[CLS]预训练语言模型的前世今生[SEP]萌芽时代[SEP]

原创 管扬 朴素人工智能

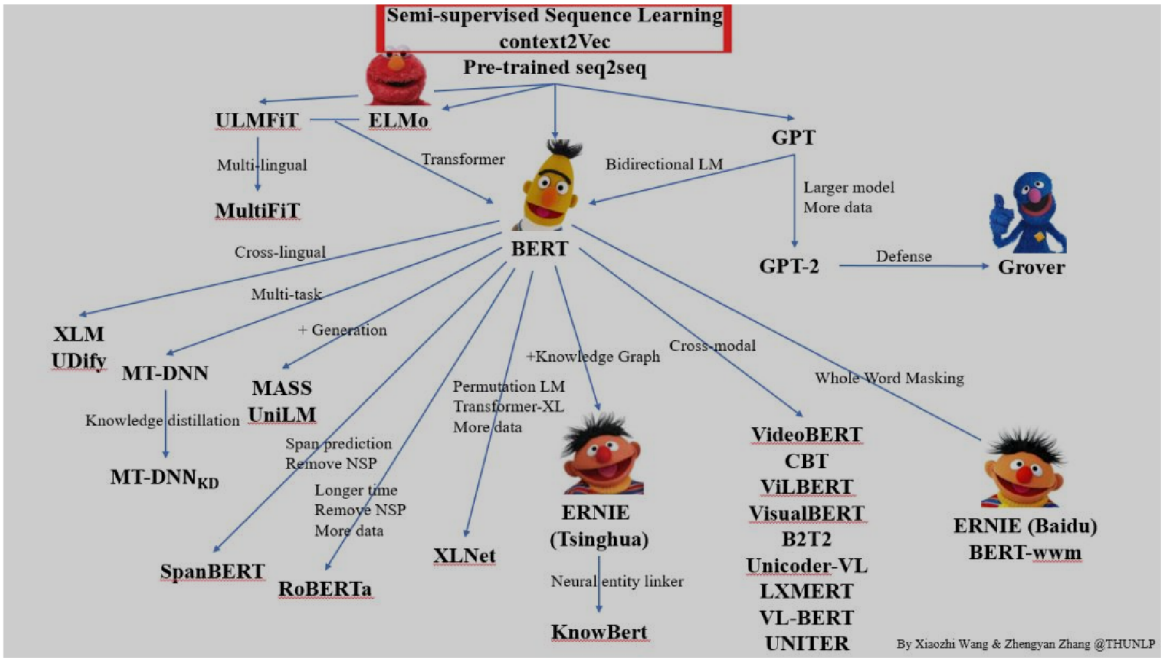
来自专辑

预训练语言模型

之前，我们公众号发表过几篇前沿论文的阅读笔记，欢迎大家去阅读和交流。而在这里，我们将会做一个NLP专题的系列阅读，专门阅读某些专题的论文。第一个当然是预训练语言模型，之后可能会有阅读理解，以及其他专题的阅读。
因为文章较多，每个专题系列，都会分多篇推送来给大家呈现，敬请期待！

前言

前段时间，在github里发现一个很不错的repo，是母校自然语言处理实验室维护的关于自然语言处理中预训练语言模型的**必读论文**推荐，在此奉上链接，并由衷感谢他们的整理。
https://github.com/thunlp/PLMpapers



在学术界，预训练语言模型的研究，已经变成一个非常火热的课题，最近几年迸发出许多可以说改变整个NLP世界格局的文章，包括BERT, GPT2等等。近段时间来，我们在工作和研究中使用BERT等来进行模型训练或业务开发变得越来越普遍。使用预训练模型大大提升了在相关任务上的效果，同时降低了训练的难度。所以，我想趁这个机会，来通过阅读这些文章，**梳理和分享**一下我眼中的预训练语言模型的演进，同时总结一些在使用预训练模型时的心得和总结，希望能给NLP的初学者们一点帮助和启示，同时也希望抛砖引玉，能吸引更多的NLP专家们交流，对我们进行指点和指正。

