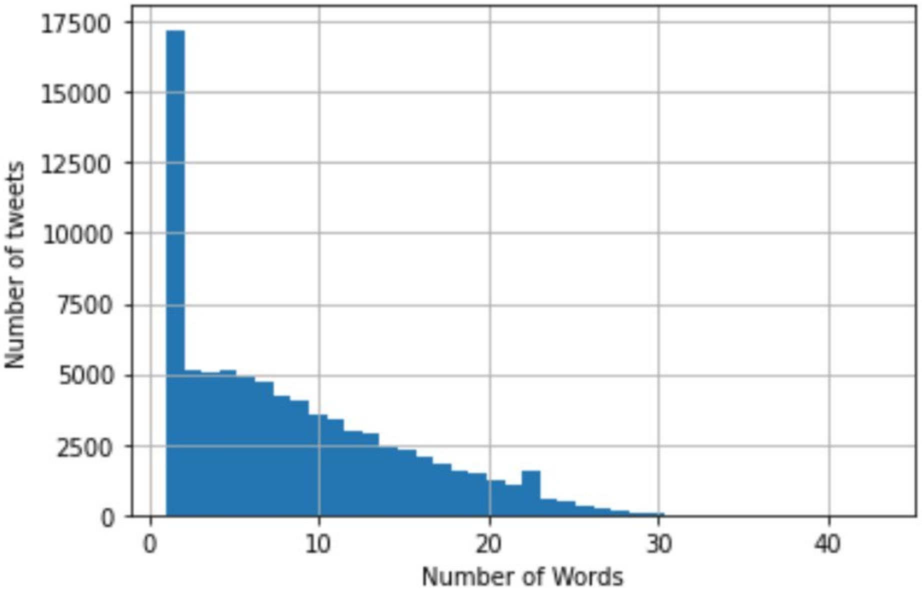


Recap Dataset



Transfer Learning

- Tokenization

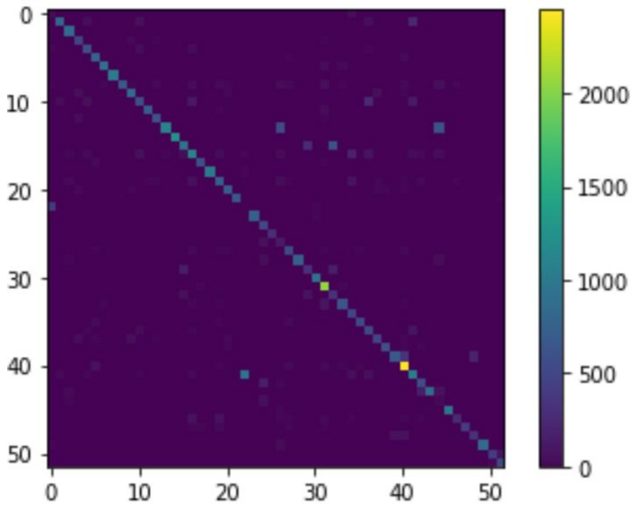


Comparison of models on the test set

roberta on our Dataset (bert was slightly worse)

