Shared Task – Report

Jessica & Rebecka

Preprocessing

Emojis:

- We tested the effect of including and excluding emojis.

<LF>:

- This string appeared in several tweets and appeared to serve no linguistic function, so we figured it was some encoding issue and replaced it with a space to not interfere with word boundaries.

Repeated characters:

- Repeated digits and punctuations were allowed, as the former makes semantic distinctions and the latter could be useful linguistic information for classification. Any other repeated characters, both in English and in Arabic, were limited to appearing twice, since double letters appear in correct spelling in both languages.

Encoding normalisation:

- Some strange encoding issues appeared in the Arabic text, primarily with space characters in the bidirectional text of code-mixed tweets. These were replaced with normal space characters. We tried using the `unicodedata` library to automate normalisation, but found that this also struggled with code-mixed text and ended up scrambling it at bidirectional boundaries.

Removing diacritics:

- Diacritics in Arabic serve no semantic or syntactic function, and only aid pronunciation. Speakers use them inconsistently, which might harm classification performance later. These were automatically stripped with the aid of `pyarabic.araby`, but since this library does not strip the more-commonly used *hamza* and *madda* diacritics, these were additionally stripped manually.

As a final step, the 'OFF' and 'NOT' true labels were converted to the binary 0 and 1, respectively.

Choice of method: Transformer

Transformer-based models, in particular BERT, are currently considered to be the state-of-the-art for various linguistic tasks and have shown great success in sentiment analysis.

Hyperparameter settings

We set the following hyperparameters:

- Set seed - 42
  The classic Hitchhiker's Guide reference.

- Optimizer – AdamW
  AdamW is an improved version of the adaptive optimiser Adam. Literature suggests that AdamW yields better training loss and that models using AdamW generalise better than those trained with Adam (see: https://towardsdatascience.com/why-adamw-matters-736223f31b5d).
  - *Learning rate*: 5e-5 yielded the best result: AUC 73%; accuracy was lower with 1e-10.
  - *Epsilon value*: 1e-8 is a commonly used, small value.

- Epochs - 4
  4 epochs yielded improved accuracy without taking too long to run.

- Loss function – Cross-entropy
  We used cross-entropy loss when adjusting model weights during training. With the cross-entropy loss function, the model learns how to map the input to the realistic probability output. In our case we have a classification problem with binary class labels, so the model needs to calculate the likelihood that a given example belongs to each class. The difference between the two probability distributions can be calculated using cross-entropy.

- BERT model – mini language
  We used the Huggingface pretrained BERT mini language model for Arabic (developed by asafaya) called `asafaya/bert-mini-arabic`. Arabic BERT is a set of BERT language models that consists of four models of different sizes trained using masked language modelling with whole-word masking. Models with large, base, medium, and mini sizes were trained with the same data for 4M steps. We decided to use the mini size for our relatively small training data.

- Batch size - 16
  For fine-tuning BERT, researchers tend to recommend a batch size of 16 or 32. We decided to use 16 for our small model and learning rate.

Results on test data

The inclusion of emojis yielded ~6-7% better accuracy than exclusion. It appears that BERT can handle emojis just like any other token and that they serve an important linguistic function, in line with much of previous research.

Interestingly, removing usernames yielded ~1% better accuracy on the tweets with emojis and ~6% worse accuracy on those without. We have not yet been able to find out exactly why this happens.

Overall, our top accuracy of 78.98% is not particularly convincing given that the majority class represents 80% of the data and that this figure could thus be taken as a baseline. This underwhelming performance could be chalked up to the class imbalance (which transformers typically do not handle well) and/or the relatively small dataset. It is also possible that more targeted preprocessing could have improved performance, as the encoding issues in particular were challenging for us given the unique problems presented by handling bidirectional text.

Encountered problems

Language-specific issues:

- This was our first time working with bidirectional text. The first and most obvious challenge we faced was that of how differently it can be displayed in different text editors, IDEs, and Jupyter vs. Colab. During development and testing of the preprocessing code it was especially tricky just to copy-paste different strings to see how different steps would affect the output.
- We also wanted to try converting the emojis to descriptive text strings in Arabic such as those offered in English by the `emoji` library, but found no equivalent in Arabic. We were surprised by this lack given that Arabic is one of the most commonly spoken languages in the world, and it highlighted the significant tendency of NLP research to focus on Latin script.
- Arabic is one of the most dialectally diverse languages in the world, to the point that some dialects are less mutually intelligible than various European languages considered to be distinct such as Danish, Norwegian, and Swedish. Since there was no information about which dialects were represented in the data, we can assume that dialect was not considered in the data collection process. Representing only one dialect might have improved classification performance.

Malformed data:

- We could not find any account for the appearance of the '<LF>' string in either the data description or online. If this was some encoding issue during data collection, we would have expected the researchers to remove this before releasing the data.
- We fortunately stumbled upon the issue of the six giant 'tweets' (1000+ characters) when we were conducting our initial data exploration and calculated the average number of characters per word for each tweet. We tried to find the correct boundaries along which to split them into separate tweets, but we abandoned this attempt when we found that different text editors displayed the text in different orders due to their bidirectionality, making it impossible to determine which labels/IDs corresponded to which tweets, and simply removed them. This was a bit of a shame since these rows could have contained as many as 68 tweets, but we figured it was better for model performance to omit than to keep them as they were since they would have represented <1% of the training data if correctly split. This experience reinforced for us the importance of conducting basic data exploration such as calculating token and character counts before moving further with code.

Methods and steps for further result improvement

With more time and familiarity with encoding issues, we would have implemented a more robust version of `normalise_encoding()`. The current code addresses only the main issues we had time to explore with the naked eye. We would also like to find out why removing usernames so significantly affected the performance of classification without emojis as compared to that with emojis, and if the data or our code is to blame for the underwhelming final performance measures.