NLP详细教程：手把手教你用ELMo模型提取文本特征（附代码&论文）

数据分析 2019-04-19

以下文章来源于数据派THU ，作者PRATEEK JOSHI

数据派THU

发布清华校内相关科研动态、教学成果及线下活动

作者：PRATEEK JOSHI 翻译：韩国钧 校对：李 浩

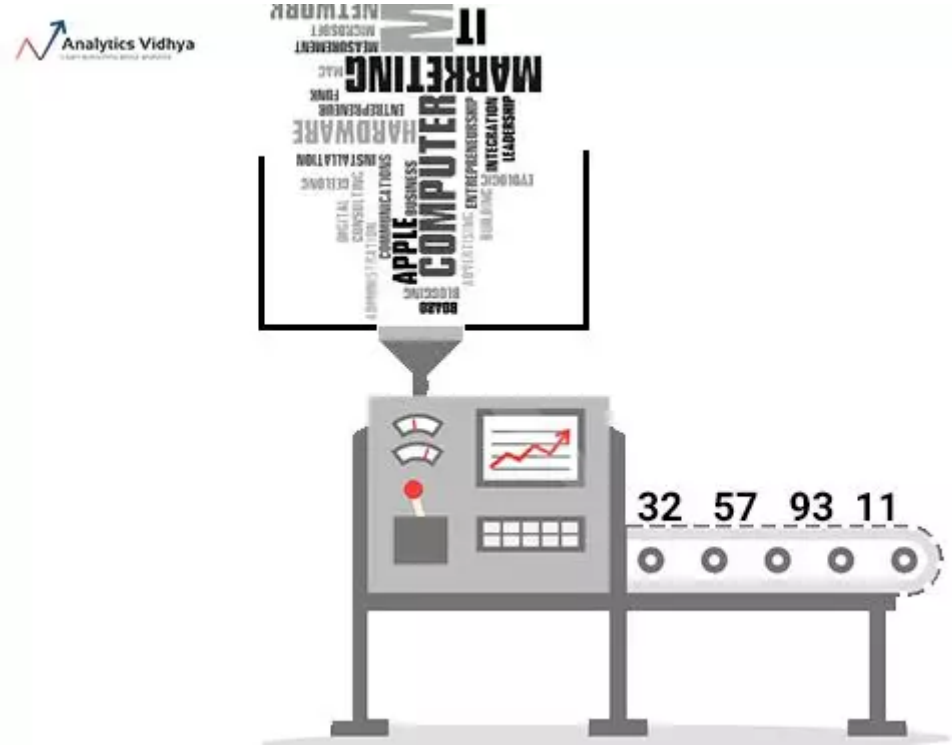
本文约**3500字**，建议阅读**15分钟。**
本文将介绍ELMo的原理和它与传统词嵌入的区别，然后通过实践来展示其效果。

简介

我致力于研究自然语言处理（NLP）领域相关问题。每个NLP问题都是一次独特的挑战，同时又反映出人类语言是多么复杂、美丽又绝妙。

但是一个让NLP从业者头疼的问题是机器无法理解语句的真正含义。是的，我指的是自然语言处理中的上下文问题。传统的NLP技术和架构能很好地处理基础任务，但当我们尝试将上下文纳入变量时其效果就会下降。

近18个月以来NLP领域的格局发生了重大变化，诸如Google的BERT和Zalando的Flair等NLP模型已经能够分析语句并掌握上下文中的信息。



ELMo模型

能够理解上下文语境是NLP领域的一项重大突破，这归功于ELMo（Embeddings from Language Models），它是AllenNLP研发的一种最先进的NLP架构。当你读完这篇文章，你会和我一样成为ELMo的忠实粉丝。

在这篇文章中，我们会探索ELMo（嵌入语言模型），并通过python使用它在一个真实的数据集上构建一个令人兴奋的NLP模型。

注：这篇文章假设你熟悉多种word embeddings和LSTM（Long short-term memory）结构，你可以参阅以下文章来了解有关这些专题的更多信息：

- An Intuitive Understanding of Word Embeddings

(https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/?utm_medium=ELMoNLParticle&utm_source=blog)

- Essentials of Deep Learning : Introduction to Long Short Term Memory

(https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/?utm_medium=ELMoNLParticle&utm_source=blog)

目录

3. ELMo与其他词嵌入的区别是什么?
4. 在python中应用ELMo模型进行文本分类：
   - 理解问题陈述
   - 数据集介绍
   - 导入库
   - 导入和检查数据
   - 文本清洗和预处理
   - 简要介绍TensorFlow Hub
   - 准备ELMo模型向量
   - 构建模型并评估
5. 我们还能用ELMo做什么?
6. 结语

1. 什么是ELMo?