Acc: 76.9%

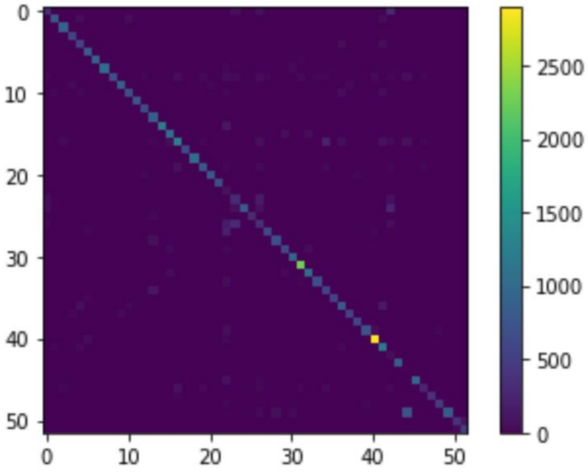
F1: 75.4%



Distilbert on our Dataset

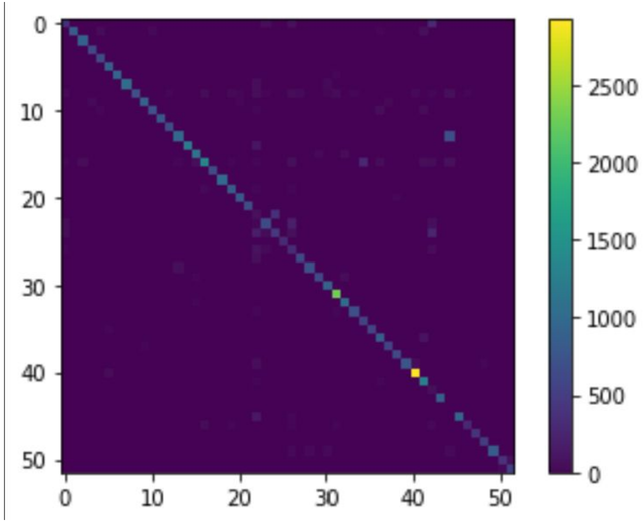
Acc: **83.6%**

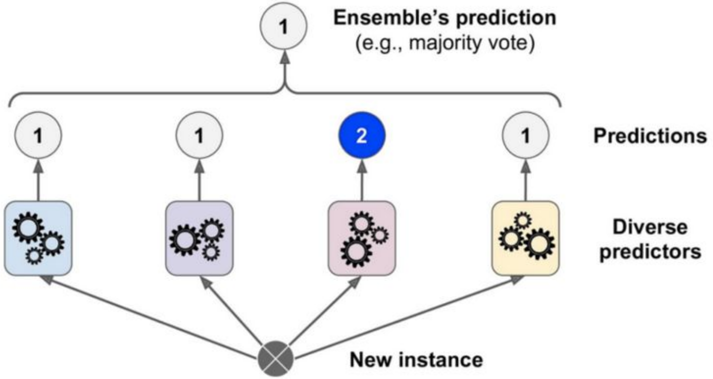
F1: **82.1%**



Ensemble method of different transfer learned classifiers

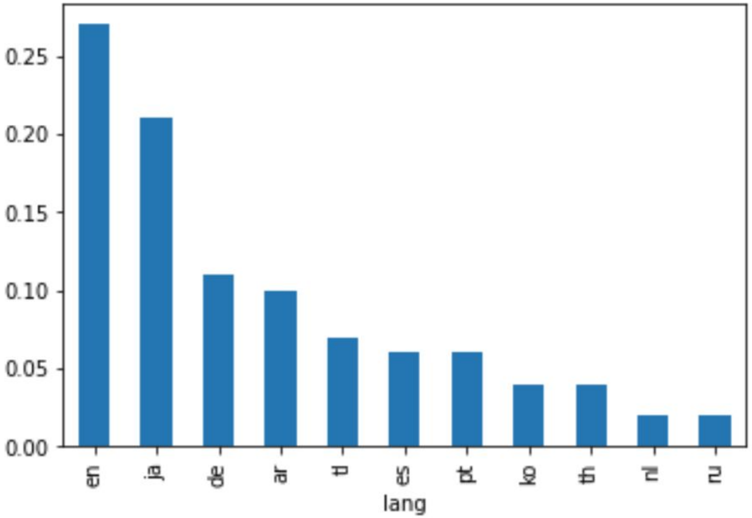
- Accuracy : 86.1
- F1: 84.9





Our own Dataset

- Extraction using the twitter API (acc)
- Labelling process (twitter & google)
- Distribution of classes on the dataset similar to trainings set

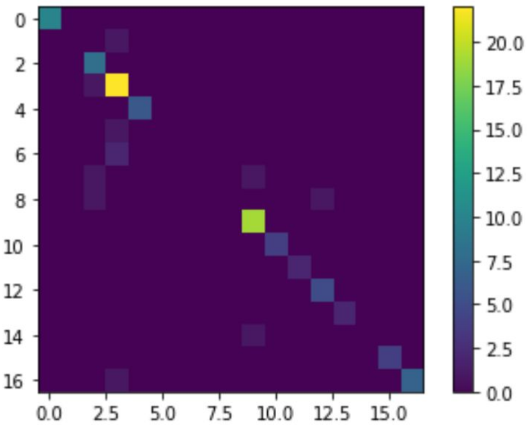


Comparison of models on our own dataset

roberta on our Dataset

Acc: **0.88**%

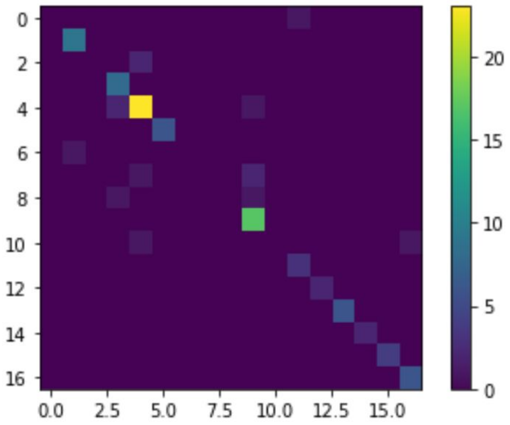
F1: **0.92**%



Ensemble on our Dataset

Acc: 86%

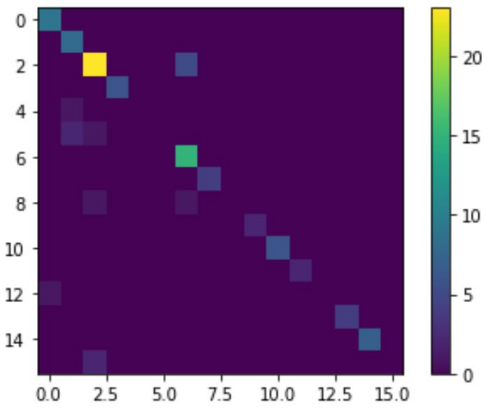
F1: 90.8%



Distilbert on our Dataset

Acc: 82%

F1: 81%



Future work

- deeper training of the LLM
- a nn instead of simple majority vote
- combination of our gru in the ensemble
- training with bigger dataset
- Use as preprocessing for GRU-CNN-Model and possibly adding it to the ensemble

Summary

- Trained our own model for tweet language classification
- Developed a good preprocessing pipeline
- Used a LLM for feature extraction
- Built an Ensemble to increase performance
- Built another dataset using the twitter API

Sources

1. https://upload.wikimedia.org/wikipedia/commons/thumb/3/37/Gated_Recurrent_Unit%2C_base_type.svg/1280px-Gated_Recurrent_Unit%2C_base_type.svg.png
2. https://upload.wikimedia.org/wikipedia/commons/thumb/9/93/LSTM_Cell.svg/1280px-LSTM_Cell.svg.png
3. 1706.03762.pdf (arxiv.org)
4. https://arxiv.org/abs/1608.06048 (nice survey)
5. https://vitalflux.com/5-common-ensemble-methods-in-machine-learning/