Large scale machine learning for text understanding (and computer vision)

Armand Joulin

Facebook AI Research
ajoulin@fb.com

Introduction

- Text classification is core to many problems (information retrieval or web search).
- Depending on the application, it requires:
    - Pre-trained word representations (word2vec),
    - Scaling to massive amount of data,
    - Small memory footprint for embedded system
- We have developed a library designed to solve these problems, called `fastText`:

  `https://github.com/facebookresearch/fastText`

Learning word representations

- Goal: learn a continuous representation of words using massive amount of data
- Idea: learn representation to *predict well* its context (Harris, 1954; Firth, 1957).
- How: by framing it as binary classification problem as in word2vec (Mikolov et al., 2013).

Learning word representations

- A score between words and their context $s(w, c)$ is maximized
- Word2vec skipgram uses a dot product, $s(w, c) = w^\top c$
- However this ignores the structure of words.

Learning word representations

- Instead, represent a word as bag of character *n*-grams:

$$\text{skiing} = \{ \text{ski, skii, kiin, iing, ing} \}$$

- Similarity becomes the dot product between context and this representation:

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top c.$$

Enriching Word Vectors with Subword Information.
P. Bojanowski, E. Grave, A. Joulin, T. Mikolov.
https://arxiv.org/abs/1607.04606

Technical details

- *n*-grams between 3 and 6 characters.
- Hashing trick (Weinberger et al., 2009).
- Stochastic gradient descent with linear decay and Hogwild (Recht et al., 2011)
- Only $\times 2$ slower than word2vec.

Experiments – word similarity

		skipgram	cbow	fastText
Ar	WS353	51	52	**55**
De	Gur350	61	62	**70**
	Gur65	78	78	**81**
	ZG222	35	38	**44**
En	RW	43	43	**47**
	WS353	72	**73**	71
Es	WS353	57	58	**59**
Fr	RG65	70	69	**75**
Ro	WS353	48	52	**54**
Ru	HJ	59	60	**66**

Table: Correlation between human judgement and similarity scores. Models trained on normalized wikipedia dumps.

Experiments – word similarity

	De		En		Es	Fr
	Gur350	ZG222	WS	RW	WS	RG
Luong et al. (2013)	-	-	64	34	-	-
Qiu et al. (2014)	-	-	65	33	-	-
Soricut and Och (2015)	64	22	**71**	42	47	**67**
`fastText`	**73**	**43**	**73**	**48**	**54**	**69**
Botha and Blunsom (2014)	56	25	39	30	28	45
`fastText`	**66**	**34**	**54**	**41**	**49**	**52**

Table: Spearman's rank correlation between human judgement and model scores.

Large scale text classification

- A document is represented as a bag of words/n-grams.
- Given labeled data (x_n, y_n), minimize the softmax loss:

$$\sum_{n=1}^{N} \ell(y_n,\ VUx_n).$$

Equivalent to linear model with rank constraint.

Bag of Tricks for Efficient Text Classification.
A. Joulin, E. Grave, P. Bojanowski, T. Mikolov.
https://arxiv.org/abs/1607.01759

Large scale text classification

We use a set of standard tricks to speed-up training:

- Feature hashing for *n*-grams (Agarwal et al., 2014);
- Hierarchical softmax for large output spaces (Goodman, 2001);
- Pre-trained word vectors.
- SGD + Hogwild

Experiments – Small output space

	Zhang et al. (2015)		Conneau et al. (2016)		fastText	
AG	87.2	3h	91.3	51m	92.5	1s
Amz. F.	59.5	5d	63.0	7h	60.2	9s
DBpedia	98.3	5h	98.7	1h	98.5	2s
Yah. A.	71.2	1d	73.4	2h	72.3	5s
Yelp F.	62.0	-	64.7	1h12	63.9	4s

Table: Test accuracy [%] and training time per epoch.

Experiments – Large output space

Model	prec@1	Running time	
		Train	Test
Freq. baseline	2.2	-	-
Tagspace (Weston et al., 2011)	35.6	5h32	15h
`fastText`	46.1	13m38	1m37

Table: Tag prediction on YFCC100M. The output space contains +300K labels.

Compressing text classifiers

- Linear models are state-of-the-art and extremely efficient
- However they require a lot of memory
- `fastText` models are often + 100Mb.
- Cannot fit on embedded device!

FastText.zip: Compressing text classification models.
A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jegou, T. Mikolov.
http://openreview.net/forum?id=SJc1hL5ee

Compressing text classifiers

2 key ideas:

- Apply quantization approaches designed for retrieval
- Prune dictionary based on the norm of the embeddings

Leads to compression of $\times 100 - 1000$ with (almost) no drop of performance or speed.

