GitHub 热文！图解 BERT 和 ELMo 模型

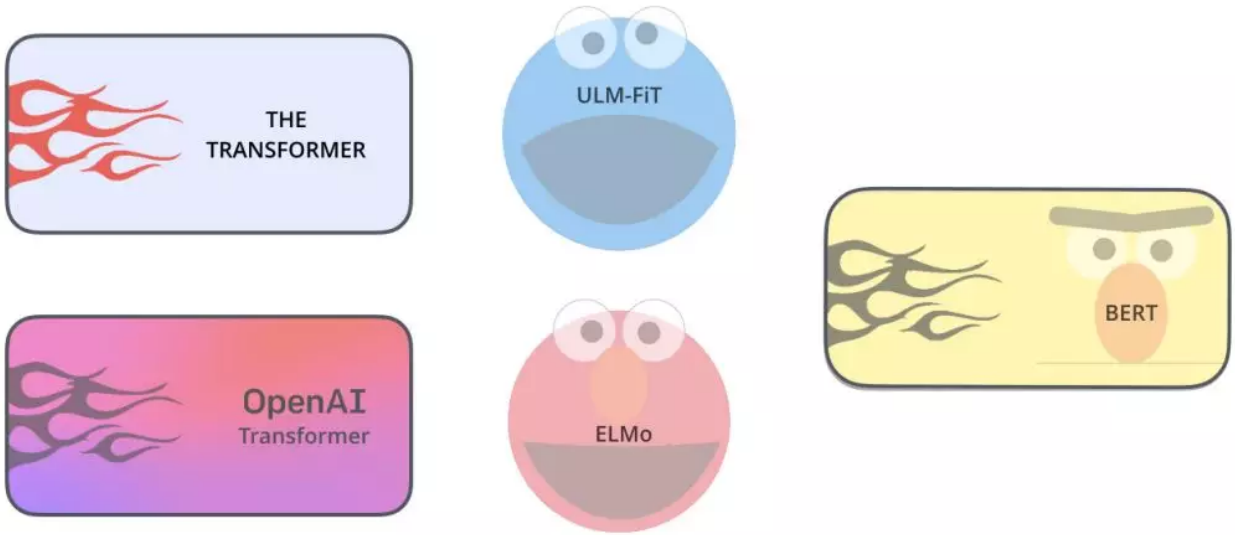
算法爱好者 2020-10-19

（给算法爱好者加星标，修炼编程内功）

> 编译：新智元 / 肖琴，英文：Jay Alammar

2018年已经成为自然语言处理机器学习模型的转折点。我们对如何以最能捕捉潜在意义和关系的方式、最准确地表示单词和句子的理解正在迅速发展。

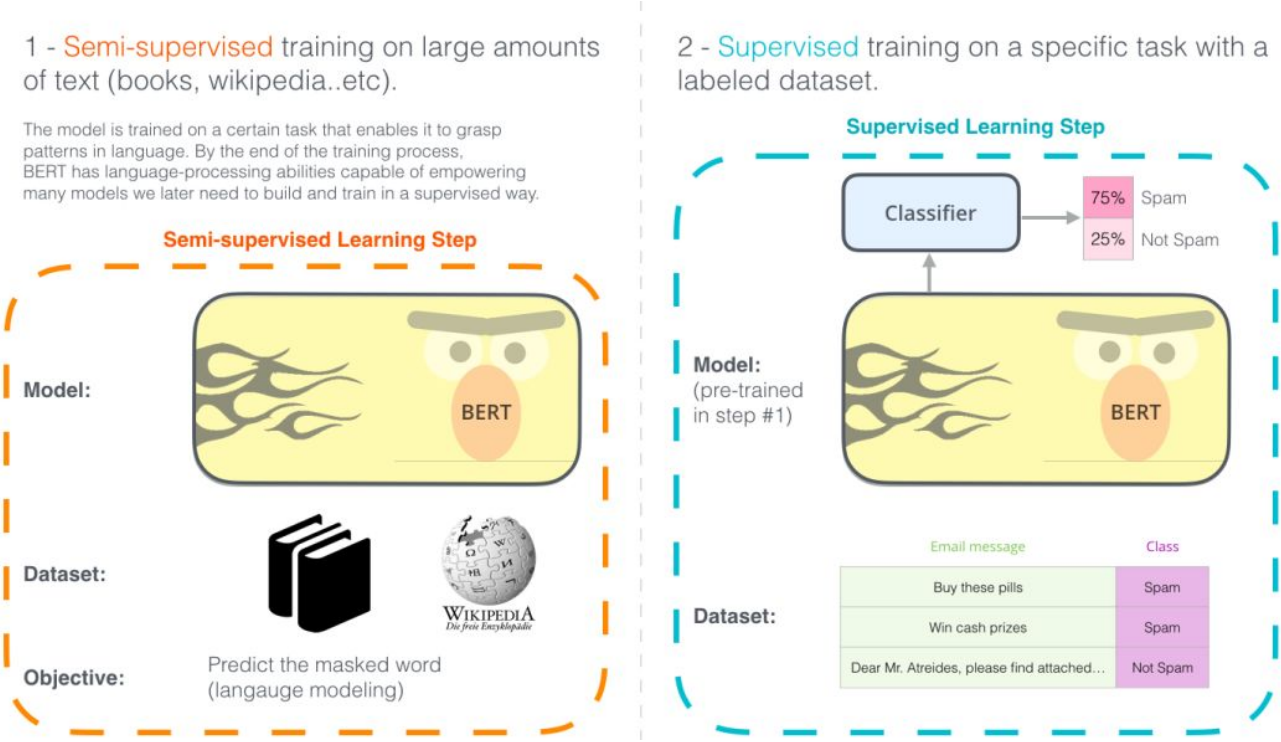
此外，NLP社区开发了一些非常强大的组件，你可以免费下载并在自己的模型和pipeline中使用。



ULM-FiT跟甜饼怪没有任何关系，但我想不出其它的了...

最新的一个里程碑是**BERT**的发布，这一事件被描述为NLP新时代的开始。BERT是一个NLP模型，在几个语言处理任务中打破了记录。在描述模型的论文发布后不久，该团队还公开了模型的源代码，并提供了已经在大量数据集上预训练过的下载版本。

这是一个重大的进展，因为任何需要构建语言处理模型的人都可以将这个强大的预训练模型作为现成的组件使用，从而节省了从头开始训练模型所需的时间、精力、知识和资源。



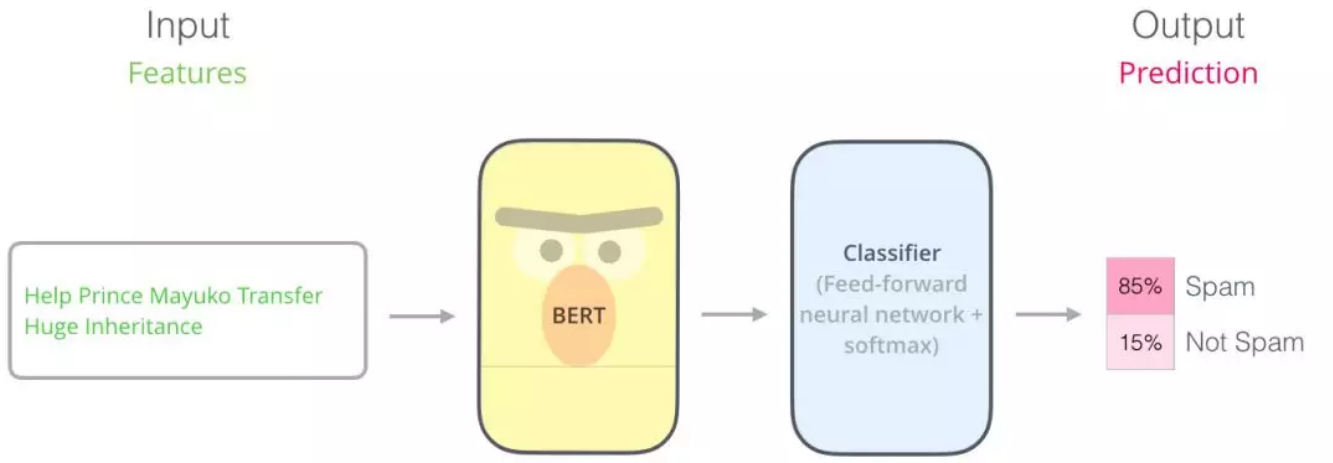
图示的两个步骤显示了BERT是如何运作的。你可以下载步骤1中预训练的模型(在未经注释的数据上训练)，然后只需在步骤2中对其进行微调。