我们提到的ELMo并不是《芝麻街》（Sesame Street）中的角色，这也是一个体现了上下文语境的重要性的典型例子。

ELMo是一种在词向量（vector）或词嵌入（embedding）中表示词汇的新方法。这些词嵌入方法在下列几种NLP问题中能有效生成最先进(SOAT)的结果：

Task	Previous SOTA		ELMo + Baseline
SQuAD	SAN	84.4	85.8
SNLI	Chen et al (2017)	88.6	88.7 +/- 0.17
SRL	He et al (2017)	81.7	84.6
Coref	Lee et al (2017)	67.2	70.4
NER	Peters et al (2017)	91.93 +/- 0.19	92.22 +/- 0.10
Sentiment (5-class)	McCann et al (2017)	53.7	54.7 +/- 0.5

全球的自然语言处理学家都开始在学术或应用领域的NLP问题中使用ELMo。建议你查看ELMo的初始论文（https://arxiv.org/pdf/1802.05365.pdf）。通常我不会建议大家去读学术论文因为它们往往又长又复杂，但这篇论文不同，它很好地解释了ELMo原理和设计过程。

2. 理解ELMo工作原理

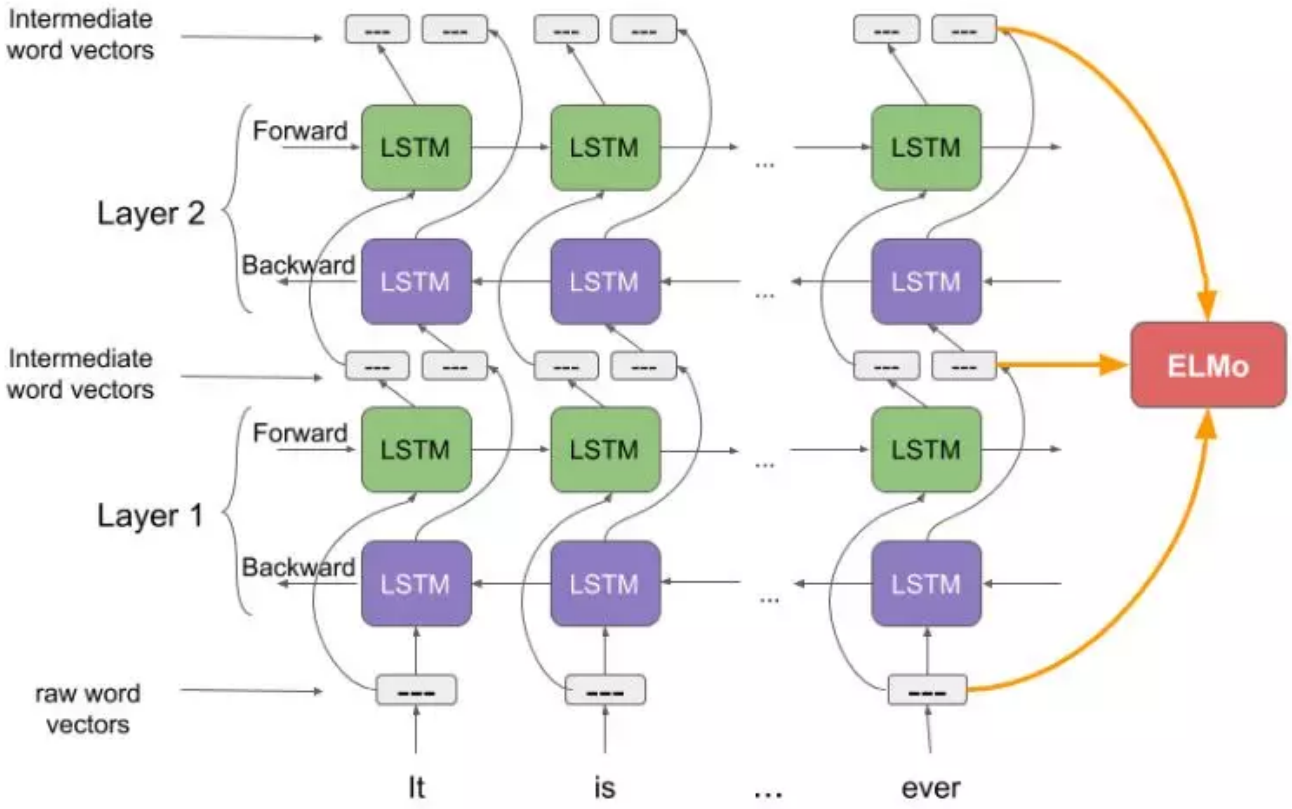
在实践之前让我们需要先直观了解一下ELMo是如何运作的。为什么说这一步很重要？

试想如下场景：你已经成功地从GitHub上下载了ELMo的python代码并在自己的文本数据集上构建了模型，但只得到了一般的结果，所以你需要改进。如果你不理解ELMo的架构你将如何改进呢？如果没有研究过又怎么知道需要调整哪些参数呢？

