fastText、TextCNN、TextRNN...这套NLP文本分类深度学习方法库供你选择

数据派THU 2017-08-02

> 文经公众号「机器人圈」授权转载（微信号：ROBO_AI）

本文长度为**4473字**，建议阅读**10分钟**
本文为你介绍一套NLP文本分类深度学习方法库及其12个模型。

这个库的目的是探索用深度学习进行NLP文本分类的方法。

它具有文本分类的各种基准模型，还支持多标签分类，其中多标签与句子或文档相关联。

虽然这些模型很多都很简单，可能不会让你在这项文本分类任务中游刃有余，但是这些模型中的其中一些是非常经典的，因此它们可以说是非常适合作为基准模型的。

每个模型在模型类型下都有一个测试函数。

我们还探讨了用两个seq2seq模型（带有注意的seq2seq模型，以及transformer：attention is all you need）进行文本分类。同时，这两个模型也可以用于生成序列和其他任务。如果你的任务是多标签分类，那么你就可以将问题转化为序列生成。

我们实现了一个记忆网络：循环实体网络（recurrent entity network）：追踪世界的状态。它用键值对块（blocks of key-value pairs）作为记忆，并行运行，从而获得新的状态。它可以用于使用上下文（或历史）来回答建模问题。例如，你可以让模型读取一些句子（作为文本），并提出一个问题（作为查询），然后请求模型预测答案；如果你像查询一样向其提供故事，那么它就可以进行分类任务。

如果你想了解更多关于文本分类，或这些模型可以应用任务的数据集详细信息，可以点击链接进行查询，我们选择了一个：
https://biendata.com/competition/zhihu/

模型：

1.fastText

2.TextCNN

3.TextRNN

4.RCNN

5.分层注意网络（Hierarchical Attention Network）

6.具有注意的seq2seq模型（seq2seq with attention）

7.Transformer("Attend Is All You Need")

8.动态记忆网络（Dynamic Memory Network）

9.实体网络：追踪世界的状态

其他模型：

1.BiLstm Text Relation；

2.Two CNN Text Relation；

3.BiLstm Text Relation Two RNN

性能：

Model	fastText	TextCNN	TextRNN	RCNN	HierAtteNetwork	Seq2seqWithAttention	EntityNetwo
Score	0.362	0.405	0.358	0.395	0.398	0.322	0.400
Training	10 minutes	2 hours	10 hours	2 hours	2 hours	3 hours	3 hour

（多标签预测任务，要求预测能够达到前5，300万训练数据，满分：0.5）

注意："HierAtteNetwork"是指Hierarchical Attention Network

用途：

- 模型在xxx_model.py中
- 运行python xxx_train.py来训练模型
- 运行python xxx_predict.py进行推理（测试）。

每个模型在模型下都有一个测试方法。你可以先运行测试方法来检查模型是否能正常工作。

环境：

python 2.7+tensorflow 1.1

（tensorflow 1.2也是可以应用的；大多数模型也应该在其他tensorflow版本中正常应用，因为我们使用非常少的特征来将其结合到某些版本中；如果你使用的是python 3.5，只要更改print / try catch函数的话，它也会运行得很好。）

注：一些util函数是在data_util.py中的；典型输入如："x1 x2 x3 x4 x5 label 323434"，其中"x1，x2"是单词，"323434"是标签；它具有一个将预训练的单词加载和分配嵌入到模型的函数，其中单词嵌入在word2vec或fastText中进行预先训练。

模型细节：

1.快速文本（fast Text）

《用于高效文本分类的技巧》（Bag of Tricks for Efficient Text Classification）论文的实现（https://arxiv.org/abs/1607.01759）

- 使用bi-gram 或者tri-gram。
- 使用NCE损失，加速我们的softmax计算（不使用原始论文中的层次softmax）结果：性能与原始论文中的一样好，速度也非常快。

