

深度学习系列讲座：谷歌翻译核心技术

Seq2Seq 1

前言

本文主要介绍的是 sequence to sequence，这个技术主要是用来做文本理解、机器阅读方面的事情。

在去年，人工智能在应用领域里有三个大的标志性的突破：



首先，大家都知道的 AlphaGo 与韩国著名棋手李世乭的人机围棋大战，突然之间打破了世界对人工智能的认知发生了180度大逆转，先前人们觉得人工智能不太靠谱，还是一个不切实际的概念，现在觉得人工智能无所不能，搞不好20年之后人工智能要统治人类了。大家觉得它20年之后能统治我们吗？我是有点怀疑，但是很多人觉得这很恐怖。

第二个，特斯拉上的 Tesla Autopilot 辅助导航系统，虽然它号称是辅助，但实际上已经属于无人驾驶系统了，因为确实有很多人上车之后不扶方向盘了。这件事情标志着说无人驾驶这个技术基本上可以商业化，所以在2017年的时候，无人驾驶这个行业火得一塌糊涂。

百度开源，为什么？因为这个无人驾驶这个领域的竞争比赛已经进入下半场了，上半场的任务是抢夺领先地位，而下半场的任务就是把竞争对手干掉，所以百度一开源，很多竞争企业都没了，这就是百度的基本战略设想。

第三个，Google Translate（谷歌翻译），自然语言翻译，中文翻英文，英文翻中文，中文翻法文等功能投入商用了。我们今天要讲一讲为什么这件事情很重要，因为 Google Translate 用证据证明了一些事情。

Google Translate 是突破性的进展

RXTHINKING 北京大数医达

为什么 Google Translate 是突破性进展?

A solid proof of the existence of a semantic representation
which is cross linguistic, differentiable, and editable.

GitChat

确凿地证明了存在
跨自然语言的、可微分的、可编辑的
语义表征方式

3 / 18

第一件事情是**跨语言**，任何自然语言，中文、日文、英文、法文等等，都可以用一种数字向量来表示它的语义。以前这只是一个猜想，Google Translate 把这个技术做到了商用化，猜想基本被证实。

第二件事情是**可微分**，那么什么叫可微分？比如一个词，腹泻和拉肚子，字面上没有一个相同，但大家知道这两个是同义词。如果我们把它表示成一个相量的话，把它减掉 0.1，那是不是近义词？以前是没有办法的，以前的词汇是离散的不可微分的，我们现在找了一个词向量，这个词向量是个数字向量，是可以微分的。

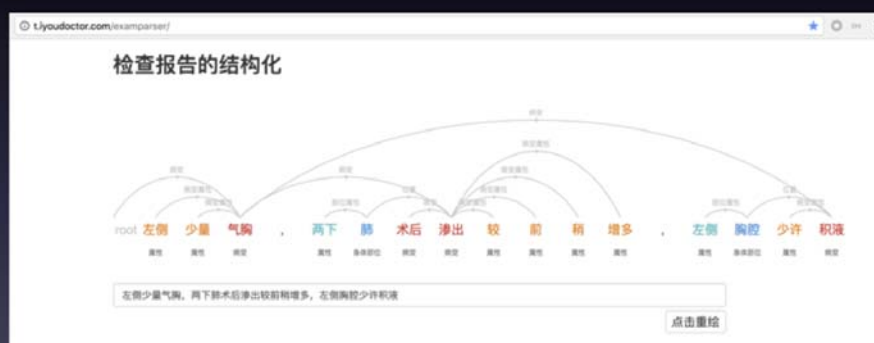
第三件事情**可编辑**，我把好几个词的词向量怎么编辑在一起，像剪接基因一样，能够搞出一个文章摘要、中心思想、关键词，所以词向量还是可编辑的。在这之前大家一直觉得这是一个学术上的研究课题，能不能成，还不确定。

但是 Google Translate 上线之后，业界基本上已经没有任何异议了，说这事基本可行。这就是说，它学术上为什么说有大突破，这个意义就在于说它证明了跨自然语言的可微分的可编辑的语义表征方式。