萌芽时代（2015-2016）

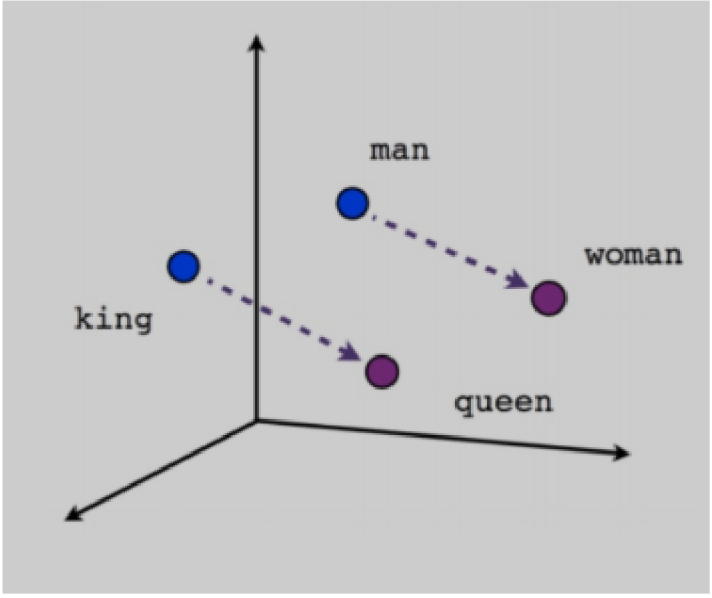
这篇文章起名为萌芽时代，一方面指的是我们今天将要介绍的两篇文章是NLP预训练语言模型刚刚如雨后春笋般冒出萌芽的阶段，他们开创了预训练语言模型的**先河**，并且与当时流行的词嵌入方法相承接。同时我们的公众号也处于萌芽阶段，日后会与大家分享更多NLP方面的思考。

语言模型

言归正传

我们通常所理解的语言模型简单来说就是建模一句句子存在的可能性，我们提到的预训练语言模型 （PLM，Pre-trained Language Model）指的是利用大量在人们生活中出现过的文本来训练，使模型在这些文本中，学习到每一个词或字出现的概率分布，以此来建模出符合这些文本分布的模型。比如这个模型预测“我要吃苹果”这句话在现实生活中出现的可能性比较高，而“我吃要果苹”就不那么高了。语言模型，几乎都是根据上下文来预测相邻的词或字是什么，根据训练任务不同会有一些差别，这个我们后面会介绍到。所以语言模型的语料的标签就是它的上下文，**不需要重新标注**，这就决定了人们几乎可以无限制地利用大规模的语料来训练语言模型，使其学习到丰富的语义知识，这点非常重要，在相关论文中也通常会提及所使用的语料，比如BERT使用了BooksCorpus (800M words) (Zhu et al., 2015) 和English Wikipedia (2,500M words) 来训练语言模型。正因为语料的规模之大，使预训练语言模型得以获得了强大的能力，进一步在我们下游相关任务上展现了其出色的效果。

预训练模型其实在图像中早已被应用，而预训练语言模型的概念于2015 年被认为首次提出（Dai & Le,2015，Semi-supervised Sequence Learning）。直到近期，PLM才被证明在各个任务中的效果优异。在此之前，NLP领域更流行的是使用**词嵌入**（word embedding），将词嵌入一个多维空间中，并且在比较大的语料库上训练后，用以捕捉词和词之间的特定关系。比较知名的词向量训练模型有Word2vec，GloVe 等。词嵌入可以用于初始化下游模型的第一层嵌入层，加上其他功能层，进行整个模型的构建，但早期的词嵌入的方法没有保留每个词上下文的信息，有其局限性。



Semi-supervised Sequence Learning（2015）

这样就引出了，我们今天要介绍的第一篇文章，也可以说是预训练语言模型的**开山之作**，由Google的两位作者Andrew M. Dai和Quoc V. Le发表的 Semi-supervised Sequence Learning。

他们主要使用无标签的数据（即普通的文本）和RNN进行了两种尝试，实际上是通过两种不同训练目标来预训练了LSTM。第一种是训练模型去预测一句句子里下一个词是什么，这是一种典型的语言模型训练方法，第二种是训练模型成为一个autoencoder，用于将句子映射成向量后重建原来的句子。他们**明确**提出了，这两种算法可以为接下来的监督学习算法提供“pretraining step"，换句话说这两种无监督训练得到的参数可以作为接下来有监督学习的模型的起始点，他们发现这样做了以后，可以使后续模型更稳定，泛化得更好，并且经过更少的训练，就能在很多分类任务上得到很不错的结果。

众所周知，RNN模型虽然对时序数据建模很强大，但是因为训练时需要 “back-propagation through time”， 所以训练过程是比较困难的。Dai 和 Le 提出的预训练的方法，可以帮助RNN更好的收敛和泛化，而且在特定业务上不需要额外的标注数据，只需要收集成本很低的无标注的文本。而且这些文本与你的特定业务越相关，效果就会越好，他们认为如此一来，这种做法就支持使用大量的无监督数据来帮助监督任务提高效果，在大量无监督数据上预训练以后只要在少量监督数据上fine-tuning就能获得良好的效果，所以他们给这篇论文取名为 “**半监督序列学习**”。

