自然语言处理系列（5）：FastText

这篇文章翻译自Bag of Tricks for Efficient Text Classification

本文探讨了一个简单而有效的文本分类baseline。我们的实验表明，快速文本分类器在精度上通常与深度学习分类器相当，并且在训练和评估方面的速度要快得多。我们可以在不到10分钟的时间内使用标准的多核CPU对超过10亿单词的快速文本进行训练，在不到一分钟的时间内对312000类的50万个句子进行分类。

一、Introduction

文本分类是自然语言处理中的一项重要任务，它有很多应用，比如web搜索、信息检索、排序和文档分类等。最近，基于神经网络的模型变得越来越流行(Kim, 2014; Zhang and LeCun, 2015; Conneau et al., 2016)。虽然这些模型在实践中取得了很好的性能，但是它们在训练和测试时都比较慢，这限制了它们在非常大的数据集上的使用。

与此同时，线性分类器通常被认为是文本分类问题的强大baseline(Joachims, 1998; McCallum and Nigam, 1998; Fan et al., 2008)。尽管它们很简单，但如果使用正确的特性，它们通常会获得最先进的性能(Wang and Manning, 2012)。它们也有可能扩展到非常大的语料库。

在这篇文章中，我们探讨了在文本分类的背景下，将这些baseline扩展到一个很大的输出空间的大型语料库。受最近的高效词汇表示法的启发(Mikolov et al.， 2013;Levy et al.， 2015)，我们展示了具有秩约束和快速损失逼近的线性模型，它在10分钟内可以训练10亿个单词，同时达到与最先进的水平相同的性能。我们在两个不同的任务上评价我们的方法FastText的质量，即标记预测和情绪分析。

二、Model architecture

一个简单而有效的文本分类baseline是将文本描述成bag of word (BoW)，训练一个线性分类器，例如，逻辑回归或SVM(Joachims, 1998; Fan et al., 2008)。然而，线性分类器并不在特征和类别之间共享参数。这可能限制了它们在大的输出空间环境中的泛化，在这个大的输出空间中，一些类别的样本可能非常少。解决这个问题的常用方法是将线性分类器分解成低秩矩阵(Schutze, 1992;Mikolov et al., 2013)或者使用多层神经网络 （Collobert and Weston, 2008 ; Zhang et al., 2015)。