这种思路适用于其他所有机器学习算法，你不需要了解它们的推导过程但必须对它们有足够的认识来玩转和改进你的模型。

现在，让我们回到ELMo的工作原理。

正如我之前提到的，ELMo的词向量是在双层双向语言模型（two-layer bidirectional language model，biLM）上计算的。这种模型由两层叠在一起，每层都有前向（forward pass）和后向（backward pass）两种迭代。



- 上图中的结构使用字符级卷积神经网络（convolutional neural network, CNN）来将文本中的词转换成原始词向量（raw word vector）
- 将这些原始词向量输入双向语言模型中第一层
- 前向迭代中包含了该词以及该词之前的一些词汇或语境的信息
- 后向迭代中包含了该词之后的信息
- 这两种迭代的信息组成了中间词向量（intermediate word vector）
- 这些中间词向量被输入到模型的下一层
- 最终表示（ELMo）就是原始词向量和两个中间词向量的加权和

因为双向语言模型的输入度量是字符而不是词汇，该模型能捕捉词的内部结构信息。比如beauty和beautiful，即使不了解这两个词的上下文，双向语言模型也能够识别出它们的一定程度上的相关性。

3. ELMo与其他词嵌入的区别是什么?

与word2vec或GLoVe等传统词嵌入不同，ELMo中每个词对应的向量实际上是一个包含该词的整个句子的函数。因此，同一个词在不同的上下文中会有不同的词向量。

你可能会问：这种区别会对我处理NLP问题有什么帮助吗？让我通过一个例子来解释清楚：

我们有以下两个句子：

- I read the book yesterday.
- Can you read the letter now?