他们接下来使用他们的两种预训练方法，第一种称为LM-LSTM， 第二种称为SA-LSTM。在sentiment analysis 任务(IMDB and Rotten Tomatoes) and text classification 任务 (20 Newsgroups and DBpedia)，与当时的state-of-the-art 方法进行对比，我就简单截取了一段对比结果如下。之后他们又进行了一些更细致的对比，不再赘述。可以看出SA-LSTM的效果会**普遍优于**之前的最佳结果

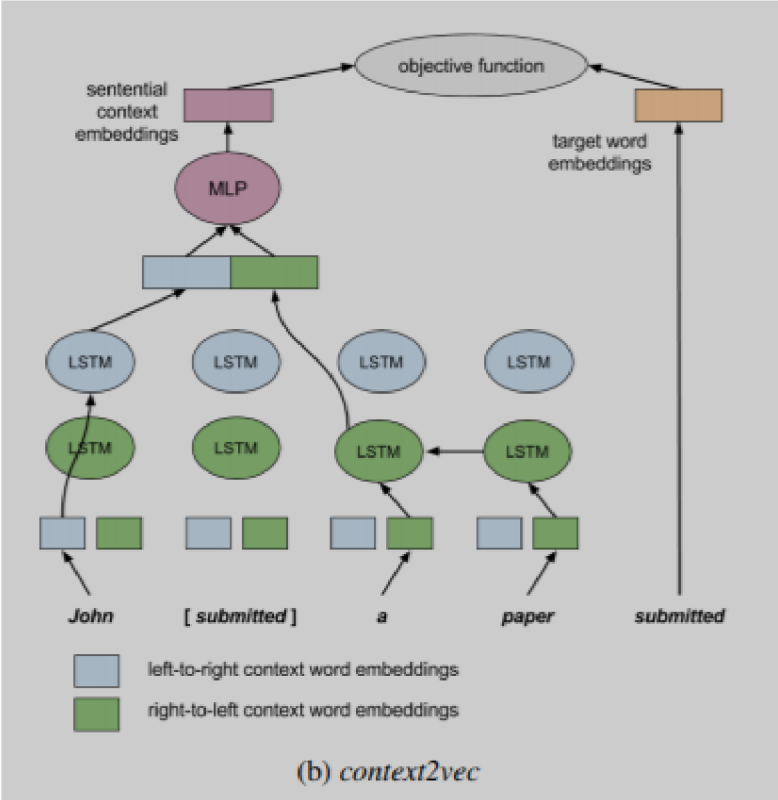
Table 1: A summary of the error rates of SA-LSTMs and previous best reported results.

Dataset	SA-LSTM	Previous best result
IMDB	7.24%	7.42%
Rotten Tomatoes	16.7%	18.5%
20 Newsgroups	15.6%	17.1%
DBpedia	1.19%	1.74%

个人感觉这篇文章最大的贡献，就是**系统地阐述**了上游预训练语言模型用以下游特定任务这个划时代的思想，并通过两个训练目标，用一系列分类任务的实验来对比支持了这个观点。从此以后，预训练语言模型渐渐步入了人们的视野，更在之后由一系列更优秀更强大的模型发扬光大。虽然这篇文章放在今天看来，它的思想已经让我们感到理所当然，它的效果也没有那么地让人感觉惊艳，但是放在那个历史时刻里，它就像一盏明灯，为人们照明了一个方向。

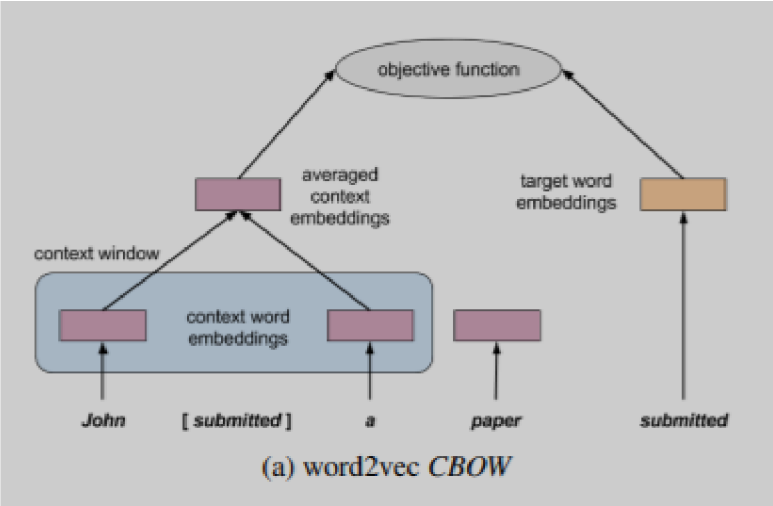
context2vec: Learning Generic Context Embedding with Bidirectional LSTM（2016）