查看：p5_fastTextB_model.py

2.文本卷积神经网络（Text CNN）

《卷积神经网络进行句子分类》（Convolutional Neural Networks for Sentence Classification）论文的实现。（http://www.aclweb.org/anthology/D14-1181）
结构：降维---> conv ---> 最大池化 --->完全连接层--------> softmax
查看：p7_Text CNN_model.py
为了能够使用TextCNN获得非常好的结果，你还需要仔细阅读此论文"用于句子分类的卷积神经网络灵敏度分析（和从业者指南）"（A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification），它可以帮助你了解一些影响性能的相关见解。（https://arxiv.org/abs/1510.03820）当然，你还需要根据具体任务来更改某些设置。

3.文本循环神经网络（Text RNN）

结构：降维--->双向lstm ---> concat输出--->平均 -----> softmax
查看：p8_Text RNN_model.py

4.双向长短期记忆网络文本关系（BiLstm Text Relation）

结构：结构与Text RNN相同。但输入是被特别设计的。例如：输入"这台电脑多少钱？笔记本电脑的股票价格（how much is the computer? EOS price of laptop）"。"EOS"是一个特殊的标记，将问题1和问题2分开。
查看：p9_BiLstm Text Relation_model.py

5.两个卷积神经网络文本关系（two CNN Text Relation）

结构：首先用两个不同的卷积来提取两个句子的特征，然后连接两个功能，使用线性变换层将投影输出到目标标签上，然后使用softmax。
查看：p9_two CNN Text Relation_model.py

6.双长短期记忆文本关系双循环神经网络（BiLstm Text Relation Two RNN）

结构：一个句子的一个双向lstm（得到输出1），另一个句子的另一个双向lstm（得到输出2）。那么：softmax（输出1 M输出2）
查看：p9_BiLstm Text Relation Two RNN_model.py
有关更多详细信息，你可以访问：《Deep Learning for Chatbots》的第2部分—在Tensorflow中实现一个基于检索的模型（Implementing a Retrieval-Based Model in Tensorflow）

7.循环卷积神经网络（RCNN）

用于文本分类的循环卷积神经网络。

《用于文本分类的循环卷积神经网络》（Recurrent Convolutional Neural Network for Text Classification）论文的实现。（https://scholar.google.com.hk/scholar?q=Recurrent+Convolutional+Neural+Networks+for+Text+Classification&hl=zh-CN&as_sdt=0&as_vis=1&oi=scholart&sa=X&ved=0ahUKEwjpx82cvqTUAhWHspQKHUbDBDYQgQMIITAA）

结构：

- 循环结构（卷积层）
- 最大池化
- 完全连接层+ softmax

它用左侧文本和右侧文本学习句子或文档中的每个单词的表示：
表示当前单词 = [left_side_context_vector，current_word_embedding，right_side_context_vecotor]。
对于左侧文本，它使用一个循环结构，前一个单词的非线性转换和左侧上一个文本；类似于右侧文本。
查看：p71_TextRCNN_model.py

8. 分层注意网络（Hierarchical Attention Network）

《用于文档分类的分层注意网络》（Hierarchical Attention Networks for Document Classification）论文的实现。（https://www.cs.cmu.edu/~diyiy/docs/naacl16.pdf）

结构：

- 降维
- 词编码器：词级双向GRU，以获得丰富的词汇表征
- 次注意：词级注意在句子中获取重要信息
- 句子编码器：句子级双向GRU，以获得丰富的句子表征
- 句子注意：句级注意以获得句子中的重点句子
- FC + Softmax

数据输入：

一般来说，这个模型的输入应该是几个句子，而不是一个句子。形式是：[None，sentence_lenght]。其中None意味着batch_size。

在我的训练数据中，对于每个样本来说，我有四个部分。每个部分具有相同的长度。我将四个部分形成一个单一的句子。该模型将句子分为四部分，形成一个形状为：[None，num_sentence，sentence_length]的张量。其中num_sentence是句子的个数（在我的设置中，其值等于4）。

