FastText

Jon Koss, Abhishek Jindal
FastText

FastText is on par with state-of-the-art deep learning classifiers in terms of accuracy

But it is way faster:

- FastText can train on more than one billion words in less than ten minutes using a standard multicore CPU
- Classify nearly 500K sentences among 312K classes in less than a minute
Continuous Bag of Words (CBOW)

- Uses a window in both directions of the target word
- Word order within the window is ignored
- Shared weight matrix between input and projection layer for all word positions
- Projection Layer + Softmax Layer
Continuous Bag of N-grams

- Uses a window in both directions of the target n-gram
- Word order within each n-gram is preserved
- Order of n-grams in window is not preserved
Skipped N-grams

In skipped N-grams, you try to predict the surrounding context words conditioned on the current word.

More formally, given a sequence of training words $w_1, w_2, w_3, \ldots, w_T$ skipped n-gram model tries to maximize the average log probability

$$\frac{1}{T} \sum_{t=1}^{T} \left[ \sum_{j=-k}^{k} \log p(w_{t+j}|w_t) \right]$$
Comparison of CBOW and Skipped N-Grams

Continuous Bag of Words:

Several times faster to train than the skip-gram

Slightly better accuracy for the more frequent words

Complexity: \[ Q = N \times D + D \times \log_2(V) \]

Skipped N-grams:

Works well with small amount of the training data

Represents well even rare words or phrases.

Performance increases with context size, but cost too
Hashing Trick

From ‘Strategies for Training Large Scale Neural Network Models’ by Mikolov et al.

Represent the data in hash table format as shown on Wiki (n-gram of n=1)

Use hashing trick to map sparse matrix to one dimensional array

The size of the hash table and how is it stored - N X V (N = Number of Documents, V = Size of the vocabulary)

Example:

1. John likes to watch movies.
2. Mary likes movies too.
3. John also likes football.
Hierarchical Softmax

First proposed by Morin and Bengio in Hierarchical Probabilistic Neural Network Language Model

Inspired by the binary tree.

Log\(_2(N)\) instead of \(N\)

\[
P(n_{l+1}) = \prod_{i=1}^{l} P(n_i).
\]

\[
\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.
\]
Neural Network Linear Model Model Architecture

Figure: Feedforward neural network based LM used by Y. Bengio and H. Schwenk
Complexity of Neural Network Linear Model (NNLM)

**Complexity:** \( Q = N \times D + N \times D \times H + H \times V \)

- \( N \): The number of previous words used for context
- \( D \): Number of dimensions of the projection matrix
- \( H \): Number of hidden units in the hidden layer
- \( V \): Size of the vocabulary

\( H \times V \) is the dominating term.

Using Hierarchical Softmax, we can get it down to \( H \times \log_2 V \).
Recurrent Neural Net Language Model

**Complexity:** $Q = H \times H + H \times V$

Dominating term = $H \times V$

RNNs don’t have a projection layer, only input, hidden and output layer
FastText Architecture

**Complexity:** $Q = N \times D + D \times \log_2(V) + H \times D$

*Figure 1: Model architecture of fastText for a sentence with $N$ ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.*
For a set of $N$ documents, the model minimizes the negative log likelihood over the classes.

Optimization was performed using stochastic gradient descent and a linearly decaying learning rate.

$$\frac{-1}{N} \sum_{n=1}^{N} y_n \log(f(BAx_n)),$$

$X_n$ is normalized bag of words of the $n^{th}$ document

$Y_n$ the label, A, B weight matrices
Task Description: Sentiment Analysis

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Classes</th>
<th>Train Samples</th>
<th>Test Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG’s News</td>
<td>4</td>
<td>120,000</td>
<td>7,600</td>
</tr>
<tr>
<td>Sogou News</td>
<td>5</td>
<td>450,000</td>
<td>60,000</td>
</tr>
<tr>
<td>DBPedia</td>
<td>14</td>
<td>560,000</td>
<td>70,000</td>
</tr>
<tr>
<td>Yelp Review Polarity</td>
<td>2</td>
<td>560,000</td>
<td>38,000</td>
</tr>
<tr>
<td>Yelp Review Full</td>
<td>5</td>
<td>650,000</td>
<td>50,000</td>
</tr>
<tr>
<td>Yahoo! Answers</td>
<td>10</td>
<td>1,400,000</td>
<td>60,000</td>
</tr>
<tr>
<td>Amazon Review Full</td>
<td>5</td>
<td>3,000,000</td>
<td>650,000</td>
</tr>
<tr>
<td>Amazon Review Polarity</td>
<td>2</td>
<td>3,600,000</td>
<td>400,000</td>
</tr>
</tbody>
</table>
### Sentiment Analysis: Performance