第二篇文章是2016年由巴伊兰大学Oren Melamud，Jacob Goldberger， Ido Dagan 三位作者所发表的context2vec 。从题目名字就可以看出，这篇文章提出的idea是学习文本中包含上下文信息的embeddings。由于词在不同上下文可以有歧义，相同的指代词也经常在不同上下文中指代不同的实体。所以NLP任务中很重要的就是考虑每个词在其上下文中所应该呈现的向量表达方式。从摘要看，这篇文章的主要贡献，是它使用了双向的LSTM可以从一个比较大的文本语料中，有效地学到了包含上下文信息的**embeddings**，在很多词义消岐，完形填空的任务上都取得了不低于state-of-the-art的效果。同时他们提到，之前的研究有把上下文的独立embedding收集起来，或是进行简单的平均，而没有一个比较好的机制来优化基于上下文的向量表达。所以，他们提出了context2vec ，一个能够通过双向LSTM学习广泛上下文embedding的非监督的模型。



(b) *context2vec*

上图是context2vec的结构图，可以说沿袭了word2vec的基本思想。但在word2vec中的CBOW架构中，上下文的embeddings只是拿过来简单的平均了一下，而context2vec则把上下文的数据通过了**双向的LSTM**，左侧蓝色的正向embeddings和右侧绿色的反向embeddings分别经过LSTM的流转，最后连接起来通过多层感知机，产生红色的包含上下文信息的context embeddings，最后通过目标函数去进行整体训练。下图是word2vec的结构图。



(a) word2vec *CBOW*

通过LSTM的长时记忆能力，context2vec可以捕捉这句话中更远位置的**上下文信息**，同时将序列之间的联系包含到了模型中来，接着多层感知机，又将左右两边的序列信息很好的结合在一起，成为真正意义上包含上下文语义信息的context embeddings.

到现在我们看明白了，无论从结构还是名字都表明，context2vec是word2vec的改进版，相比于word2vec，context2vec利用双向LSTM获得了**句子级别**的的上下文语义表示。因此，我们就能够衡量一个词和某一整段上下文文本的相似性，比如下图就是context2vec对于某一整段上下文的本文，最接近的目标词的预测。

Sentential Context	Closest target words
This [ ] is due	item, fact-sheet, offer, pack, card
This [ ] is due not just to mere luck	offer, suggestion, announcement, item, prize
This [ ] is due not just to mere luck, but to outstanding work and dedication	award, prize, turnabout, offer, gift
[ ] is due not just to mere luck, but to outstanding work and dedication	it, success, this, victory, prize-money

看到这我们不禁要问了，我们这个专题不是预训练语言模型吗？那它跟PLM有什么关系呢？其实看着这张图，我们就可以联想到，建模最可能出现的文本序列，不就是语言模型的功能吗。论文中也提到，由于context2vec 利用文本上下文的机制，context2vec 和语言模型的关系是**非常紧密**的，比如它们都是通过上下文和目标词来训练模型，从而捕捉每个词和周围词的联系。主要区别在于语言模型着眼于优化通过上下文来预测目标词的条件概率，而context2vec是想找出一个有效的向量表达来刻画目标词和整个上下文的关系，所以这篇文章可以认为是传统词嵌入和预训练语言模型的一条纽带。最后，文章又在完形填空，词义消岐等任务上评测了一下context2vec的效果，都能达到或超过state-of-the-art的水平。

我认为的这篇文章的主要贡献，是它开创性地通过**双向**LSTM将上下文的语义信息融入到了词嵌入中去，并且首先承接起了当时正流行的词嵌入和语言模型之间的关系，展示了可以利用大量的无标注文本数据，在可接受的时间内，训练出高质量的**包含上下文信息的向量表示**，并显著超过使用传统的词向量的效果。

未完待续

由于篇幅关系，这期就先介绍这两篇论文，他们是整张图中发表最早的两篇论文，也是比较有开创性的两篇，如果想要对词嵌入技术进行更深的了解，建议深入阅读词嵌入方面的相关论文，我也会在下方进行一些推荐。

本专题下期将会继续给大家分享图中的论文，邀请大家一起体会和见证预训练语言模型的发展。在我们心得和记录中难免会有不妥之处，请大家多多包涵，多多指正，**下期再见**，感谢！

推荐阅读

1. 1. Efficient Estimation of Word Representations in Vector Space (word2vec)

   https://arxiv.org/abs/1301.3781
2. 2. GloVe: Global Vectors for Word Representation

   https://nlp.stanford.edu/pubs/glove.pdf
3. 3. Listen, attend and spell

   https://arxiv.org/abs/1508.01211

文章已于修改