图1显示了一个具有秩约束的简单线性模型。

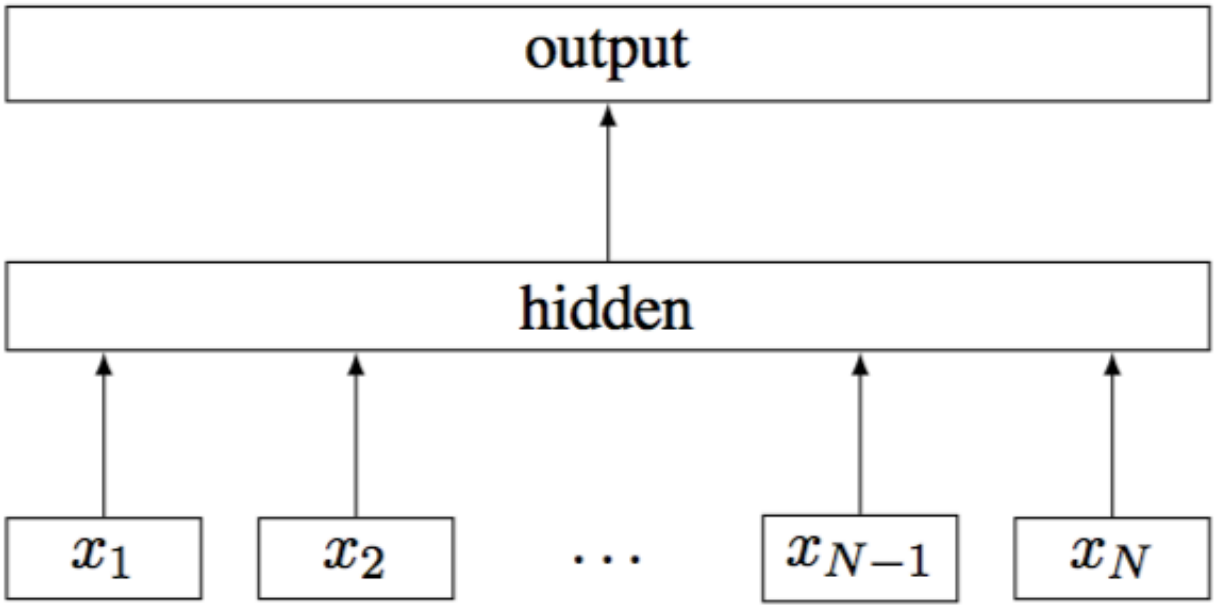

Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

第一个权重矩阵A是对单词的查找表。然后对文本的中所有词向量求平均得到一个文本的表示，依次进入线性分类器。这些文本表示是一个潜在的可重用的隐藏变量。这个架构类似于Mikolov et al.(2013)的cbow模型，只是它中间的单词被一个标签代替。我们使用将$softmax$函数f计算预定义类的概率分布。对一组N个文档，最小化负对数似然函数

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

其中x_n是第n个文档的标准化特征，y_n是类别标签，A和B是权重矩阵。该模型通过随机梯度下降和线性衰减学习率在多个cpu上进行异步训练。

2.1 Hierarchical softmax

当类的数量很大时，计算线性分类器的计算代价很高。更准确地说，计算复杂度是$O(kh)$，k是类的数量，h是文本表示的维度。为了提高我们的运行时间，我们在Huffman编码树(Mikolov et al.， 2013)的基础上使用了一个分层的softmax (Goodman, 2001)。在训练过程中，计算复杂度下降到$O(hlog_2(k))$。在搜索最有可能的类时，分层的softmax在测试时也是有利的。每个节点都与从根到该节点的路径的概率相关。如果深度$l+1$的节点，它的父节点为n_1、n_2、n_l，则它的概率为

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

这意味着子节点的概率总是低于其父节点的概率。探索树的深度优先搜索和跟踪最大概率的叶子允许我们丢弃任何与一个小概率相关的分支。在实践中，我们观察到在测试时将复杂度降低到$O(hlog_2(k))$。使用二进制堆，这种方法进一步扩展到以$O(log(T))$的代价计算T-top目标。

2.2 N-gram features

词袋中的词顺序是固定的，但明确地把这个顺序考虑进去通常是非常昂贵的。相反，我们使用一个n-grams作为额外的特征，以获取关于局部词序的部分信息。这在实践中非常有效，同时实现了与明确使用顺序的方法(Wang and Manning, 2012)差不多的结果。

我们通过一个哈希（hashing）技巧维持了快速且有效的 n-gram 映射（Weinberger et al., 2009），这个哈希技巧带有与 Mikolov et al.（2011）论文中相同的哈希函数和 10M bins（如果我们仅使用 bigram 的话），否则就需要 100M bin.

三、Experiments

我们在两个不同的任务上评估fastText。首先，我们将其与现有的文本分类器对情绪分析问题进行比较。然后，我们评估其在标记预测数据集上扩展到大的输出空间的能力。请注意,我们的模型可以用Vowpal Wabbit库来实现,但是在实践中我们看到,我们的实现至少有2到5倍的提升。

3.1 Sentiment analysis

> Datasets and baselines

我们采用了同样的8个数据集，并采用了Zhangetal(2015)的评价协议。我们比较了了Zhang et al. (2015)的n-gram和TF-IDF 的baseline，以及Zhang and LeCun(2015)的字符级卷积模型(char-cnn)，基于字符的卷积循环网络(char-CRNN) (Xiao and Cho, 2016)和Conneau等人(2016)的深度卷积网络(VDCNN)。我们还比较了Tang等人(2015)的评估方案。我们比较了他们的主要baseline，以及基于循环网络的两种方法(Conv-GRNN和LSTM-GRNN)。

> Results

我们在图1中展示了结果。我们使用10个隐藏单元，并在5个epochs中运行fastText，其学习率从{0.05,0.1,0.25,0.5}中选择。

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.

