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

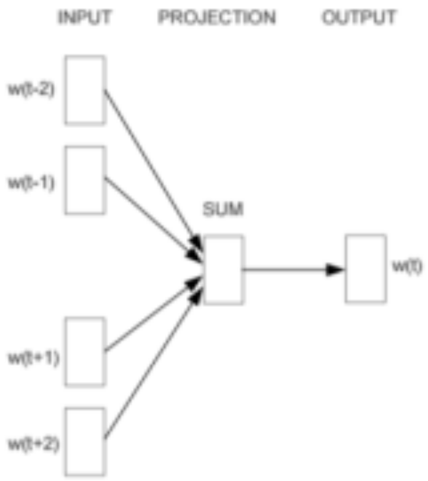
Continuous Bag of Words (CBOW)

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

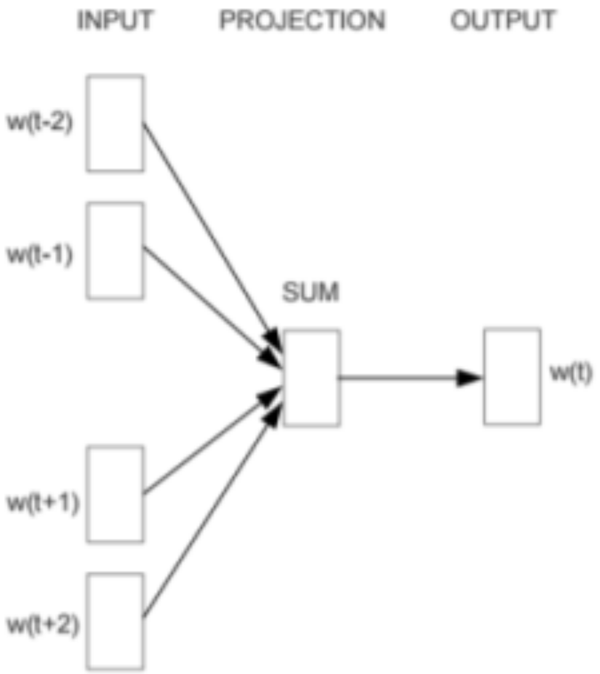


Continuous Bag of N-grams

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

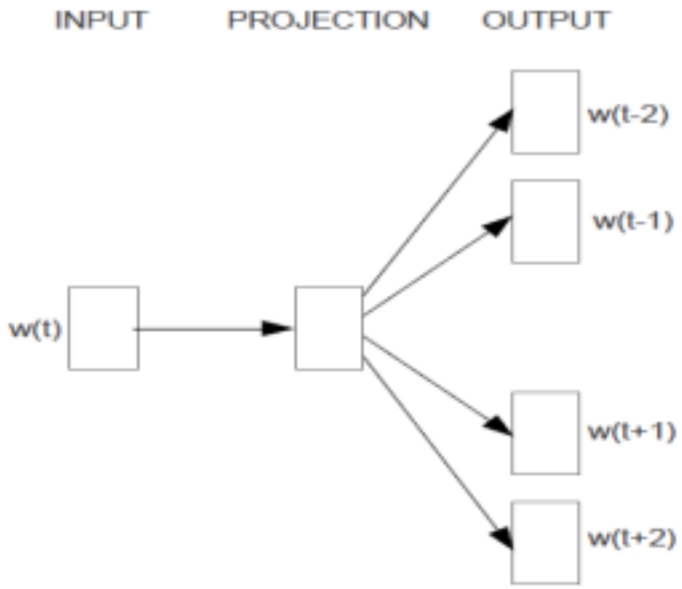


Skipped N-grams

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 w1, w2, w3, ….wT skipped n-gram model tries to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\left[\sum_{j=-k}^{k}\log p(w_{t+j}|w_t)\right]$$



Skip-gram

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.

$$
\begin{array}{c}
\begin{array}{ccccccccc} \text{John} & \text{likes} & \text{to} & \text{watch} & \text{movies} & \text{Mary} & \text{too} & \text{also} & \text{football} \end{array} \\
\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}
\end{array}
$$

Hierarchical Softmax

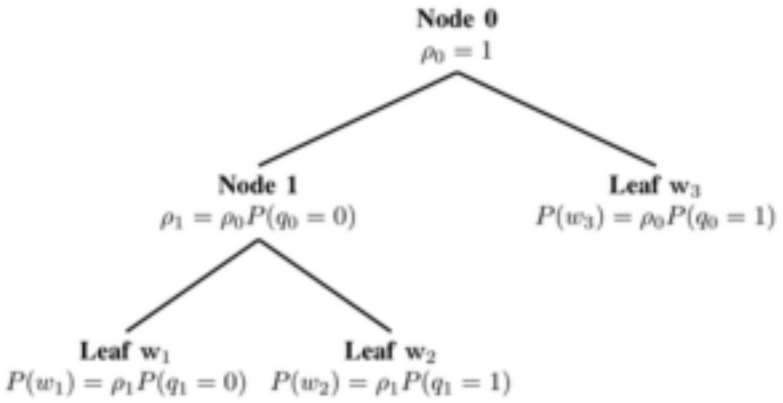
First proposed by Morin and Bengio in Hierarchical Probabilistic Neural Network Language Model

Inspired by the binary tree.