BERT建立在最近NLP领域涌现的许多聪明方法之上——包括但不限于**半监督序列学习**(作者是Andrew Dai 和 Quoc Le)、**ELMo** (作者是Matthew Peters和来自AI2和UW CSE的研究人员)、**ULMFiT** (作者是fast.ai创始人Jeremy Howard和Sebastian Ruder)，**OpenAI transformer** (作者是OpenAI研究员Radford、Narasimhan、Salimans和Sutskever)，以及**Transformer** (作者是Vaswani et al .)。

要正确理解BERT是什么，我们需要了解一些概念。让我们先看看如何使用BERT，然后再看模型本身涉及的概念。

例子：句子分类

最直接的使用BERT的方法就是使用它来对单个文本进行分类。这个模型看起来是这样的：



要训练一个这样的模型，主要需要**训练分类器**，在训练阶段对BERT模型的更改非常小。这种训练过程称为**微调 (Fine-Tuning)**，并且具有半监督序列学习(Semi-supervised Sequence Learning)和ULMFiT的根源。

具体来说，由于我们讨论的是分类器，这属于机器学习的监督学习范畴。这意味着我们需要一个标记数据集来训练模型。比如说，对于一个垃圾邮件分类器，标记数据集是一个电子邮件列表及其标签(将每封电子邮件标记为“垃圾邮件”或“非垃圾邮件”)。

Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached…	Not Spam

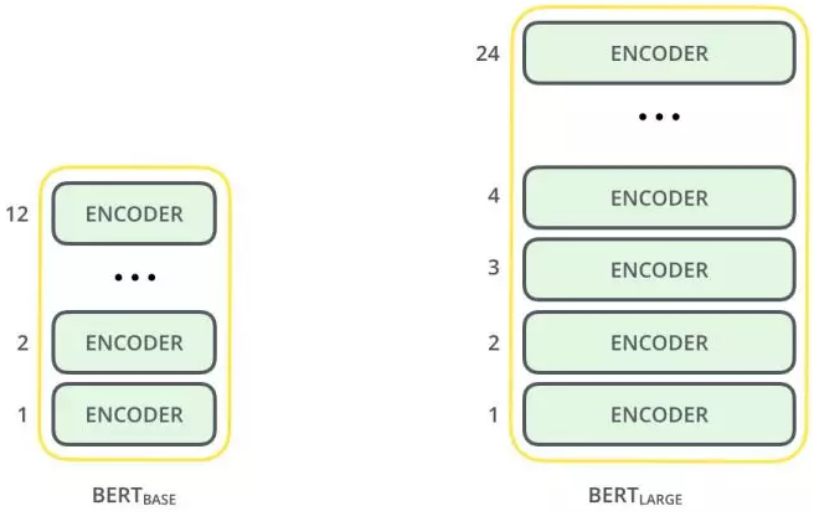
模型架构

现在，你已经有了一个如何使用BERT的示例用例，接下来让我们进一步了解它是如何工作的。

论文中提供了两种尺寸的BERT模型：

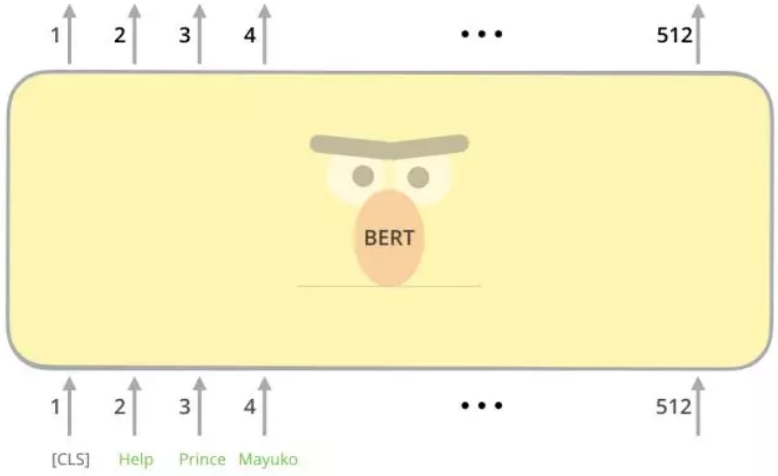
- BERT BASE - 大小与OpenAI Transformer相当
- BERT LARGE - 一个非常庞大的模型，实现了最先进的结果

BERT基本上是一个训练好的Transformer Encoder堆栈。Transformer模型是BERT的一个基本概念，我们将在下文中讨论。



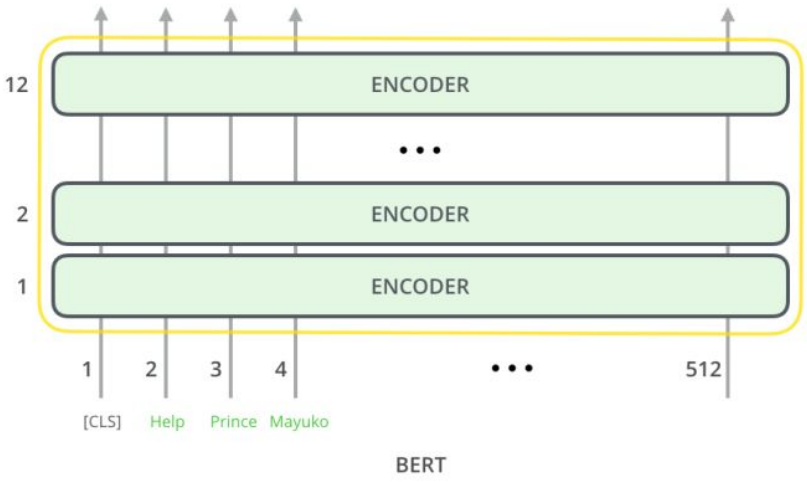
这两种BERT模型都有大量的**编码器层**(论文中称之为Transformer Blocks)—— Base 版本有12层，Large版本有24层。它们也比初始论文里的Transformer的默认配置(6个编码器层，512个隐藏单元，8个attention heads)有更大的前馈网络(分别为768个和1024个隐藏单元)， attention heads(分别为12个和16个)。

模型输入



第一个输入token是一个特殊的[CLS]token，这里的CLS代表分类。

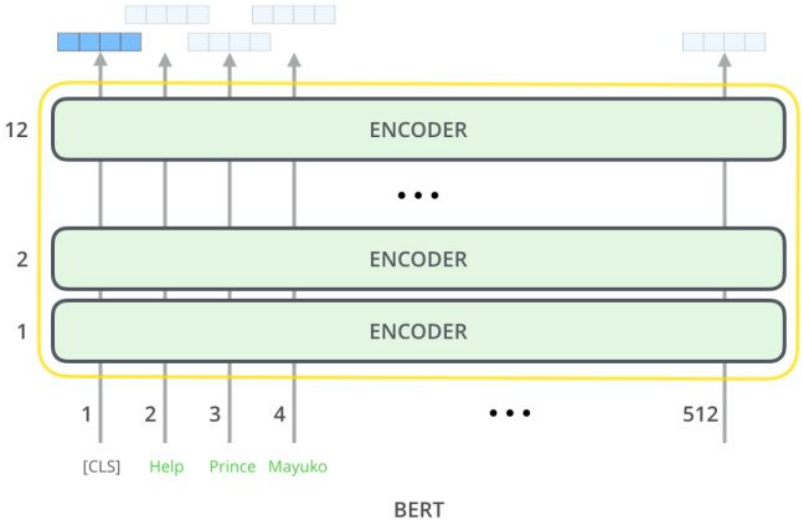
就像transformer的普通编码器一样，BERT以一串单词作为输入。每一层应用self-attention，并通过前馈网络传递其结果，然后将结果传递给下一个编码器。