花些时间考虑下这两个句子的区别，第一个句子中的动词“read”是过去式，而第二个句子中的“read”却是现在式，这是一种一词多义现象。

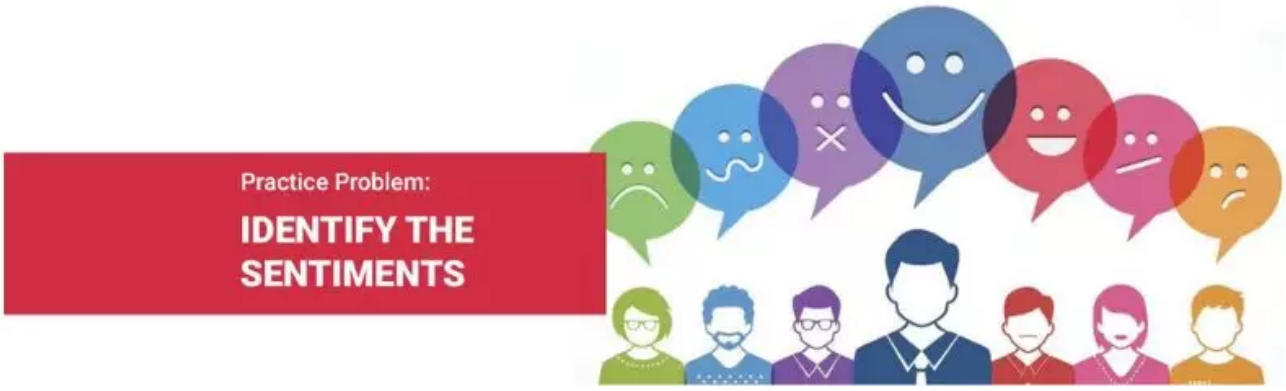
语言是多么精妙而复杂

传统的词嵌入会对两个句子中的词“read”生成同样的向量，所以这些架构无法区别多义词，它们无法识别词的上下文。

与之相反，ELMo的词向量能够很好地解决这种问题。ELMo模型将整个句子输入方程式中来计算词嵌入。因此，上例中两个句子的“read”会有不同的ELMo向量。

4. 实现：在python中应用ELMo模型进行文本分类

现在是你们最期待的部分——在python中实现ELMo！让我们逐步进行：



理解问题陈述

处理数据科学问题的第一步是明确问题陈述，这将是你接下来行动的基础

对于这篇文章，我们已经有如下问题陈述：

情感分析一直是NLP领域中的一个关键问题。这次我们从Twitter上收集了消费者对于生产并销售手机、电脑等高科技产品的多个公司的推文，我们的任务是判断这些推文是否包含负面评价。

这显然是一个文本的二分类任务，要求我们从提取的推文预测情感。

数据集介绍

我们已经分割了数据集：

- 训练集中有7920条推文
- 测试集中有1953条推文

你可以从这里下载数据集

https://datahack.analyticsvidhya.com/contest/linguipedia-codefest-natural-language-processing-1/#data_dictionary

注：数据集中的多数脏话已经被替换成了“$&@*#”，但可能部分推文中还有一些脏话存在。

好了，让我们打开最喜欢的Python IDE开始编程吧！

导入库

导入我们将要用到的库：

```
import pandas as pd
import numpy as np
import spacy
from tqdm import tqdm
import re
import time
import pickle
pd.set_option('display.max_colwidth', 200)
```

导入和检查数据

```
# read data
train = pd.read_csv("train_2kmZucJ.csv")
test = pd.read_csv("test_oJQbWVk.csv")

train.shape, test.shape
```

Output:

```
((7920, 3), (1953, 2))
```

训练集中有7920条推文，训练集中有1953条推文。接下来我们检查下训练集中的类别分布：

```
train['label'].value_counts(normalize = True)
```

Output:

0 0.744192
1 0.255808
Name: label, dtype: float64

这里1代表负面推文，0代表非负面的推文。

现在让我们看一下训练集的前五行：

train.head()

	id	label	tweet
0	1	0	#fingerprint #Pregnancy Test https://goo.gl/h1MfQV #android #apps #beautiful #cute #health #igers #iphoneonly #iphonesia #iphone
1	2	0	Finally a transparant silicon case ^^ Thanks to my uncle :) #yay #Sony #Xperia #S #sonyexperias... http://instagram.com/p/YGEt5JC6JM/
2	3	0	We love this! Would you go? #talk #makememories #unplug #relax #iphone #smartphone #wifi #connect... http://fb.me/6N3LsUpCu
3	4	0	I'm wired I know I'm George I was made that way ;) #iphone #cute #daventry #home http://instagr.am/p/Li_5_ujS4k/
4	5	1	What amazing service! Apple won't even talk to me about a question I have unless I pay them $19.95 for their stupid support!