查看：p1_HierarchicalAttention_model.py

9. 具有注意的Seq2seq模型

具有注意的Seq2seq模型的实现是通过《共同学习排列和翻译的神经机器翻译》来实现的。（https://github.com/brightmart/text_classification/blob/master/README.md）

结构：

- 降维
- bi-GRU也从源语句（向前和向后）获取丰富的表示。
- 具有注意的解码器。

数据输入：

使用三种输入中的两种：

- 编码器输入，这是一个句子;
- 解码器输入，是固定长度的标签列表;
- 目标标签，它也是一个标签列表。

例如，标签是：“L1 L2 L3 L4”，则解码器输入将为：[_ GO,L1,L2,L2,L3,_PAD]；目标标签为：[L1,L2,L3,L3,_END,_PAD]。长度固定为6，任何超出标签将被截断，如果标签不足以填

补，将填充完整。

注意机制：

- 传输编码器输入列表和解码器的隐藏状态
- 计算每个编码器输入隐藏状态的相似度，以获得每个编码器输入的可能性分布。
- 基于可能性分布的编码器输入的加权和。

通过RNN Cell使用这个权重和解码器输入以获得新的隐藏状态。

Vanilla E编码解码工作原理：
在解码器中，源语句将使用RNN作为固定大小向量（“思想向量”）进行编码：
当训练时，将使用另一个RNN尝试通过使用这个“思想向量”作为初始化状态获取一个单词，并从每个时间戳的解码器输入获取输入。解码器从特殊指令“_GO”开始。在执行一步之后，新的隐藏状态将与新输入一起获得，我们可以继续此过程，直到我们达到特殊指令“_END”。我们可以通过计算对数和目标标签的交叉熵损失来计算损失。logits是通过隐藏状态的投影层（对于解码器步骤的输出，在GRU中，我们可以仅使用来自解码器的隐藏状态作为输出）。
当测试时，没有标签。所以我们应该提供我们从以前的时间戳获得的输出，并继续进程直到我们到达“_END”指令。

注意事项：
这里我使用两种词汇。 一个是由编码器使用的单词; 另一个是用于解码器的标签。
对于词汇表，插入三个特殊指令：“_ GO”，“_ END”，“_ PAD”；“_UNK”不被使用，因为所有标签都是预先定义的。

10. Transformer（“Attention Is All You Need”）

状态：完成主要部分，能够在任务中产生序列的相反顺序。你可以通过在模型中运行测试功能来检查它。然而，我还没有在实际任务中获得有用的结果。我们在模型中也使用并行的style.layer规范化、残余连接和掩码。
对于每个构建块，我们在下面的每个文件中包含测试函数，我们已经成功测试了每个小块。

带注意的序列到序列是解决序列生成问题的典型模型，如翻译、对话系统。大多数时候，它使用RNN完成这些任务。直到最近，人们也应用卷积神经网络进行序列顺序问题。但是，Transformer，它仅仅依靠注意机制执行这些任务，是快速的、实现新的最先进的结果。

它还有两个主要部分：编码器和解码器。看以下内容：

- 编码器：

共6层，每个层都有两个子层。第一是多向自我注意结构；第二个是位置的全连接前馈网络。
对于每个子层使用LayerNorm（x + Sublayer（x）），维度= 512。

- 解码器：
  - 解码器由N = 6个相同层的堆叠组成。
  - 除了每个编码器层中的两个子层之外，解码器插入第三子层，其在编码器堆栈的输出上执行多向注意。
  - 与编码器类似，我们采用围绕每个子层的残余连接，然后进行层归一化。我们还修改解码器堆栈中的自我注意子层，以防止位置参与到后续位置。这种掩蔽与输出嵌入偏移一个位置的事实相结合确保了位置i的预测只能取决于位于小于i的位置的已知输出。

主要从这个模型中脱颖而出：

- 多向自我注意：使用自我注意，线性变换多次获取关键值的投影，然后开始注意机制
- 一些提高性能的技巧（剩余连接、位置编码、前馈、标签平滑、掩码以忽略我们想忽略的事情）。