RXTHINKING 北京大数医达

Seq2Seq 技术的一个应用，文本结构化



GitChat

4 / 18

Google Translate 这个东西其实不仅仅是翻译软件那么简单，它是一个通用的对自然语言的一种新的处理方法。

在医疗领域，做大量的病例结构化时，除了其他传统方法，我们也可以用 sequence to sequence 这种技术来对病例做结构化处理。比如说，有什么病变，发生在什么位置，病变会有很多性质，位置也会有很多性质，所以词与词之间事实上有这种语义的关联性。

我们用这种技术就完全可以把一个自然语言写成语句，把它翻译成结构化的一种表格。Google Translate 用的 sequenceto sequence 并局限于翻译，它实际上是机器阅读通用的方法，所以它的意义是非常大的。

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi
yonghui,schuster,zhifengc,qvl,mnorouzi@google.com

Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey,
Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser,
Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens,
George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa,
Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

Abstract

Neural Machine Translation (NMT) is an end-to-end learning approach for automated translation, with the potential to overcome many of the weaknesses of **conventional phrase-based translation systems**. Unfortunately, NMT systems are known to be computationally expensive both in training and in translation inference – sometimes prohibitively so in the case of very large data sets and large models. Several authors have also charged that NMT systems **lack robustness, particularly when input sentences contain rare words**. These issues have hindered NMT's use in practical deployments and services, where both accuracy and speed are essential. In this work, we present GNMT, Google's Neural Machine Translation system, which attempts to address many of these issues. Our model consists of **a deep LSTM network with 8 encoder and 8 decoder layers using residual connections as well as attention connections from the decoder network to the encoder**. To improve parallelism and therefore decrease training time, **our attention mechanism connects the bottom layer of the decoder to the top layer of the encoder**. **To accelerate the final translation speed, we employ low-precision arithmetic during inference computations**. **To improve handling of rare words, we divide words into a limited set of common sub-word units ("wordpieces")** for both input and output. This method provides a good balance between the flexibility of "character"-delimited models and the efficiency of "word"-delimited models, naturally handles translation of rare words, and ultimately improves the overall accuracy of the system. **Our beam search technique employs a length-normalization procedure and uses a coverage penalty, which encourages generation of an output sentence that is most likely to cover all the words in the source sentence**. To directly optimize the translation BLEU scores, we consider refining the models by using reinforcement learning, but we found that the improvement in the BLEU scores did not reflect in the human evaluation. On the WMT'14 English-to-French and English-to-German benchmarks, GNMT achieves competitive results to state-of-the-art. Using a human side-by-side evaluation on a set of isolated simple sentences, it reduces translation errors by an average of 60% compared to Google's phrase-based production system.

Google Translate 整个的机制，谷歌还是蛮有道义感的一个公司，他非常开放的把他内部的一些细节都写入论文发表出来了，所以大家都可以偷看他到底怎么做的，这篇论文很出名。这篇论文里核心的其实就是下面这张图，如果看懂了这张图，整个论文就看懂了。