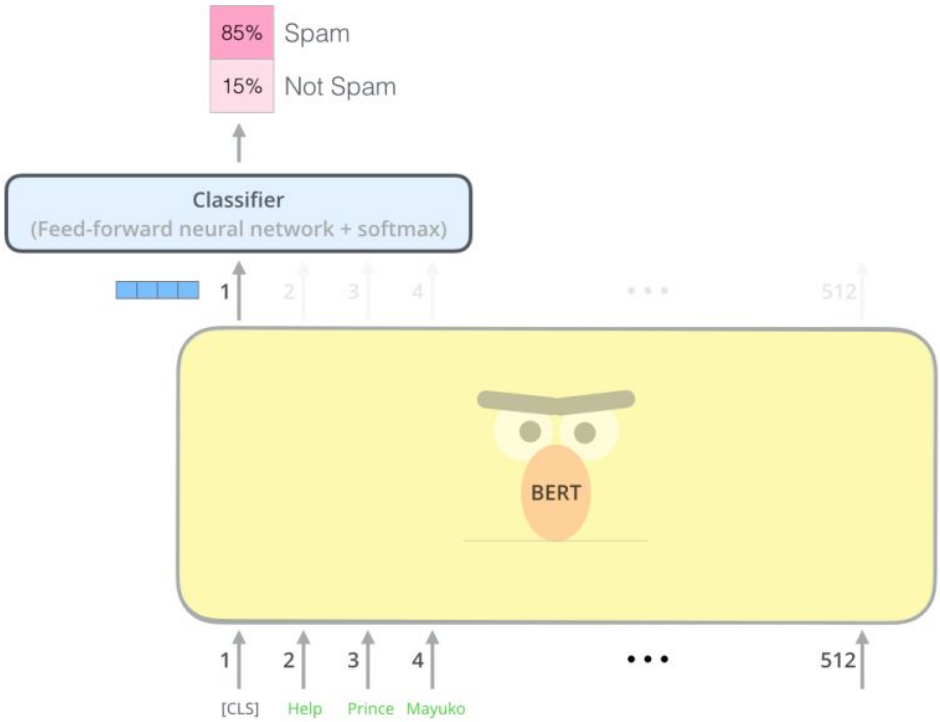
在架构方面，到目前为止，这与Transformer完全相同(除了大小之外，不过大小是我们可以设置的配置)。在输出端，我们才开始看到两者的区别。

模型输出

每个位置输出大小为hidden_size的向量(BERT Base中为768)。对于上面看到的句子分类示例，我们只关注第一个位置的输出(我们将那个特殊的[CLS]标记传递给它)。



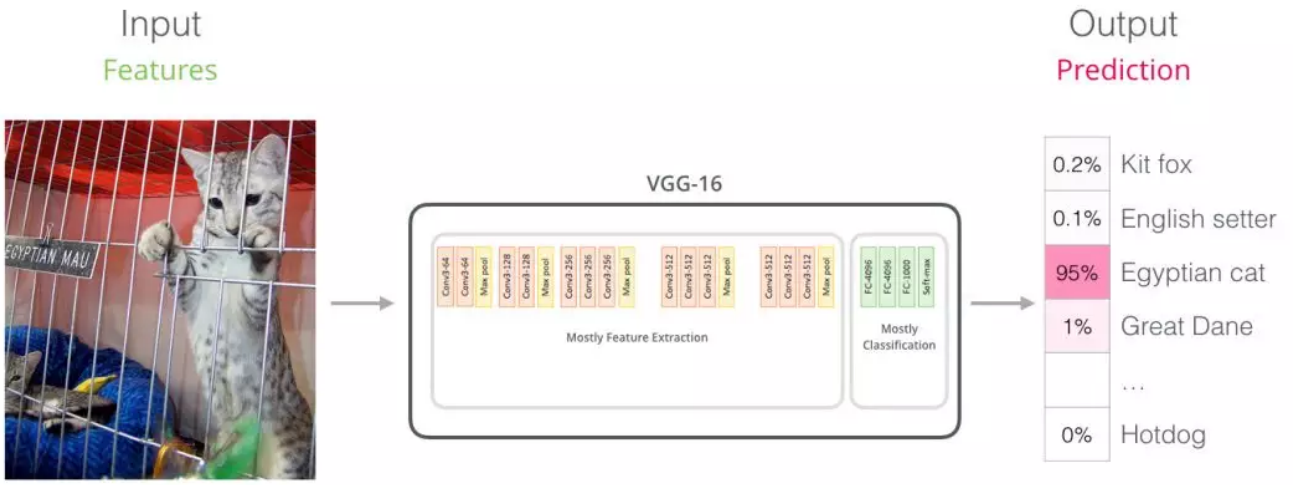
这个向量可以作为我们选择的分类器的输入。论文中利用单层神经网络作为分类器，取得了很好的分类效果。



如果你有更多的标签(例如，如果是电子邮件，你可以将邮件标记为“垃圾邮件”、“非垃圾邮件”、“社交”和“促销”)，只需调整分类器网络，使其有更多的输出神经元，然后通过softmax。

与卷积网络的相似之处

对于具有计算机视觉背景的人来说，这种向量传递的方式很容易让人联想到VGGNet之类的网络的卷积部分与网络末端完全连接的分类部分之间的事情。



嵌入的新时代

这些新进展带来了词汇编码方式的新变化。词汇嵌入一直是领先的NLP模型处理语言的主要能力。Word2Vec、Glove等方法已广泛应用于此类任务。让我们先回顾一下如何使用它们。

词汇嵌入的回顾

对于要由机器学习模型处理的单词，它们需要以某种数字形式表示，以便模型可以在其计算中使用。Word2Vec表明我们可以用一个向量(一个数字列表)以捕捉语义或意义关系(如判断单词的近义、反义关系)、以及语法或语法关系(例如, “had”和“has” 、“was” and “is”有同样的语法关系)的方式恰当地表示单词。

研究人员很快发现，使用经过大量文本数据进行预训练的嵌入(embeddings)是一个好主意，而不是与小数据集的模型一起训练。因此，通过使用Word2Vec或GloVe进行预训练，可以下载单词列表及其嵌入。如下图是单词“stick”的GloVe 嵌入示例(嵌入向量大小为200)

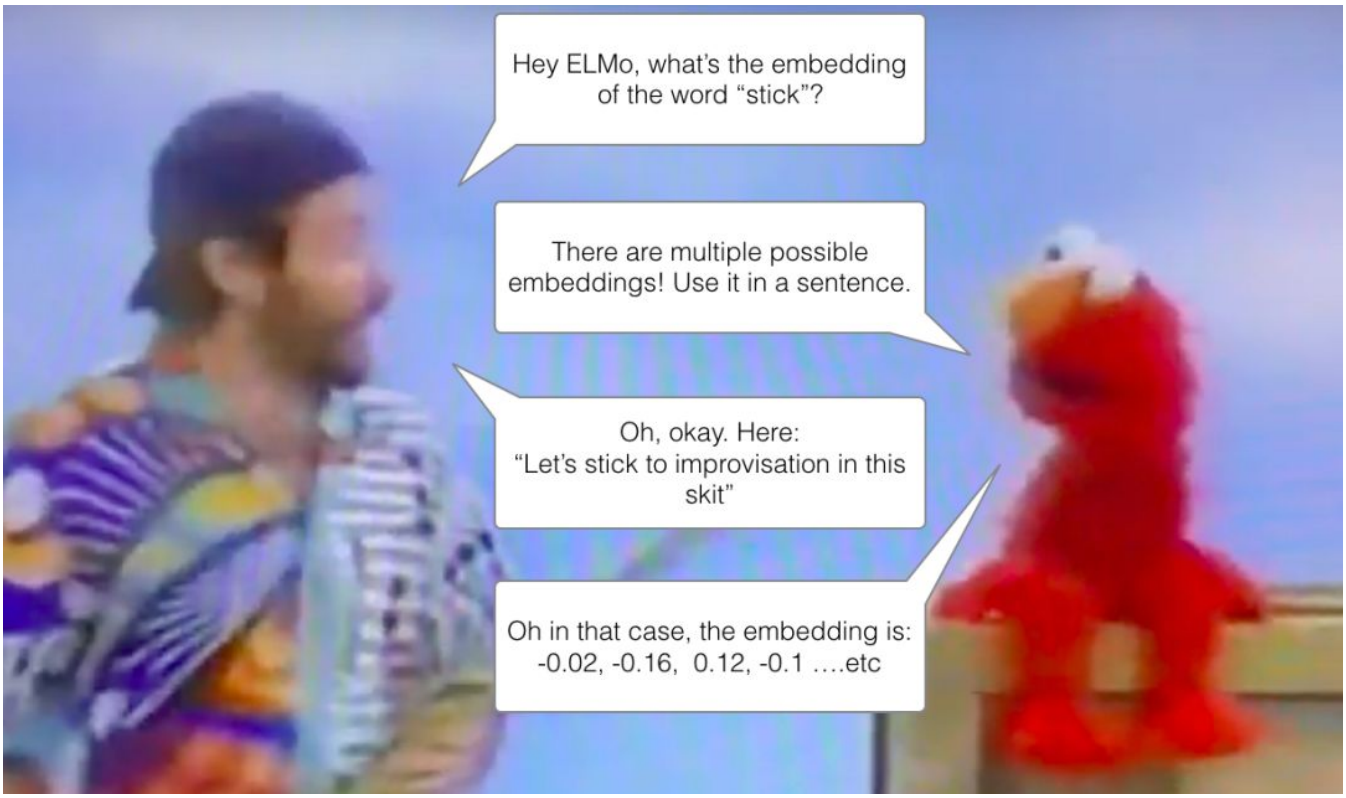
-0.34	-0.84	0.20	-0.26	-0.12	0.23	1.04	-0.16	0.31	0.06	0.30	0.33	-1.17	-0.30	0.03	0.09	0.35	-0.28	-0