我们有三列数据，“tweet”列是独立变量，“label”列是目标变量

文本清洗和预处理

理想状况下我们会有一个整洁且结构化的数据集，但目前NLP领域还很难做到。

我们需要花费一定时间来清洗数据，为模型构建做准备。从清洗后的文本中提取特征会变得简单，甚至特征中也会包含更多信息。你会发现你的数据质量越高，模型的表现也就会越好。

所以让我们先清理一下已有的数据集吧。

可以发现有些推文中有URL链接，它们对情感分析没有帮助，所以我们需要移除它们。

```
# remove URL's from train and test
train['clean_tweet'] = train['tweet'].apply(lambda x: re.sub(r'http\S+', '', x))
test['clean_tweet'] = test['tweet'].apply(lambda x: re.sub(r'http\S+', '', x))
```

我们使用正则表达式（Regular Expression）来移除URL。

注：你可以从这里了解正则表达式
https://www.analyticsvidhya.com/blog/2015/06/regular-expression-python/regular-expressions-python?utm_medium=ELMoNLParticle&utm_source=blog

我们现在来做一些常规的数据清理工作：

```
# remove punctuation marks
punctuation = '!"#$%&()*+-/:;<=>?@[\\]^_`{|}~'

train['clean_tweet'] = train['clean_tweet'].apply(lambda x: ''.join(ch for ch in x if ch not in set(
punctuation)))
test['clean_tweet'] = test['clean_tweet'].apply(lambda x: ''.join(ch for ch in x if ch not in set(pu
nctuation)))

# convert text to lowercase
train['clean_tweet'] = train['clean_tweet'].str.lower()
test['clean_tweet'] = test['clean_tweet'].str.lower()

# remove numbers
train['clean_tweet'] = train['clean_tweet'].str.replace("[0-9]", " ")
test['clean_tweet'] = test['clean_tweet'].str.replace("[0-9]", " ")

# remove whitespaces
train['clean_tweet'] = train['clean_tweet'].apply(lambda x:' '.join(x.split()))
test['clean_tweet'] = test['clean_tweet'].apply(lambda x: ' '.join(x.split()))
```

接下来我们将文本标准化，这一步会将词简化成它的基本形式，比如“produces”、“production”和“producing”会变成“product”。通常来讲，同一个词的多种形式并不重要，我们只需要它们的基本形式就可以了。

我们使用流行的spaCy库来进行标准化：

```
# import spaCy's language model
nlp = spacy.load('en', disable=['parser', 'ner'])

# function to lemmatize text
def lemmatization(texts):
    output = []
    for i in texts:
            s = [token.lemma_ for token in nlp(i)]
        output.append(' '.join(s))
    return output
```

在测试集和训练集中进行归类（Lemmatize）：

```
train['clean_tweet'] = lemmatization(train['clean_tweet'])
test['clean_tweet'] = lemmatization(test['clean_tweet'])
```

现在让我们看一下原始推文和清洗后的推文的对比：

train.sample(10)

	id	label	tweet	clean_tweet
3802	3803	0	My father is taking me out to buy me my first gaming console ever - #PS4. #speechless #firsttimeever #sony #playstation	-PRON- father be take -PRON- out to buy -PRON- -PRON- first gaming console ever ps . speechless firsttimeever sony playstation
1314	1315	0	My morning #Toronto #coffee #cup #cheesecake #yum #breakfest #Apple #MacBook #girl #morning … http://instagram.com/p/sVGp2rP51_/	-PRON- morning toronto coffee cup cheesecake yum breakf apple macbook girl morning …
1289	1290	1	Why is there always a new version of iTunes to download?!	why be there always a new version of itune to download
7787	7788	0	So true. Plus when I changed from Macbook Pro to Imac I find that Iphoto and Aperture are no longer available and a kids programme called Photos designed for I know not whom is substituted with no...	so true . plus when i change from macbook pro to imac i find that iphoto and aperture be no longer available and a kid programme call photo design for i know not whom be substitute with no alterna...
6579	6580	0	Sooo loving my #samsung galaxy s3....cant seem to put it down.	sooo love -PRON- samsung galaxy s .... can not seem to put -PRON- down .
2646	2647	0	Gain Followers RT This MUST FOLLOW ME I FOLLOW BACK Follow everyone who rts Gain #iphone #sougofollow +rz6	gain follower rt this must follow -PRON- i follow back follow everyone who rt gain iphone sougofollow rz
3534	3535	0	1 week with the #samsung #note3 ... loving it so far! Not as big as I thought it would be... #androidpic.twitter.com/OPWk5Jgtec	week with the samsung note ... love -PRON- so far not as big as i think -PRON- would be ... androidpic.twitter.comopwk jgtec
3027	3028	0	If Microsoft had anything even close to the iPod Id be the first one to switch over #worstcustomerservice	if microsoft have anything even close to the ipod -PRON- would be the first one to switch over worstcustomerservice
4852	4853	0	Yes! Finally got the latest snapchat update! Now I get the dancing ghost just like everyone else! #samsung #galaxy @SnapchatProbbz	yes finally get the late snapchat update now i get the dancing ghost just like everyone else samsung galaxy snapchatprobbz
7474	7475	1	@gdavidson88 fannies. Don't they know #apple #products and that #Steve #Jobs was #KKK ??? Sent from my iPhone	gdavidson fanny . do not -PRON- know apple product and that steve job be kkk send from -PRON- iphone