$\text{Log}_2(\text{N})$ instead of N

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

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$



Neural Network Linear Model Model Architecture

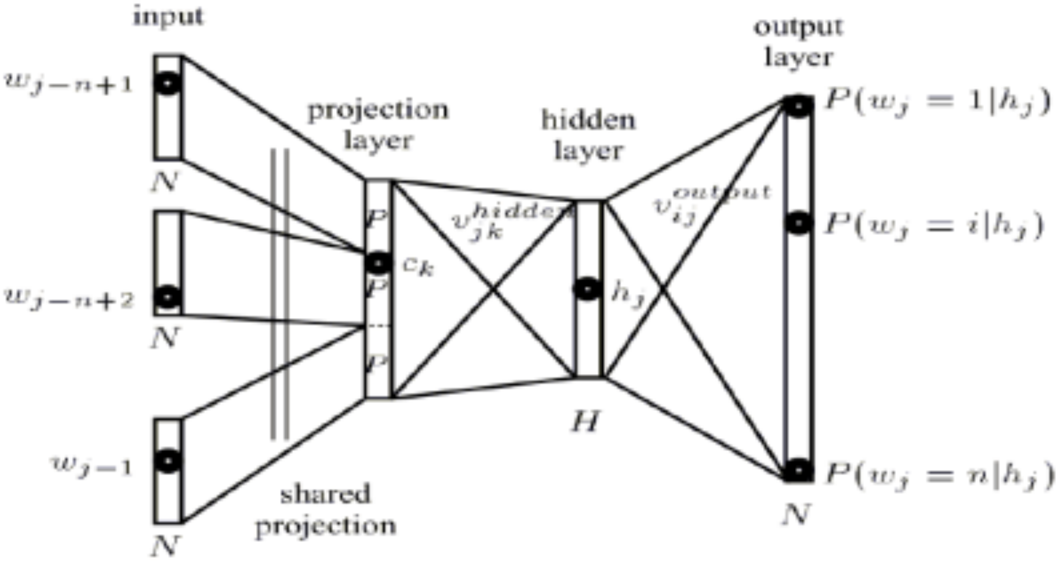


Figure: *Feedforward neural network based LM used by Y. Bengio and H. Schwenk*

Complexity of Neural Network Linear Model (NNLM)

Complexity: Q = N X D + N X D X H + H X 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 X V is the dominating term.

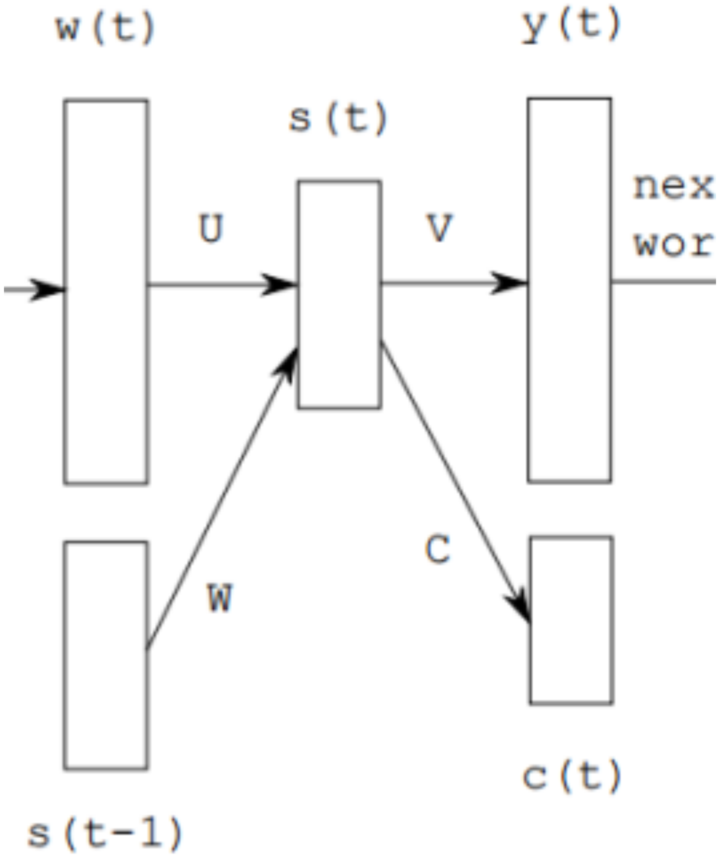
Using Hierarchical Softmax, we can get it down to H X $\log_2 V$