单词“stick”的GloVe嵌入

因为这些向量很大，并且数字很多，所以本文后面用下面这个基本图形来表示向量：

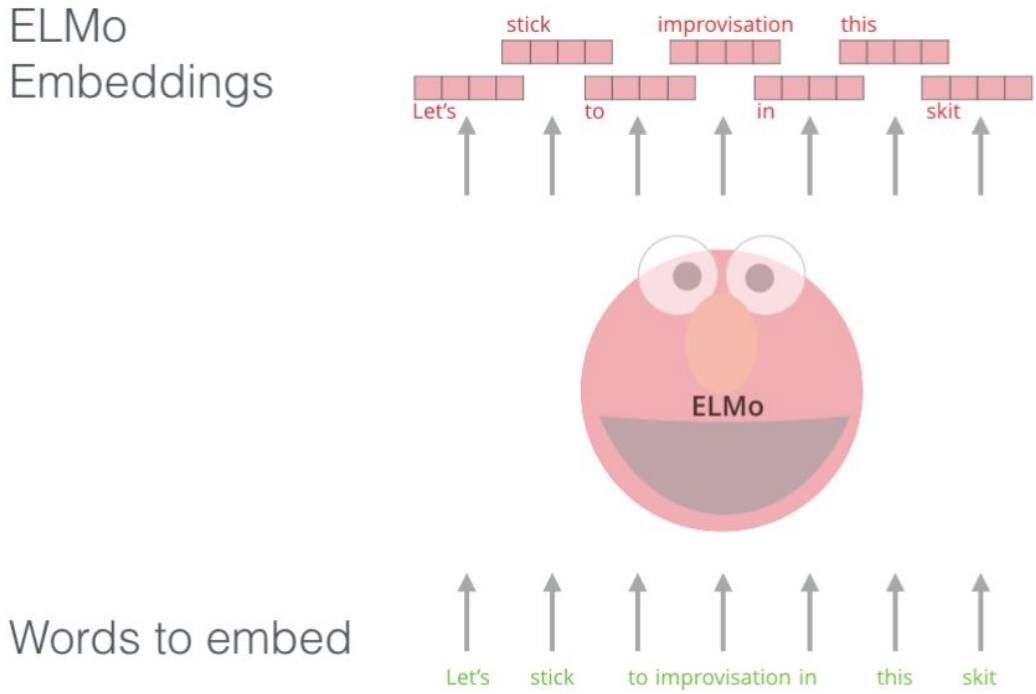
ELMo: 上下文很重要

如果我们使用GloVe表示，那么不管上下文是什么，“stick”这个词都会由这个向量表示。很多研究人员就发现不对劲了。“stick””有多种含义，取决于它的上下文是什么。那么，为什么不根据它的上下文给它一个嵌入呢——既要捕捉该上下文中的单词含义，又要捕捉其他上下文信息？因此，**语境化的词嵌入(contextualized word-embeddings)**就出现了。



语境化词嵌入可以根据单词在句子的上下文中表示的不同含义，给它们不同的表征

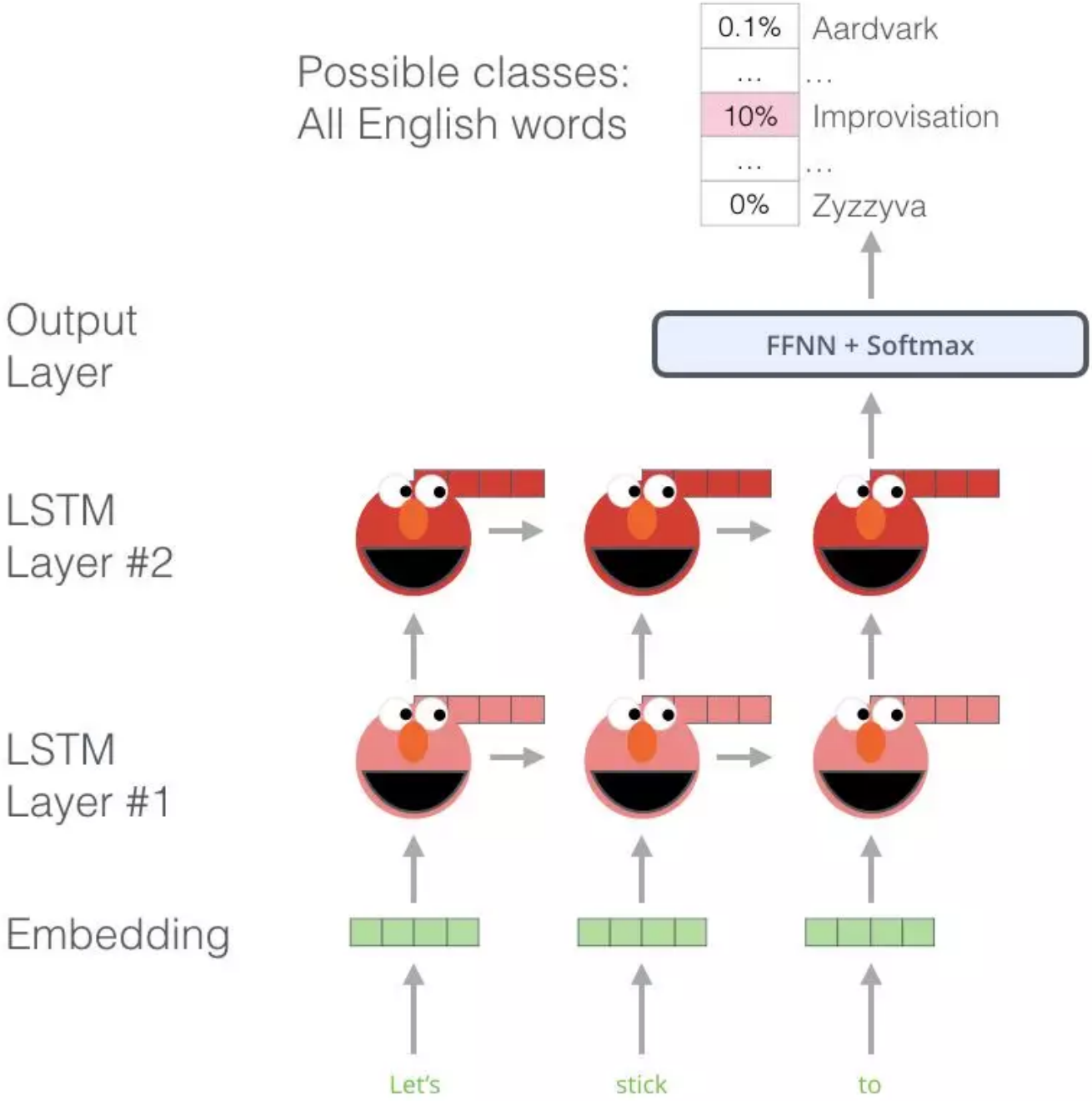
ELMo不是对每个单词使用固定的嵌入，而是在为每个单词分配嵌入之前查看整个句子。它使用针对特定任务的双向LSTM来创建嵌入。



ELMo为NLP中的预训练提供了重要的一步。ELMo LSTM在大型数据集上进行训练，然后我们可以将其用作所处理语言的其他模型中的组件使用。

ELMo的秘诀是什么?