<table>
<thead>
<tr>
<th>Model</th>
<th>AG</th>
<th>Sogou</th>
<th>DBP</th>
<th>Yelp P.</th>
<th>Yelp F.</th>
<th>Yah. A.</th>
<th>Amz. F.</th>
<th>Amz. P.</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoW (Zhang et al., 2015)</td>
<td>88.8</td>
<td>92.9</td>
<td>96.6</td>
<td>92.2</td>
<td>58.0</td>
<td>68.9</td>
<td>54.6</td>
<td>90.4</td>
</tr>
<tr>
<td>ngrams (Zhang et al., 2015)</td>
<td>92.0</td>
<td>97.1</td>
<td>98.6</td>
<td>95.6</td>
<td>56.3</td>
<td>68.5</td>
<td>54.3</td>
<td>92.0</td>
</tr>
<tr>
<td>ngrams TFIDF (Zhang et al., 2015)</td>
<td>92.4</td>
<td>97.2</td>
<td>98.7</td>
<td>95.4</td>
<td>54.8</td>
<td>68.5</td>
<td>52.4</td>
<td>91.5</td>
</tr>
<tr>
<td>char-CNN (Zhang and LeCun, 2015)</td>
<td>87.2</td>
<td>95.1</td>
<td>98.3</td>
<td>94.7</td>
<td>62.0</td>
<td>71.2</td>
<td>59.5</td>
<td>94.5</td>
</tr>
<tr>
<td>char-CRNN (Xiao and Cho, 2016)</td>
<td>91.4</td>
<td>95.2</td>
<td>98.6</td>
<td>94.5</td>
<td>61.8</td>
<td>71.7</td>
<td>59.2</td>
<td>94.1</td>
</tr>
<tr>
<td>VDCNN (Conneau et al., 2016)</td>
<td>91.3</td>
<td>96.8</td>
<td>98.7</td>
<td>95.7</td>
<td>64.7</td>
<td>73.4</td>
<td>63.0</td>
<td>95.7</td>
</tr>
<tr>
<td>fastText, $h = 10$</td>
<td>91.5</td>
<td>93.9</td>
<td>98.1</td>
<td>93.8</td>
<td>60.4</td>
<td>72.0</td>
<td>55.8</td>
<td>91.2</td>
</tr>
<tr>
<td>fastText, $h = 10$, bigram</td>
<td>92.5</td>
<td>96.8</td>
<td>98.6</td>
<td>95.7</td>
<td>63.9</td>
<td>72.3</td>
<td>60.2</td>
<td>94.6</td>
</tr>
</tbody>
</table>

**Table 1:** Test accuracy [%] on sentiment datasets. FastText has been run with the same parameters for all the datasets. It has 10 hidden units and we evaluate it with and without bigrams. For char-CNN, we show the best reported numbers without data augmentation.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>small char-CNN</td>
<td>big char-CNN</td>
<td>depth=9</td>
</tr>
<tr>
<td>AG</td>
<td>1h</td>
<td>3h</td>
<td>24m</td>
</tr>
<tr>
<td>Sogou</td>
<td>-</td>
<td>-</td>
<td>25m</td>
</tr>
<tr>
<td>DBpedia</td>
<td>2h</td>
<td>5h</td>
<td>27m</td>
</tr>
<tr>
<td>Yelp P.</td>
<td>-</td>
<td>-</td>
<td>28m</td>
</tr>
<tr>
<td>Yelp F.</td>
<td>-</td>
<td>-</td>
<td>29m</td>
</tr>
<tr>
<td>Yah. A.</td>
<td>8h</td>
<td>1d</td>
<td>1h</td>
</tr>
<tr>
<td>Amz. F.</td>
<td>2d</td>
<td>5d</td>
<td>2h45</td>
</tr>
<tr>
<td>Amz. P.</td>
<td>2d</td>
<td>5d</td>
<td>2h45</td>
</tr>
</tbody>
</table>

**Table 2**: Training time for a single epoch on sentiment analysis datasets compared to char-CNN and VDCNN.
Task: Tag Prediction

Predicting tags according to the titles and caption for the Yahoo Flickr Creative Commons 100 Million dataset which contains 100 M of images with titles, caption and tags. [http://yfcc100m.appspot.com/](http://yfcc100m.appspot.com/)
Results: 566,094 out of 100,000,000 items

Show Statistics:

Tag Cloud::

tree nature trees sky green christmas landscape winter park snow None autumn water canon leaves forest spring flower blue california nikon fall sunset sun grass garden flowers travel plant clouds angland light red white christmas tree night summer mountain lake usa arbre japan leaf river beach photography wood uk bird city

Showing Page 1 of 18,173

:: First ... 1 - 2 - 3 - 4 ... Last ::
Tag Prediction Task

FastText was evaluated for scalability on the tag prediction of 100 M images with captions, titles and tags.

Remove sparse words and tags occurring < 100 times.

Train Set: 91,188,648 examples (1.5B tokens).

Validation Set: 930,497

Test Set: 543,424

Vocabulary Size: 297,141

Tag Size: 312,116
TagSpace Network - The competing network
Training methodology for Tag Prediction

FastText is run for 5 epochs and compared to TagSpace for:

50 Hidden Units

200 Hidden Units

Similar results between two networks for the small hidden layer

Bigrams (n=2, n-grams) significantly improved performance

Test Phase: Speedup of 600X
Comparison with Tagspace

<table>
<thead>
<tr>
<th>Model</th>
<th>prec@1</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Train Test</td>
</tr>
<tr>
<td>Freq. baseline</td>
<td>2.2</td>
<td>-</td>
</tr>
<tr>
<td>Tagspace, $h = 50$</td>
<td>30.1</td>
<td>3h8</td>
</tr>
<tr>
<td>Tagspace, $h = 200$</td>
<td>35.6</td>
<td>5h32</td>
</tr>
<tr>
<td>fastText, $h = 50$</td>
<td>31.2</td>
<td>6m40</td>
</tr>
<tr>
<td>fastText, $h = 50$, bigram</td>
<td>36.7</td>
<td>7m47</td>
</tr>
<tr>
<td>fastText, $h = 200$</td>
<td>41.1</td>
<td>10m34</td>
</tr>
<tr>
<td>fastText, $h = 200$, bigram</td>
<td>46.1</td>
<td>13m38</td>
</tr>
</tbody>
</table>

Table 5: Prec@1 on the test set for tag prediction on YFCC100M. We also report the training time and test time. Test time is reported for a single thread, while training uses 20 threads for both models.