仔细查看上图中的两列推文的对比，清洗后的推文变得更加清晰易理解。

然而，在清洗文本这一步中其实还有很多可以做的，我鼓励大家进一步探索数据，去发现文本中可以提升的地方。

简要介绍TensorFlow Hub

等等，TensorFlow跟我们这篇教程有什么关系呢？

TensorFlow Hub是一个允许迁移学习（Transfer learning）的库，它支持用多种机器学习模型处理不同任务。ELMo是其中一例，这也是为什么我们的实现中需要通过TensorFlow Hub来使用ELMo。

我们首先需要安装TensorFlow Hub，你必须安装或升级到1.7版本以上来使用：

```
$ pip install "tensorflow>=1.7.0"
$ pip install tensorflow-hub
```

准备ELMo模型向量

我们现在需要引入事先训练过的ELMo模型。请注意这个模型的大小超过350 mb所以可能需要一些时间来下载。

```
import tensorflow_hub as hub
import tensorflow as tf

elmo = hub.Module("https://tfhub.dev/google/elmo/2", trainable=True)
```

我会先展示如何给一个句子生成ELMo向量，你只需要将一列字符串输入elmo中：

```
# just a random sentence
x = ["Roasted ants are a popular snack in Columbia"]

# Extract ELMo features
embeddings = elmo(x, signature="default", as_dict=True)["elmo"]

embeddings.shape
```

Output:

```
TensorShape([Dimension(1), Dimension(8), Dimension(1024)])
```

这个输出是一个三维张量（1, 8, 1024）：

- 第一个维度表示训练样本的数量，在这个案例中是1；
- 第二个维度表示输入列表中的最大长度，因为我们现在只输入了一个字符串，所以第二个维度就是该字符串的长度8；
- 第三个维度等于ELMo向量的长度。

输入中的每个词都有个长度为1024的ELMo向量。

让我们开始提取测试集和训练集中清洗过推文的ELMo向量。如果想得到整个的推文的ElMo向量，我们需要取推文中每个词的向量的平均值。

我们可以通过定义一个函数来实现：

```
def elmo_vectors(x):

  embeddings = elmo(x.tolist(), signature="default", as_dict=True)["elmo"]

  with tf.Session() as sess:

    sess.run(tf.global_variables_initializer())
    sess.run(tf.tables_initializer())
    # return average of ELMo features
    return sess.run(tf.reduce_mean(embeddings,1))
```

如果使用上述代码来一次性处理所有推文，你可能会耗尽所有内存。我们可以通过将训练集和测试集分割成一系列容量为100条的样本来避免这个问题，然后将他们相继传递给elmo_vectors()函数。

我选择用列表储存这些样本：

```
list_train = [train[i:i+100] for i in range(0,train.shape[0],100)]
list_test = [test[i:i+100] for i in range(0,test.shape[0],100)]
```

现在让我们在这些样本上迭代并提取ELMo向量，这会花很长时间：

```
# Extract ELMo embeddings

elmo_train = [elmo_vectors(x['clean_tweet']) for x in list_train]
elmo_test = [elmo_vectors(x['clean_tweet']) for x in list_test]
```

一旦我们得到了所有向量，我们可以将它们整合成一个数组：

```
elmo_train_new = np.concatenate(elmo_train, axis = 0)
elmo_test_new = np.concatenate(elmo_test, axis = 0)
```

