Multitask Learning for Twitter Sentiment Analysis
Université Grenoble-Alpes, CNRS, Grenoble INP*, LIG
Georgios.Balikas@imag.fr, Simon.Moura@imag.fr, Massih-Reza.Amini@imag.fr

1. MOTIVATION

- In machine learning, usually we focus on a single task. We optimize our model over a loss or a metric for the task at hand, either using a single model or an ensemble, whose hyperparameters we tune.
- As a result, we ignore information that may be useful, for instance training signals from related tasks.
- Hypothesis: We can enable our model to generalize better on the original task, by sharing representations between related tasks.

2. MULTI-TASK LEARNING WITH BiLSTM

The neural network architecture for multitask learning using hard weight sharing between the N tasks. A bidirectional LSTM is used for learning a tweet representation that can be combined with “Additional Features”. These can be domain-specific features or prior knowledge, are extracted in the preprocessing step and enable the network to integrate additional knowledge.

3. THE EXPERIMENTAL FRAMEWORK: TWITTER SENTIMENT CLASSIFICATION

We experiment with two sentiment classification tasks:
- Fine-grained: VeryPositive, …, VeryNegative. The primary task for this work.
- Ternary: Positive, Neutral, Negative

The performance improvements of may be due to:
1. Data Augmentation: Expands the set of training examples
2. Inductive bias: Prefer a hypothesis that performs well across tasks
3. Feature attention focusing: Decrease the weights on noisy features

The scores on macro-averaged mean absolute error. The best (lowest) score is shown in bold and is achieved in the multitask setting with the biLSTM architecture. nbow uses only text (GloVe embeddings), while nbow+ incorporates the “Additional Features”.

F1 scores with nbow+: Multitask learning achieves the best score.

REFERENCES

Open Source Implementation: https://github.com/balikasg/sigir2017

ACKNOWLEDGEMENTS
This work is partially supported by the CIFRE N 28/2015. Part of the SIGIR 2017 conference registration is covered by a SIGIR student grant. The models are implemented with Scikit-Learn and Keras.