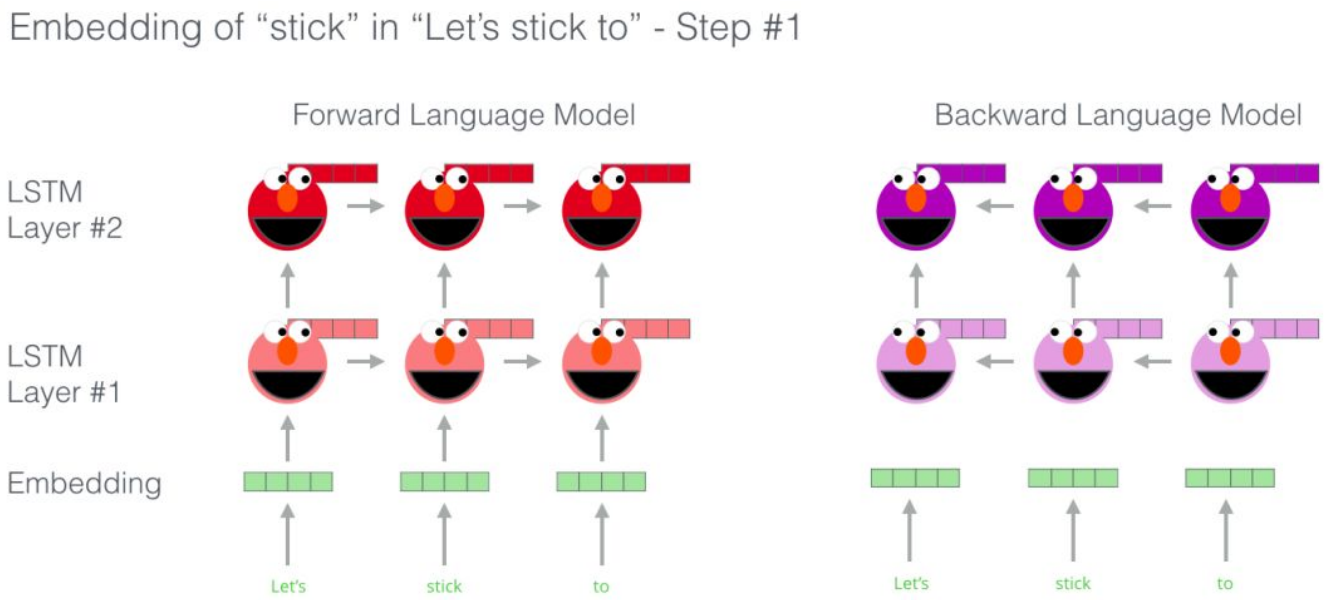
ELMo通过训练预测单词序列中的下一个单词来获得语言理解能力——这项任务被称为语言建模。这很方便，因为我们有大量的文本数据，这样的模型可以从这些数据中学习，不需要标签。



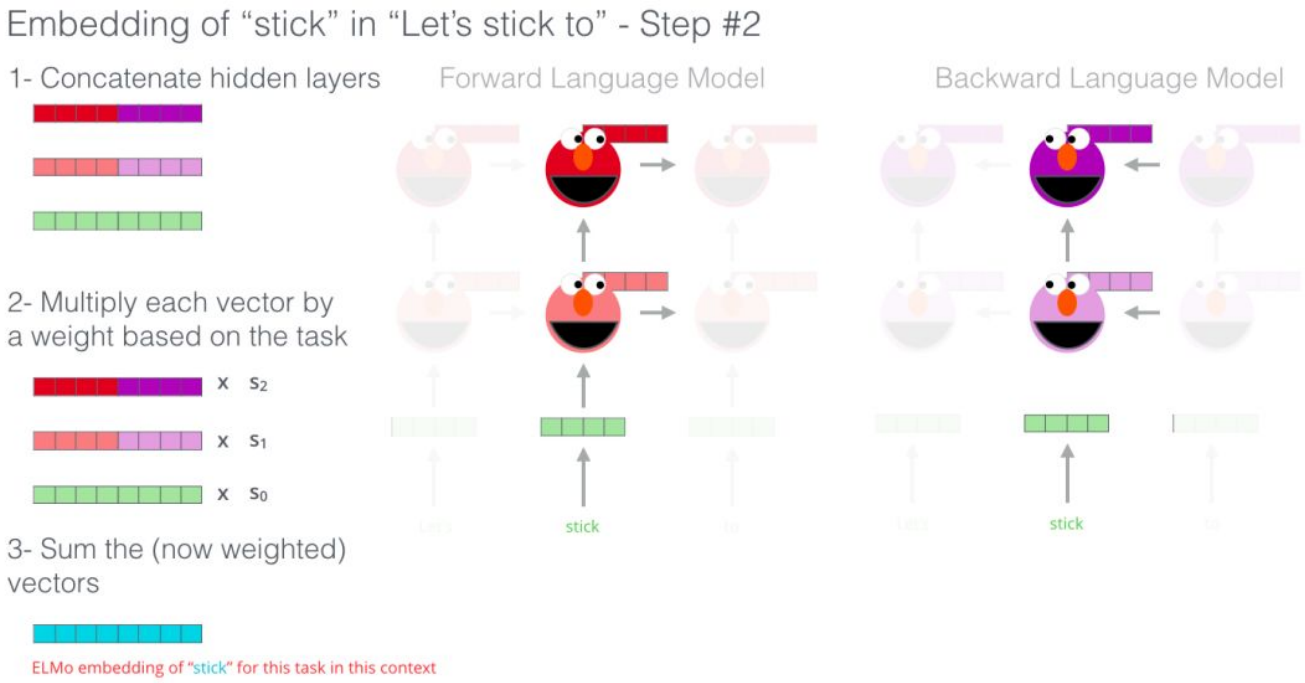
ELMo预训练的一个步骤

我们可以看到每个展开的LSTM步骤的隐藏状态从ELMo的头部后面突出来。这些在预训练结束后的嵌入过程中会派上用场。

ELMo实际上更进一步，训练了**双向LSTM**——这样它的语言模型不仅考虑下一个单词，而且考虑前一个单词。



ELMo通过将隐藏状态(和初始嵌入)以某种方式组合在一起(连接后加权求和)，提出语境化词嵌入。



ULM-FiT：NLP中的迁移学习

ULM-FiT引入了一些方法来有效地利用模型在预训练期间学到的知识——不仅是嵌入，也不仅是语境化嵌入。ULM-FiT提出了一个语言模型和一个流程(process)，以便针对各种任务有效地优化该语言模型。

NLP终于找到了一种方法，可以像计算机视觉那样进行迁移学习。

Transformer：超越LSTM

Transformer的论文和代码的发布，以及它在机器翻译等任务上取得的成果，开始使一些业内人士认为Transformers是LSTM的替代品。而且，Transformer在处理长期以来性方便比LSTM更好。

Transformer的编码器-解码器结构使其非常适合于机器翻译。但是如何使用它来进行句子分类呢？如何使用它来预训练可以针对其他任务进行微调的语言模型(在NLP领域，使用预训练模型或组件的监督学习任务被称为下游任务)。

OpenAI Transformer：为语言建模预训练Transformer解码器

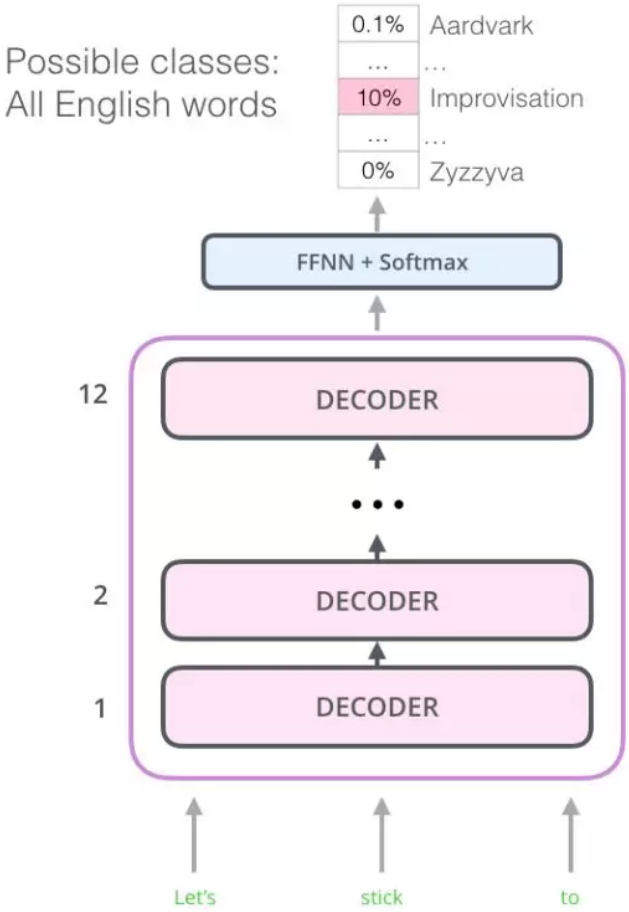
事实证明，我们不需要一个完整的Transformer来采用迁移学习，也不需要为NLP任务采用一个可微调的语言模型。我们只需要Transformer的解码器。解码器是一个很好的选择，因为它是语言建模(预测下一个单词)的首选，因为它是为屏蔽未来的token而构建的——在逐字生成翻译时，这是一个有用的特性。