我建议你将这些数组储存好，因为我们需要很长时间来得到它们的ELMo向量。我们可以将它们存为pickle文件：

```
# save elmo_train_new

pickle_out = open("elmo_train_03032019.pickle","wb")
pickle.dump(elmo_train_new, pickle_out)
pickle_out.close()

# save elmo_test_new

pickle_out = open("elmo_test_03032019.pickle","wb")
pickle.dump(elmo_test_new, pickle_out)
pickle_out.close()
```

然后用以下代码来将它们重新加载：

```
# load elmo_train_new

pickle_in = open("elmo_train_03032019.pickle", "rb")

elmo_train_new = pickle.load(pickle_in)

# load elmo_train_new

pickle_in = open("elmo_test_03032019.pickle", "rb")
elmo_test_new = pickle.load(pickle_in)
```

构建模型并评估

让我们用ELMo来构建NLP模型吧！

我们可以用训练集的ELMo向量来构建一个分类模型。然后，我们会用该模型在测试集上进行预测。但在做这些之前，我们需要将elmo_train_new分成训练集和验证集来检验我们的模型。

```
from sklearn.model_selection import train_test_split

xtrain, xvalid, ytrain, yvalid = train_test_split(elmo_train_new, train['label'], random_state=42, test_size=0.2)
```

由于我们的目标是设置基线分数，我们将用ELMo向量作为特征来构建一个简单的逻辑回归模型：

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
```

```
lreg = LogisticRegression()
lreg.fit(xtrain, ytrain)
```

到了预测的时间了！首先，在验证集上：

```
preds_valid = lreg.predict(xvalid)
```

我们用F1矩阵来评估我们的模型，因为这是竞赛的官方评估指标：

```
f1_score(yvalid, preds_valid)
```

Output: 0.789976

验证集上的F1分数很不错，接下来我们在测试集上进行预测：

```
# make predictions on test set
preds_test = lreg.predict(elmo_test_new)
```

准备将要上传到竞赛页面的提交文件：

```
# prepare submission dataframe
sub = pd.DataFrame({'id':test['id'], 'label':preds_test})

# write predictions to a CSV file
sub.to_csv("sub_lreg.csv", index=False)
```

公开排行榜显示我们的预测结果得到了0.875672分，可以说这个结果非常的好，因为我们只进行了相对基础的预处理过程，而且用了一个很简单的模型。可以预见如果我们用了更先进的技术将会得到更好的分数，大家可以自行尝试并将结果告诉我！

5. 我们还能用ELMo做什么?

我们刚刚见证了在文本识别中ELMo是多么高效，如果能搭配一个更复杂的模型它一定会有更出色的表现。ELMo的应用并不局限于文本分类，只要你需要将文本数据向量化都可以用它。

以下是几种可以使用ELMo进行处理的NLP问题：

- 机器翻译（Machine Translation）
- 语言模型（Language Modeling）
- 文本摘要（Text Summarization）
- 命名实体识别（Named Entity Recognition）
- 问答系统（Question-Answering Systems）

6. 结语

ELMo无疑是NLP的重大进步，并且将保持趋势。鉴于NLP研究的进展速度非常快，最近几个月还出现了其他新的最先进的词嵌入，如Google BERT和Falando's Flair。可以说令NLP从业者激动的时代到来了！

我强烈建议你在其他数据集上使用ELMo，并亲自体验性能提升的过程。如果你有任何问题或希望与我和社区分享你的经验，请在下面的评论板块中进行。如果你刚在NLP领域起步，你还应该查看以下NLP相关资源：

- Natural Language Processing (NLP) course

(https://courses.analyticsvidhya.com/courses/natural-language-processing-nlp?utm_medium=ELMoNLParticle&utm_source=blog)

- Certified Program: Natural Language Processing (NLP) for Beginners

(https://courses.analyticsvidhya.com/bundles/nlp-combo?utm_medium=ELMoNLParticle&utm_source=blog)

你也可以通过Analytics Vidhya的安卓软件来阅读本文。

原文标题：A Step-by-Step NLP Guide to Learn ELMo for Extracting Features from Text
原文链接：https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/

译者简介：韩国钧，本科毕业于中国人民大学信息学院，目前墨尔本大学数据科学专业研二在读。对挖掘数据背后的信息感兴趣，希望认识志同道合的朋友，一起深入学习并不断提升自己。
转自：数据派THU 公众号；

END