FastText.zip: Compressing text classification models.
A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jegou, T. Mikolov.
http://openreview.net/forum?id=SJc1hL5ee

Compressing text classifiers

- We use Product Quantizer (PQ) (Jegou et al., 2011)
- Product quantization approximates a vector x by

$$\hat{x} = [q_1(x_1), ..., q_k(x_k)],$$

  where x_i are subvectors and q_i are k-means quantizers.
- $k = d/2$ and 2^8 centroids per k-means: compression of $\times 8$.
- Simple to implement and easily parallelizable.

Compressing text classifiers

A few additional tricks:

- Compress norm and vectors seperately
- Bottom-up compression: first the embedding, then the classifiers.
- Retraining the classifier often helps but is costly.

Compressing text classifiers

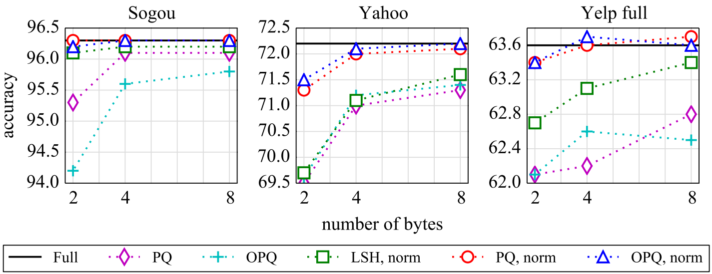


Figure: Comparison of different compression algorithm.

Compressing text classifiers

- Second strategy: feature selection.
- Cast it as finding the closest sparse model under coverage constraints
- Approximate solution by selecting K largest embeddings that covers the dataset.

Compressing text classifiers

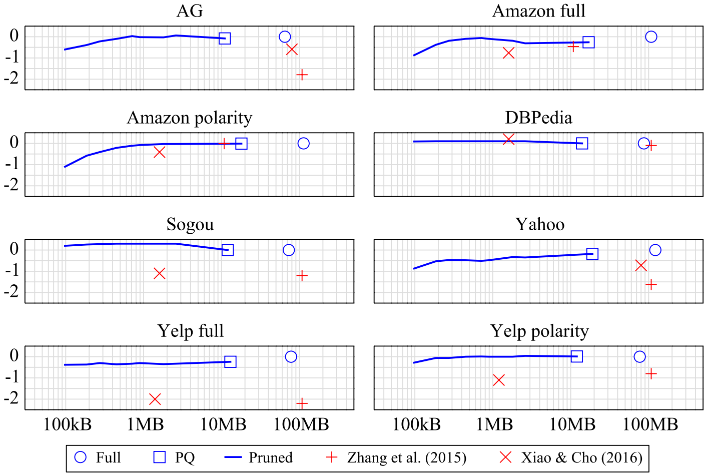


Figure: Loss of accuracy vs model size.

Extreme compression

Dataset	full		64KiB	32KiB	16 KiB
AG	65M	92.1	91.4	90.6	89.1
Amazon full	108M	60.0	58.8	56.0	52.9
Amazon pol.	113M	94.5	93.3	92.1	89.3
DBPedia	87M	98.4	98.2	98.1	97.4
Sogou	73M	96.4	96.4	96.3	95.5
Yahoo	122M	72.1	70.0	69.0	69.2
Yelp full	78M	63.8	63.2	62.4	58.7
Yelp pol.	77M	95.7	95.3	94.9	93.2
Average diff. [%]		0	-0.8	-1.7	-3.5

Table: Performance of very small models.

Summary

- Simple linear models are fast and get good accuracy.
- With standard compression techniques, they hve small memory footprint with almost no drop in accuracy or speed.
- Code available online:

  `https://github.com/facebookresearch/fastText`

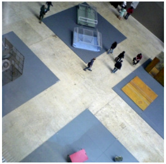
Future work

- Use better parallelization approaches for a greater speed-up (Smith et al., 2016),
- Compress while training with sparsity inducing norms (Meier et al., 2008; Bach et al., 2012),
- Revisit efficent features to extend it to images (Bay et al., 2008; Calonder et al., 2010)

Why efficient visual features are important

the veranda hotel portixol palma

plane approaching zrh avro regional jet rj

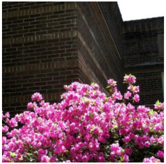


not as impressive as embankment that s for sure

student housing by lungaard tranberg architects in copenhagen click here to see where this photo was taken

Unusual is commonplace

article in the local paper about all the unusual things found at otto s home



this was another one with my old digital camera i like the way it looks for some things though slow and lower resolution than new cameras another problem is that it s a bit of a brick to carry and is a pain unless you re carrying a bag with some room it s nearly x x and weighs ounces new one is x x and weighs ounces i underexposed this one a bit did exposure bracketing script underexposure on that camera looks melty yummy gold kodak film like

Figure: Six randomly picked photos and captions from Flickr.

Why efficient visual features are important

- Current approaches too slow to train on "webly" datasets (several weeks on 100M images with 4 high-end GPUs).
- But there are evidence that more data is better than annotated data.

