# fastText

## Technical Contributions

* Modified a linear classifier to work with a large dataset
* Uses rank constraint on a linear classifier
* Probability distribution calculated using a softmax function
* Improved running time using hierarchical softmax, based on Huffman coding tree - complexity went down from O(kh) to O(h log2 k)
* Using depth-first search increases speed by not exploring small probability branches
* Bag of n-grams used to retain information about local word order
* Uses skip-gram with negative sampling (sub-words are positive examples and random samples from dictionary are negative examples)

## Strengths

* Relatively much faster than conventional neural networks (both training and testing)
* Accuracy on par with neural network-based models, further improved by adding bigram information
* N-grams help capturing the meanings of shorter words, including prefixes and suffixes

## Weaknesses

* Cannot capture different contexts

## Improvements

* FastText.zip: An improved version of fastText designed to fit in limited memory. Employs a method based on product quantization to store embeddings. By further avoiding quantization artefacts, reduces memory requirements by half with only a small loss in accuracy

## Questions

* Are the methods of evaluation used good enough to evaluate deep learning models? - *from the paper’s conclusion*