OpenAI Transformer由Transformer的解码器堆栈组成

模型堆叠了12个解码器层。由于在这种设置中没有编码器，这些解码器层将不会有普通transformer解码器层所具有的编码器-解码器注意力子层。但是，它仍具有自注意层。

通过这个结构，我们可以继续在相同的语言建模任务上训练模型：使用大量(未标记的)数据集预测下一个单词。只是，我们可以把足足7000本书的文本扔给它，让它学习！书籍非常适合这类任

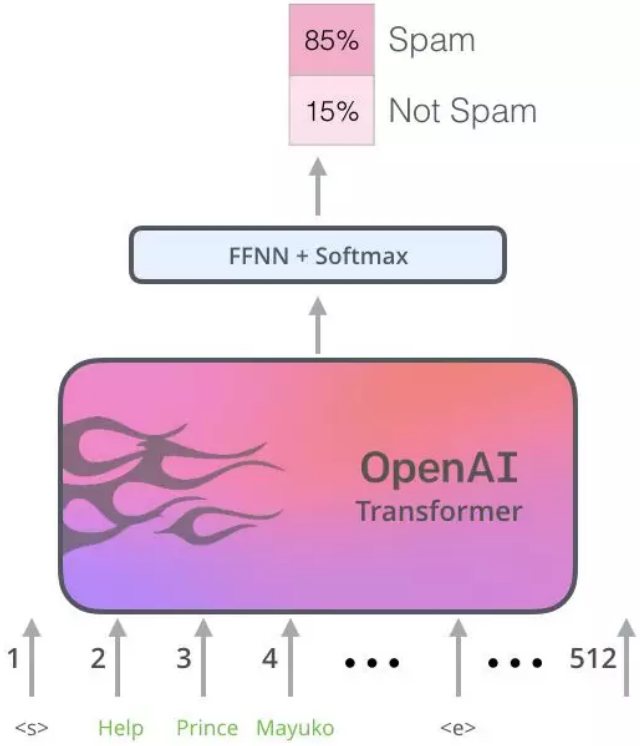
务，因为它允许模型学习相关信息，即使它们被大量文本分隔——假如使用推特或文章进行训练，就无法获得这些信息。



OpenAI Transformer用由7000本书组成的数据集进行训练，以预测下一个单词。

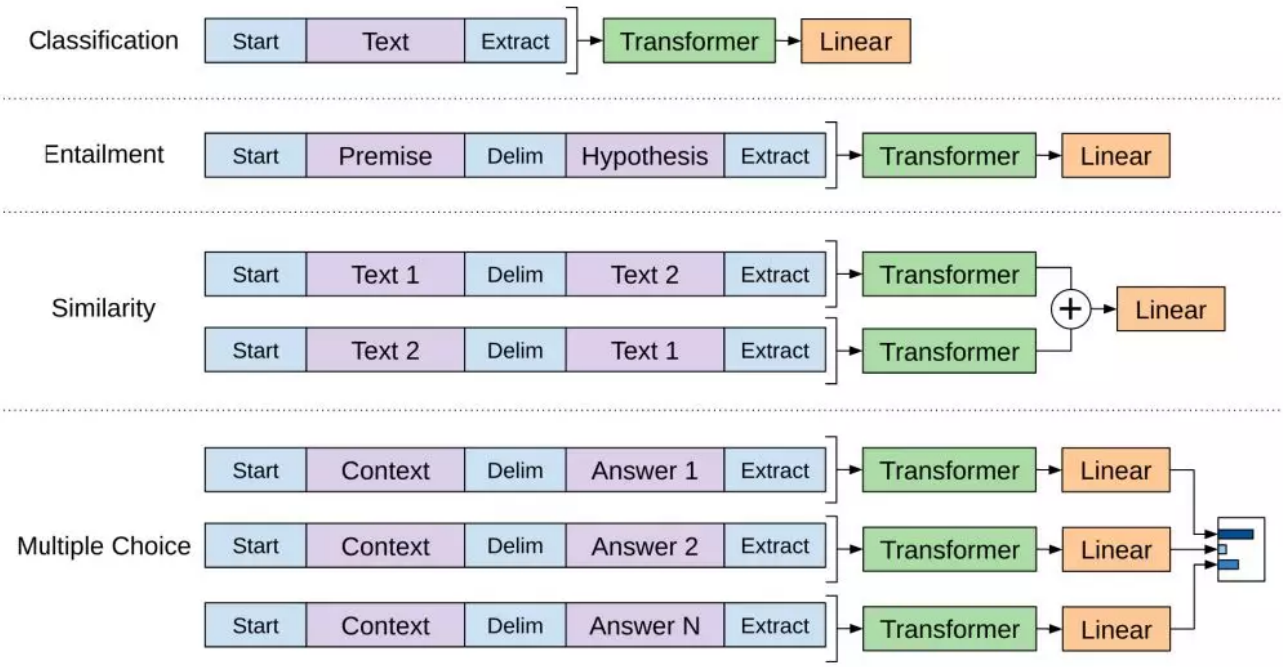
将学习转移到下游任务

既然OpenAI transformer已经经过了预训练，并且它的层已经被调优以合理地处理语言，那么我们就可以开始将其用于下游任务。让我们先来看看句子分类(将邮件分为“垃圾邮件”或“非垃圾邮件”)：



如何使用预训练的OpenAI transformer来进行句子分割

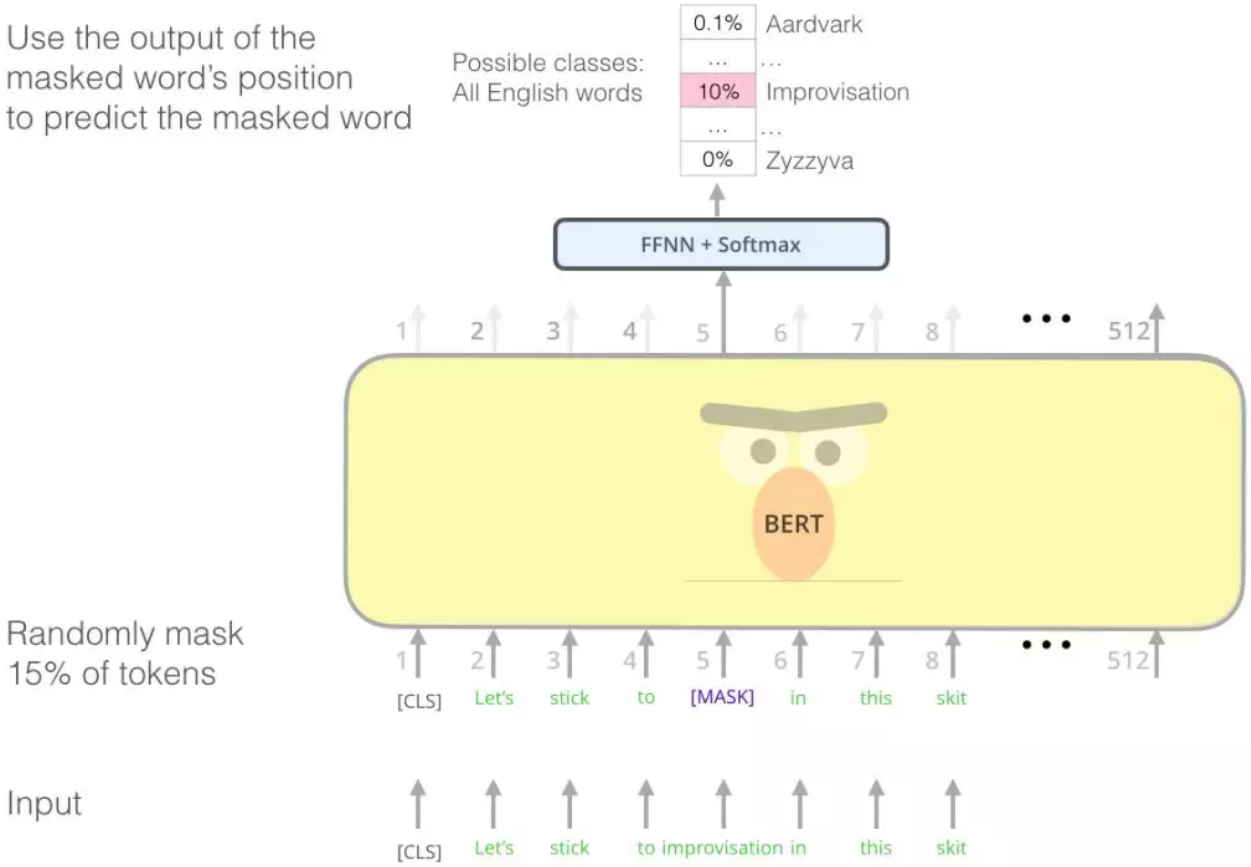
OpenAI论文中概述了一些用于处理不同类型任务输入的输入转换。下图描绘了模型的结构和执行不同任务的输入转换。



这是不很是聪明?

