OffensEval 2020: Preliminary Results

Christian, Johannes, Kathryn

Preprocessing

Size of training datasets for task A:

English: 9,000,000	Tweets	1.21 GB
Turkish: 31,756	Tweets	3,94 MB
Greek: 8,743	Tweets	1.63 MB
Arabic: 6,999	Tweets	1,26 MB
Danish: 2,961	Tweets	340 KB

Preprocessing

- Create modified subset(s) of a given dataset
- Can specify via commandline:
  - size of the resulting subset
  - number of subsets to create
  - confidence threshold
  - ratio of offensive to not offensive tweets
  - what emoji handler to use
  - what tokenizer to use (nltk, twokenize, wordsegment)
  - additional features to include

```
id	text	confidence
1159533703758061570 @USER His ass need to stay up 😂😂	1
```

```
text	confidence
@USER His ass need to stay up :face_with_tears_of_joy: :face_with_tears_of_joy: 1
```

Next Step

- Automate creation and testing of additional features
- Adapt current code to also be able to handle Subtask B
    - Add proper, helpful documentation
    - Maybe refactor and clean up in the process

xlm-R

- split non-English training data into unbalanced train/balanced dev
  - 95/5 split
  - english dev data is from last year's data; human-annotated
- fine-tuned language model on training/dev data
- trained on concatenated 50k-200k-500k machine-annotated english tweets + non-english human-annotated train tweets
  - also trained on each language individually
  - best results with 200k machine-annotated english tweets
  - adjusted number of warm-up steps

xlm-R

Language	our_model + 1 lang	our_model + all_lang
English	-	**0.79**
Danish	0.705	**0.755**
Arabic	0.844	**0.845**
Turkish	**0.785**	0.763
Greek	0.783	**0.798**

Table 1: F1 scores - xlm-r model fine-tuned on tweets

'our model' meaning xlm-r language model fine-tuned on masked language modeling using the train/dev data from shared task organizers

xlm-R

- Observations:
  - Danish benefits most from multi-lingual model; not a lot of data
  - Turkish did not benefit at all; a lot of its own data
  - English performance on machine-annotated tweets much higher than human-annotated tweets
    - 0.932 vs 0.792
  - performance increase after adding 12k human-annotated english tweets to train data
  - better performance w/ fine-tuning xlm-r language model on the train + dev tweets
  - slight increase observed when adding class weights to loss function
- Next steps:
  - re-run same experiments after handling emojis
  - ensemble all models using a "majority vote" for final test set predictions
    - could use precision/recall/f1 to give certain models stronger votes on certain languages

Machine Learning models

- Twokenized tweets + Tweet length + Avg. word length
- Cross-validation on smaller subset to determine viable models:

~50k English tweets

Linear SVC	~0.91 F1
Linear SVM with SGD	~0.81 F1
Multinomial NB	~0.80 F1
Perceptron	~0.73 F1
RandomForest	~0.70 F1
RBF SVM	~0.65 F1

Machine Learning models

- Twokenized tweets + Tweet length + Avg. word length
- English:

Model	#Tweets	F1 score on dev
Linear SVC	167k	0.80
Linear SVM with SGD (adaptive lr)	167k	0.76
Linear SVM with SGD (constant lr)	167k	0.79
Linear SVC	500k	0.82
Linear SVM with SGD (constant lr)	500k	0.81

Machine Learning models

- Arabic:

Model	Data set combination	F1 score
Linear SVC	Official Train/Dev	~0.87 F1
Linear SVM with SGD	Official Train/Dev	~0.90 F1
Multinomial NB	Official Train/Dev	~0.90 F1
Linear SVC	Balanced Dev	~0.83 F1
Linear SVM with SGD	Balanced Dev	~0.81 F1
Multinomial NB	Balanced Dev	~0.72 F1