Recurrent Neural Net Language Model

Complexity: Q = H X H + H X V

Dominating term = H X V

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



FastText Architecture

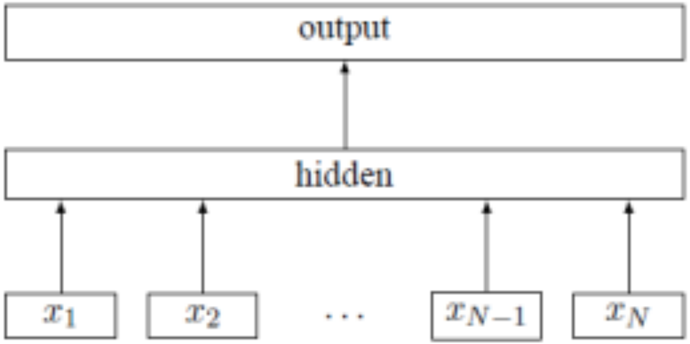


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

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

Fast Text Architecture (cont.)

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

Dataset	Classes	Train Samples	Test Samples
AG's News	4	120,000	7,600
Sogou News	5	450,000	60,000
DBPedia	14	560,000	70,000
Yelp Review Polarity	2	560,000	38,000
Yelp Review Full	5	650,000	50,000
Yahoo! Answers	10	1,400,000	60,000
Amazon Review Full	5	3,000,000	650,000
Amazon Review Polarity	2	3,600,000	400,000

Sentiment Analysis: Performance

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

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.