BERT：从解码器到编码器

OpenAI transformer为我们提供了一个基于Transformer的可微调预训练模型。但是在从LSTM到Transformer的转换过程中缺少了一些东西。ELMo的语言模型是双向的，而OpenAI Transformer只训练一个正向语言模型。我们能否建立一个基于transformer的模型，它的语言模型既考虑前向又考虑后向(用技术术语来说，“同时受左右上下文的制约”)?



BERT聪明的语言建模任务遮盖了输入中15%的单词，并要求模型预测丢失的单词。

找到合适的任务来训练Transformer的编码器堆栈不容易，BERT采用了“masked language model”的概念(文献中也成为完形填空任务)来解决这个问题。

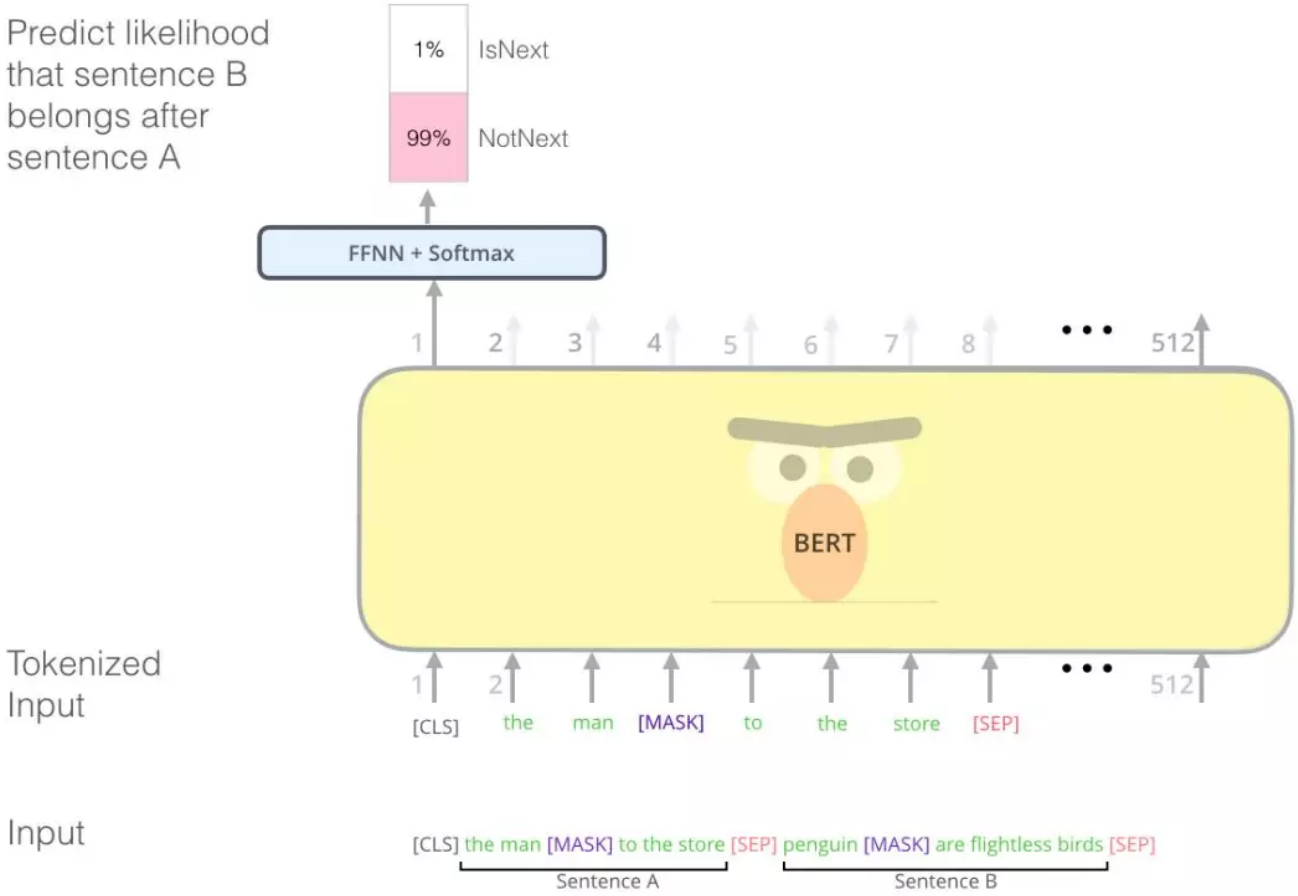
除了遮盖15%的输入，BERT还混入了一些东西，以改进模型后来的微调方式。有时它会随机地将一个单词替换成另一个单词，并要求模型预测该位置的正确单词。

两句话任务

如果你回顾OpenAI transformer处理不同任务时所做的输入转换，你会注意到一些任务需要模型说出关于两个句子的一些信息(例如，它们是否只是同件事情的相互转述?假设一个维基百科条目作

为输入，一个关于这个条目的问题作为另一个输入，我们能回答这个问题吗?)

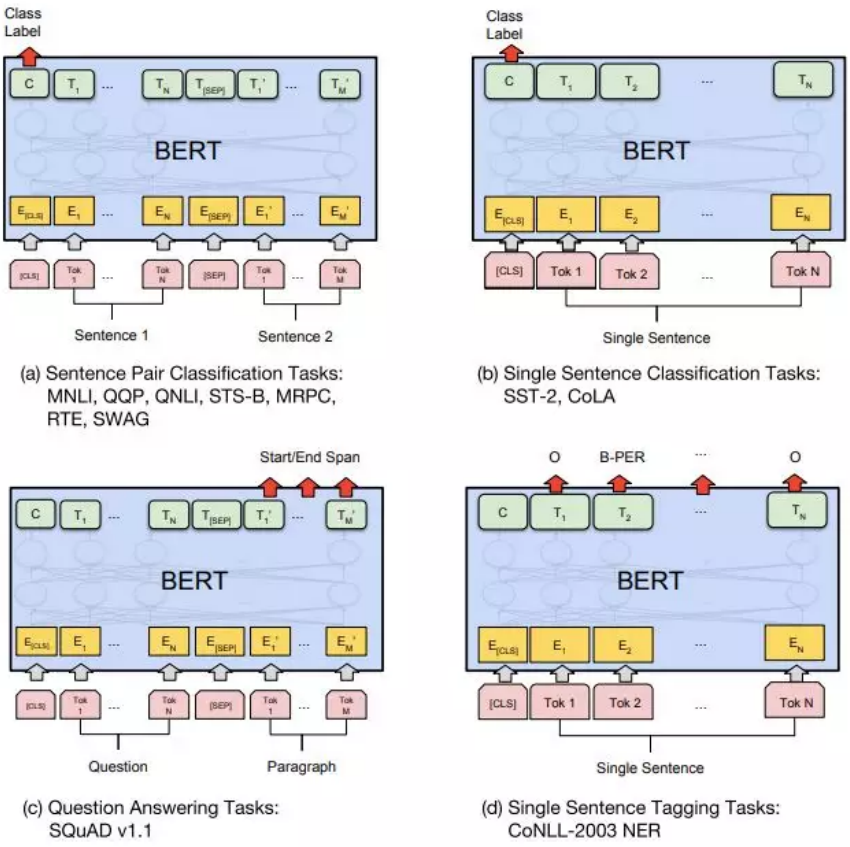
为了让BERT更好的处理多个句子之间的关系，预训练过程增加了一个额外的任务：给定两个句子(A和B)， B可能是A后面的句子，还是A前面的句子?