有关模型的详细信息，请查看：a2_transformer.py

11.循环实体网络（Recurrent Entity Network）

输入：

- 故事：它是多句话，作为上下文。
- 问题：一个句子，这是一个问题。
- 回答：一个单一的标签。

型号结构：

- 输入编码：

使用一个词来编码故事（上下文）和查询（问题）；通过使用位置掩码将位置考虑在内。
通过使用双向rnn编码故事和查询，性能从0.392提高到0.398，增长了1.5％。

- 动态记忆：
  - 通过使用键的“相似性”，输入故事的值来计算门控。
  - 通过转换每个键，值和输入来获取候选隐藏状态。
  - 组合门和候选隐藏状态来更新当前的隐藏状态。
- 输出（使用注意机制）：
  - 通过计算查询和隐藏状态的“相似性”来获得可能性分布。
  - 使用可能性分布获得隐藏状态的加权和。

- 查询和隐藏状态的非线性变换获得预测标签。

这个模型的关键点：

- 使用彼此独立的键和值块，可以并行运行。
- 上下文和问题一起建模。使用记忆来追踪世界的状态，并使用隐性状态和问题（查询）的非线性变换进行预测。
- 简单的型号也可以实现非常好的性能。简单的编码作为词的使用包。

有关模型的详细信息，请查看：a3_entity_network.py
在这个模型下，它包含一个测试函数，它要求这个模型计算故事（上下文）和查询（问题）的数字，但故事的权重小于查询。

12.动态记忆网络

模块：Outlook

- 输入模块：将原始文本编码为向量表示。
- 问题模块：将问题编码为向量表示。
- 独特的记忆模块：通过输入，通过注意机制选择哪些部分输入、将问题和以前的记忆考虑在内=====>它将产生"记忆"向量。
- 答案模块：从最终的记忆向量生成答案。

详情：

- 输入模块：

一个句子：使用gru获取隐藏状态b.list的句子：使用gru获取每个句子的隐藏状态。例如 [隐藏状态1,隐藏状态2,隐藏状态...,隐藏状态n]。

- 问题模块：使用gru获取隐藏状态。
- 记忆模块：

使用注意机制和循环网络来更新其记忆。

- 需要多集===>传递推理。

```
e.g. ask where is the football? it will attend to sentence of "john put down the football"), then in
```

- 注意机制：

```
two-layer feed forward nueral network.input is candidate fact c,previous memory m and question q. fea
```

- 记忆更新机制：h = f（c,h_previous,g）。 最后一个隐藏状态是应答模块的输入。

- 答案模块

要做的事情：

- 文本分类的字符级卷积网络
- 文本分类的卷积神经网络：浅词级与深字符级
- 文本分类的深度卷积网络
- 半监督文本分类的对抗训练方法

参考：

1.《用于高效文本分类的技巧》Bag of Tricks for Efficient Text Classification

2.《语音分类的卷积神经网络》Convolutional Neural Networks for Sentence Classification

3.《卷积神经网络对句子分类的敏感性分析（和使用指南）》A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification

4.《聊天机器人中的深度学习》，第2部分—在Tensorflow中实现基于检索的模型Deep Learning for Chatbots, Part 2 – Implementing a Retrieval-Based Model in Tensorflow（www.wildml.com）

5.《文本分类的复杂卷积神经网络》Recurrent Convolutional Neural Network for Text Classification

6.《文档分类的分层注意网络》Hierarchical Attention Networks for Document Classification

7.《共同学习对齐排列和翻译的神经机器翻译》Neural Machine Translation by Jointly Learning to Align and Translate

8. Attention Is All You Need

9.《问我任何事情：自然语言处理的动态记忆网络》Ask Me Anything:Dynamic Memory Networks for Natural Language Processing

10.《用循环实体网络跟踪世界的状况》Tracking the state of world with recurrent entity networks

编辑：王璇

校对：王红玉