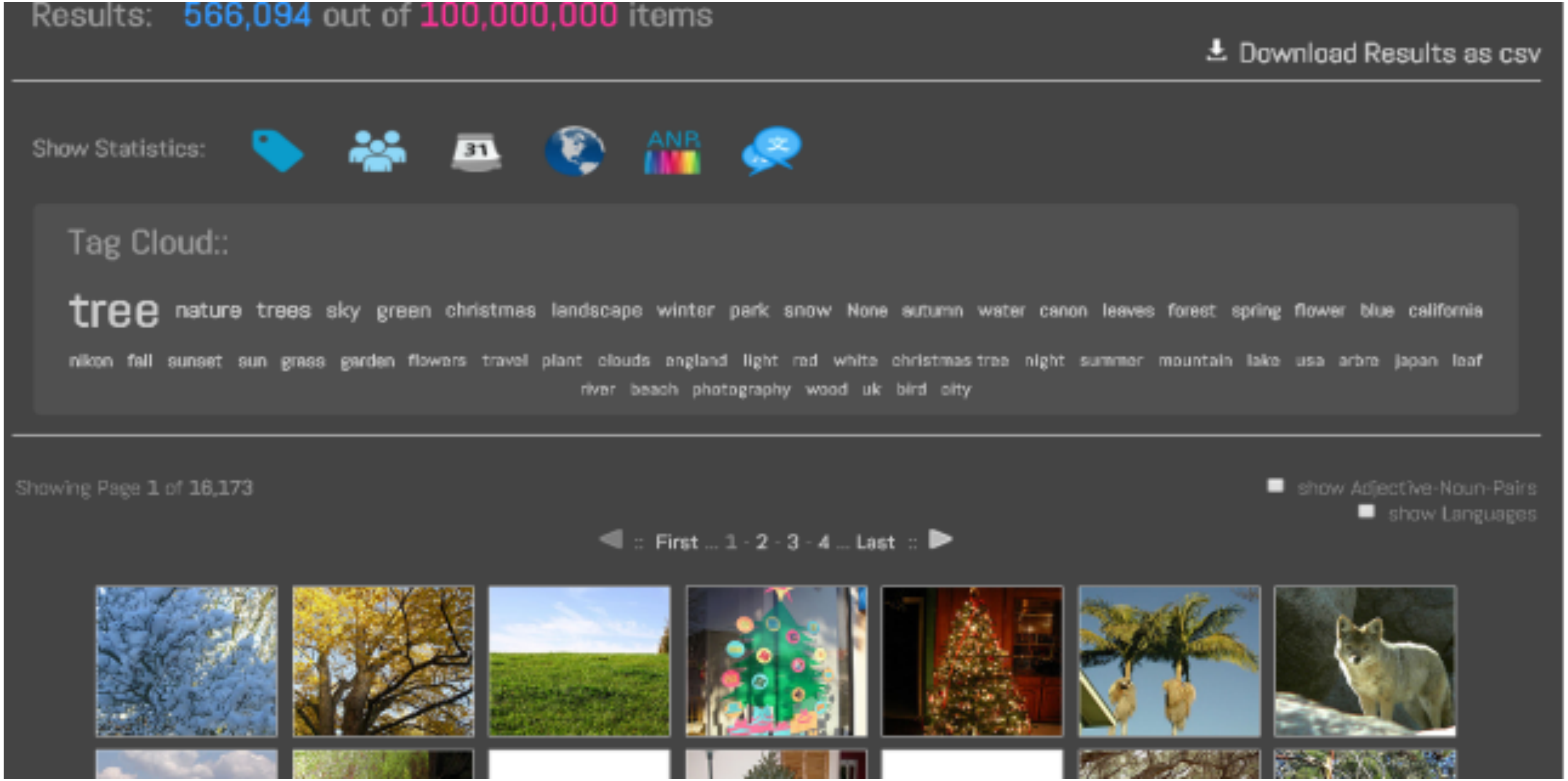
Sentiment Analysis: Speed

	Zhang and LeCun (2015)		Conneau et al. (2016)			fastText
	small char-CNN	big char-CNN	depth=9	depth=17	depth=29	$h = 10$, bigram
AG	1h	3h	24m	37m	51m	1s
Sogou	-	-	25m	41m	56m	7s
DBpedia	2h	5h	27m	44m	1h	2s
Yelp P.	-	-	28m	43m	1h09	3s
Yelp F.	-	-	29m	45m	1h12	4s
Yah. A.	8h	1d	1h	1h33	2h	5s
Amz. F.	2d	5d	2h45	4h20	7h	9s
Amz. P.	2d	5d	2h45	4h25	7h	10s

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/


Results: 566,094 out of 100,000,000 items
Download Results as csv
Show Statistics:
31
ANR
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 england light red white christmas tree night summer mountain lake usa arbre japan leaf
river beach photography wood uk bird city
Showing Page 1 of 16,173
show Adjective-Noun-Pairs
show Languages
:: 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).

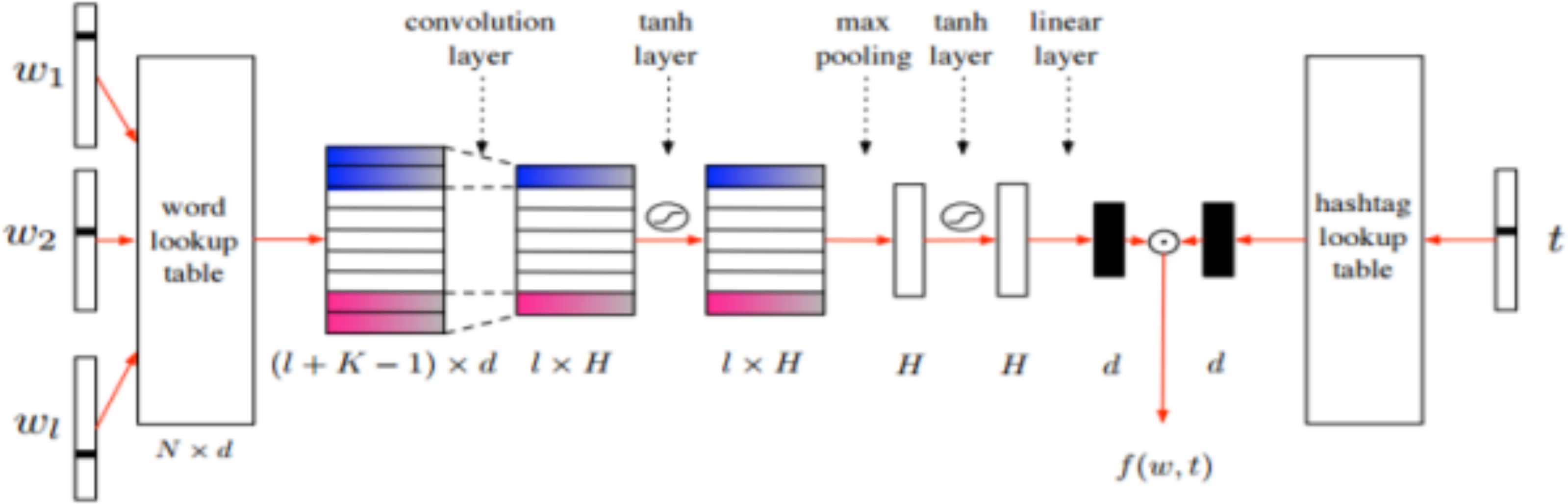
Validation Set: 930,497

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

Model	prec@1	Running time	
		Train	Test
Freq. baseline	2.2	-	-
Tagspace, $h = 50$	30.1	3h8	6h
Tagspace, $h = 200$	35.6	5h32	15h
`fastText`, $h = 50$	31.2	6m40	48s
`fastText`, $h = 50$, bigram	36.7	7m47	50s
`fastText`, $h = 200$	41.1	10m34	1m29
`fastText`, $h = 200$, bigram	46.1	13m38	1m37

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.