BERT预训练的第二个任务是一个两句话分类任务。

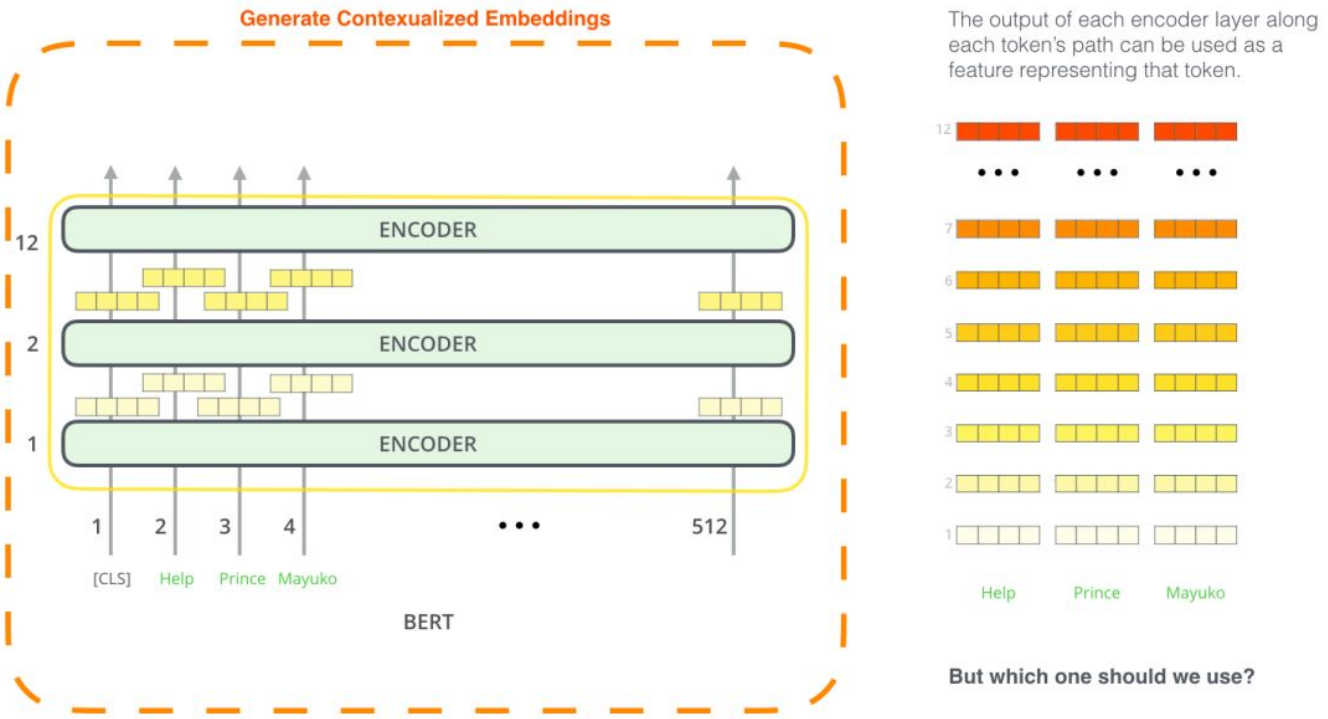
特定任务的模型

BERT的论文展示了在不同的任务中使用BERT的多种方法。

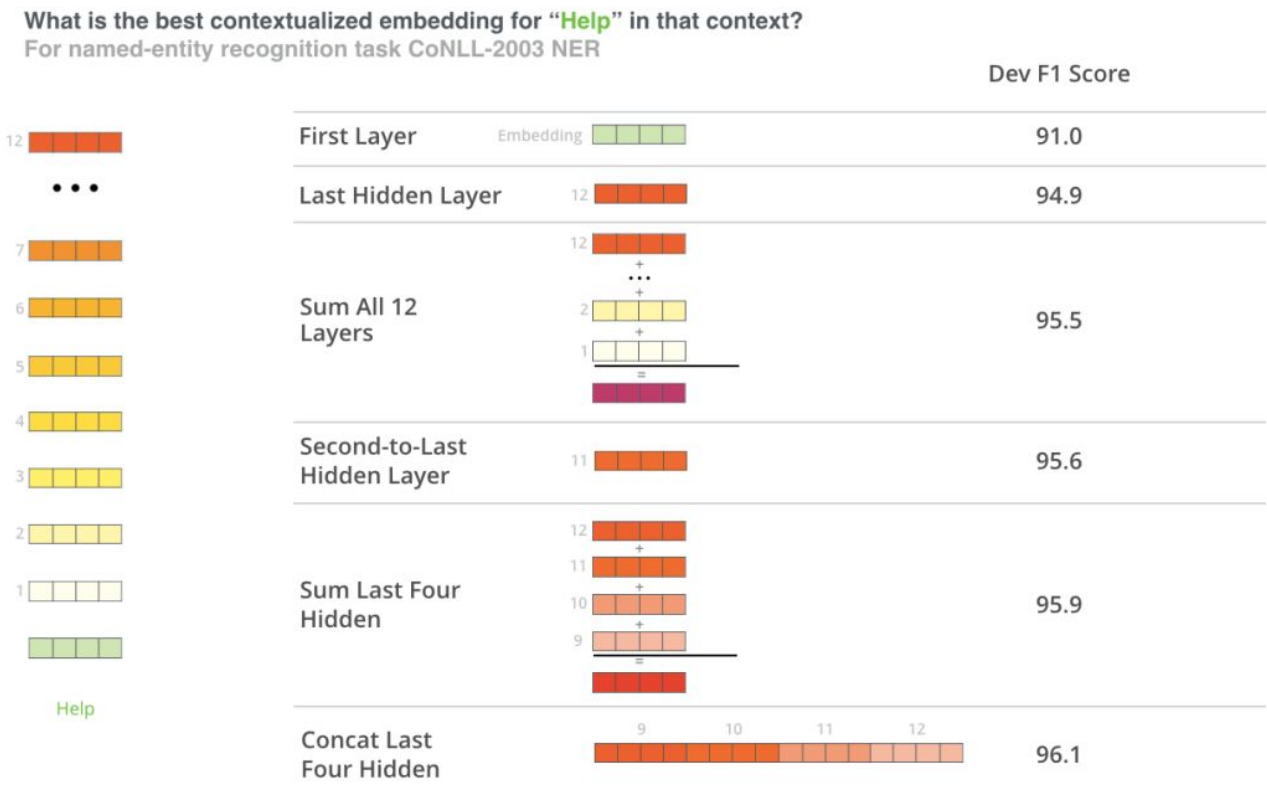


BERT用于特征提取

fine-tuning 方法并不是使用BERT的唯一方法。就像ELMo一样，你可以使用经过预训练的BERT来创建语境化的单词嵌入。然后，你可以将这些嵌入提供给现有的模型——论文中证明了，在诸如名称-实体识别之类的任务上，这个过程产生的结果与对BERT进行微调的结果相差不远。



哪个向量最适合作为语境化化嵌入？我认为这取决于任务。论文考察了6个选项(与得分96.4的fine-tuned模型相比)：



结语

试用BERT的最佳方式是通过使用托管在谷歌Colab上的Cloud TPUs notebook的BERT FineTuning。如果你以前从未使用过云TPU，那么这也是一个很好的起点，可以尝试使用它们。BERT代码也适用于TPU、CPU和GPU。

下一步是查看BERT repo中的代码:

- 该模型是在modeling.py（BertModel类）中构建的，与原始Transformer编码器完全相同。
- run_classifier.py是fine-tuning过程的一个示例。它还构建了监督模型的分类层。如果要构建自己的分类器，请查看文件中的create_model() 方法。
- 有几个预训练模型可供下载。包括BERT Base和BERT Large，以及英语，中文等语言的单语言模型，以及涵盖102种语言的多语言模型，这些语言在维基百科上训练。

- BERT不是将单词看作token。相反，它关注的是词块(WordPieces)。tokennization.py 是将单词转换成适合BERT的WordPieces的工具。

BERT也有PyTorch实现。AllenNLP library 使用这个实现，允许在任何模型中使用BERT嵌入。

英文原文：

https://jalammar.github.io/illustrated-bert/

- EOF -

推荐阅读 — 点击标题可跳转

1、10 个常用机器学习算法

2、机器学习算法优缺点综述

3、回归、分类与聚类：三大方向剖解机器学习算法的优缺点