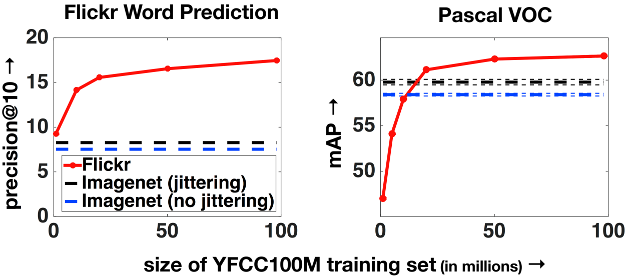


Figure: Alexnet trained on Imagenet vers trained on Flickr100M.

Learning Visual Features from Large Weakly Supervised Data.
A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache.
European Conference on Computer Vision, 2016.

Why efficient visual features are important

- Training simpler models on more data works often better than complex model on less data.

	COCO-5K			Flickr-30K		
	R@1	R@5	R@10	R@1	R@5	R@10
STD-RNN (Socher et al., 2014)	–	–	–	9.6	29.8	41.1
BRNN (Karpathy and Fei-Fei, 2015)	16.5	39.2	52.0	22.2	48.2	61.4
Kiros et al. (Kiros et al., 2014)	–	–	–	23.0	50.7	62.9
NIC (Vinyals et al., 2015)	–	–	–	23.0	–	63.0
Jelinek-Mercer + finetuning	**17.8**	**41.9**	**53.9**	**28.6**	**54.7**	**66.0**

Table: Comparison of language models for caption retrieval on the COCO-5K and Flickr-30K datasets.

Learning Visual N-Grams from Web Data.
A. Li, A. Jabri, A. Joulin and L. van der Maaten.
submitted, 2016.

Thank you!

Experiments – effect of *n*-gram size

Semantic					
	2	3	4	5	6
2	59	55	56	59	60
3		60	58	60	62
4			62	62	63
5				64	64
6					**65**

Syntactic					
	2	3	4	5	6
2	45	50	53	54	55
3		51	55	55	**56**
4			54	**56**	**56**
5				**56**	**56**
6					54

Table: Study of the effect of *n*-gram size on performance (language: German).

Experiments

Model	AG	DBP	Yelp F.	Yah. A.	Amz. F.
BoW (Zhang et al., 2015)	88.8	96.6	58.0	68.9	54.6
ngrams (Zhang et al., 2015)	92.0	98.6	56.3	68.5	54.3
ngrams TFIDF (Zhang et al., 2015)	92.4	98.7	54.8	68.5	52.4
`fastText`	92.5	98.6	63.9	72.3	60.2

Table: Test accuracy [%] on datasets with small output space.

References I

Agarwal, A., Chapelle, O., Dudík, M., and Langford, J. (2014). A reliable effective terascale linear learning system. *Journal of Machine Learning Research*, 15(1):1111–1133.

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2012). Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (SURF). *CVIU*, 110(3):346–359.

Botha, J. A. and Blunsom, P. (2014). Compositional morphology for word representations and language modelling. In *Proc. ICML*.

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *ECCV*.

Conneau, A., Schwenk, H., Barrault, L., and Lecun, Y. (2016). Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*.

References II

Firth, J. R. (1957). *Papers in linguistics, 1934-1951*. Oxford University Press.

Goodman, J. (2001). Classes for fast maximum entropy training. In *ICASSP*.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3).

Jegou, H., Douze, M., and Schmid, C. (2011). Product quantization for nearest neighbor search. *IEEE Trans. PAMI*.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Kiros, J. R., Salakhutdinov, R., and Zemel, R. S. (2014). Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539.

Luong, T., Socher, R., and Manning, C. D. (2013). Better word representations with recursive neural networks for morphology. In *Proc. CoNLL*.

References III

Meier, L., Van De Geer, S., and Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Adv. NIPS*.

Qiu, S., Cui, Q., Bian, J., Gao, B., and Liu, T.-Y. (2014). Co-learning of word representations and morpheme representations. In *Proc. COLING*.

Recht, B., Re, C., Wright, S., and Niu, F. (2011). Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Adv. NIPS*.

Smith, V., Forte, S., Ma, C., Takac, M., Jordan, M. I., and Jaggi, M. (2016). Cocoa: A general framework for communication-efficient distributed optimization. *arXiv preprint arXiv:1611.02189*.

Socher, R., Karpathy, A., Le, Q., Manning, C. D., and Ng, A. (2014). Grounded compositional semantics for finding and describing images with sentences. *TACL*.

References IV

Soricut, R. and Och, F. (2015). Unsupervised morphology induction using word embeddings. In *Proc. NAACL*.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *CVPR*.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., and Attenberg, J. (2009). Feature hashing for large scale multitask learning. In *ICML*.

Weston, J., Bengio, S., and Usunier, N. (2011). Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Adv. NIPS*.