在这个任务中，添加bigram信息可以提高1-4%的性能。总体而言，我们的准确性略好于char-CNN和char-CRNN，差于VDCNN。请注意，我们可以通过使用更多的n-gram来略微提高精确度，例如使用trigrams，在搜狗数据集上的性能达到了97.1%。最后，图3显示了我们的方法与Tang et al.(2015)中的方法相竞争。

Model	Yelp'13	Yelp'14	Yelp'15	IMDB
SVM+TF	59.8	61.8	62.4	40.5
CNN	59.7	61.0	61.5	37.5
Conv-GRNN	63.7	65.5	66.0	42.5
LSTM-GRNN	65.1	67.1	67.6	45.3
`fastText`	64.2	66.2	66.6	45.2

Table 3: Comparision with Tang et al. (2015). The hyper-parameters are chosen on the validation set. We report the test accuracy.

我们对验证集上的超参数进行调优，并观察到使用n-grams达到5时性能最佳。与Tang et al.(2015)不同，fastText不使用预先训练的词嵌入，这可以解释1%的准确性差异。

> Training time

CNN和VDCNN都用NVIDIA Tesla K40 GPU训练，而我们的模型在CPU上使用了20个线程。表2显示使用卷积的方法比fastText慢几个数量级。虽然利用10倍的速度提升的CUDA来加速char-CNN的训练，FastText花了不到一分钟的训练。Tang et al.(2015)的GRNNs方法在CPU上以单个

线程的时间为12小时左右。随着数据量增加，与神经网络方法相比，我们的模型加速至少达到了15000倍

	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.

3.2 Tag prediction

> Dataset and baselines.

为了测试我们的方法的可扩展性，在YFCC100M数据集(Thomee et al.， 2016)上进行了进一步的评估，该数据集包含了近1亿张带有说明、标题和标签的图像。我们专注于根据标题和标题预测标签(我们不使用图像)。我们删除了出现少于100次的单词和标记，并将数据分成训练集、验证集和测试集。该训练集包含91,188,648个样本(1.5B tokens)。验证集有930497个样本，测试集有543,424个样本。词汇量为297,141，有312,116个独特的标签。我们将发布一个脚本，重新创建这个数据集，以便可以复制我们的数据。我们报告的精度为1。

我们考虑一个基于频率的baseline，它预测最频繁的标记。我们也比较了Tagspace (Weston et al.， 2014)，这是一个类似于我们的标签预测模型，但是基于Weston et al.(2011)的Wsabie模型。虽然Tagspace模型是使用卷积来描述的，但是我们考虑的是线性版本，它实现了类似的性能，但是速度要快得多。

> Results and training time

表5给出了fastText和baseline的比较。我们运行5个epochs的fastText，并将其与隐藏层的两个不同尺寸的Tagspace进行比较，即50和200。这两种模型都实现了一种类似的性能，它有一个隐藏的小层，但是添加了bigrams，这就大大提高了精度。在测试时，Tagspace需要计算所有类的分数，这使得它相对较慢，而我们的快速推理在类的数量很大(这里超过300K)时，会有显著的加速增长。总的来说，我们的模型速度超过了一个数量级，且获得了更高质量的效果。测试的加速阶段更显著(600倍加速)。表4显示了一些定性的例子。

Input	Prediction	Tags
taiyoucon 2011 digitals: individuals digital photos from the anime convention taiyoucon 2011 in mesa, arizona. if you know the model and/or the character, please comment.	#cosplay	#24mm #anime #animeconvention #arizona #canon #con #convention #cos **#cosplay** #costume #mesa #play #taiyou #taiyoucon
2012 twin cities pride 2012 twin cities pride parade	#minneapolis	#2012twincitiesprideparade **#minneapolis** #mn #usa
beagle enjoys the snowfall	#snow	#2007 #beagle #hillsboro #january #maddison #maddy #oregon **#snow**
christmas	#christmas	#cameraphone #mobile
euclid avenue	#newyorkcity	#cleveland #euclidavenue

Table 4: Examples from the validation set of YFCC100M dataset obtained with `fastText` with 200 hidden units and bigrams. We show a few correct and incorrect tag predictions.

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.

四、Discussion and conclusion

在本文中，我们提出了一种简单的文本分类baseline方法。不像从word2vec中训练出来的不受监督的单词向量，我们的单词特征可以被平均起来组成一个好的句子表示。在一些任务中，fastText获得的性能与最近提出的以深度学习为灵感的方法相比，速度更快。虽然深层神经网络在理论上比浅层模型具有更高的表征能力，但尚不清楚简单的文本分类问题，如情绪分析是否是正确的评价方法。我们将发布我们的代码，以便研究社区能够轻松地构建我们的工作。