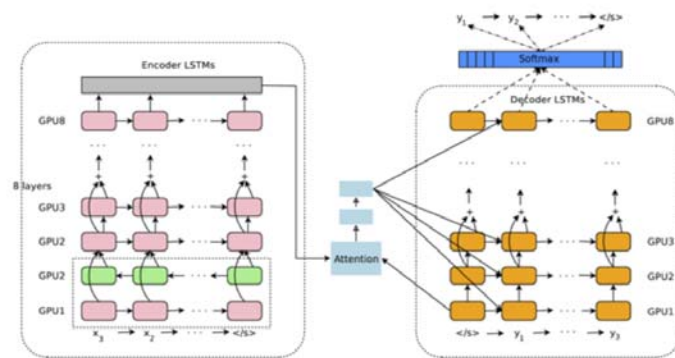
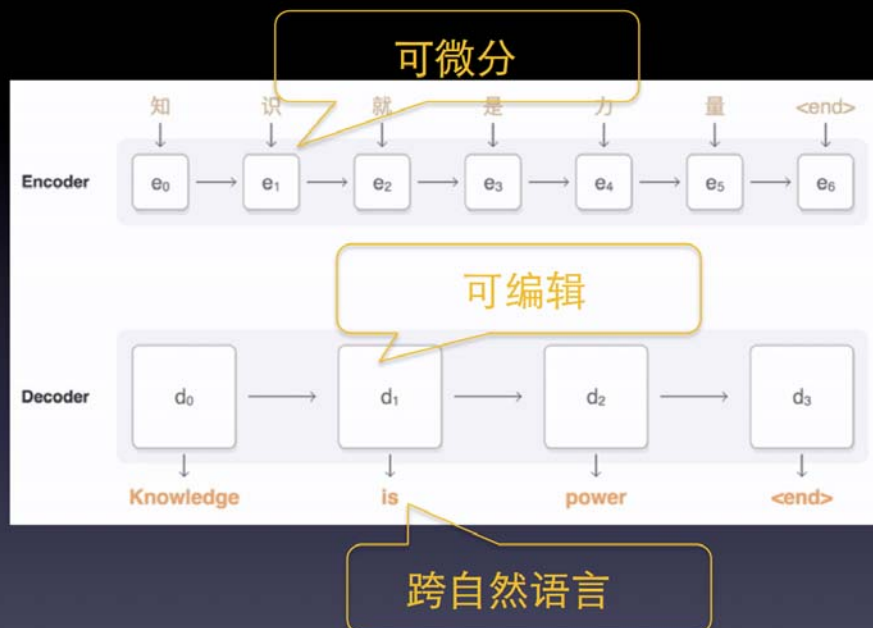


Figure 1: The model architecture of GNMT, Google's Neural Machine Translation system. On the left is the encoder network, on the right is the decoder network, in the middle is the attention module. The bottom encoder layer is bi-directional: the pink nodes gather information from left to right while the green nodes gather information from right to left. The other layers of the encoder are uni-directional. Residual connections start from the layer third from the bottom in the encoder and decoder. The model is partitioned into multiple GPUs to speed up training. In our setup, we have 8 encoder LSTM layers (1 bi-directional layer and 7 uni-directional layers), and 8 decoder layers. With this setting, one model replica is partitioned 8-ways and is placed on 8 different GPUs typically belonging to one host machine. During training, the bottom bi-directional encoder layers compute in parallel first. Once both finish, the uni-directional encoder layers can start computing, each on a separate GPU. To retain as much parallelism as possible during running the decoder layers, we use the bottom decoder layer output only for obtaining recurrent attention context, which is sent directly to all the remaining decoder layers. The softmax layer is also partitioned and placed on multiple GPUs. Depending on the output vocabulary size we either have them run on the same GPUs as the encoder and decoder networks, or have them run on a separate set of dedicated GPUs.

这张图说了些什么？假如有一个中文翻译英文的例子，**知识就是力量**，最终的输出是**Knowledge is power**，输入是中文，翻译成英文，这是它主要的工作目的。



它怎么做呢？实际上是这么几个步骤：

- 第一，它把中文词先翻成一个词向量，变成一个数字向量。
- 第二，它对这个词向量再编辑，变成一个语义表示的方式。
- 第三，再把它翻译成它的目标语言，也就是英文。

它整个用的技术又称为 **encoder（编码）** 和 **decoder（解码）**，另外一种表示就是 **sequence to sequence**，或者概括说是谷歌翻译的核心技术。

这个地方有几件事情，我们刚才说为什么谷歌翻译的核心技术是个划时代的？

之前说了三个关键词：一个是“跨自然语言”，第二个是“可微分”，第三个是“可编辑”。上图这个例子是把中文搞成数字语言，再从数字语言翻译成英文。

大家可以想一想，中文是不是可以翻译成数字语言，再从数字语言翻译成法文呢？当然是可以的。同样的，也可以把现代的白话中文翻译成数字语言，再把它翻译成中国的古汉语文言文。

再给你一段中文的现代文，翻译成数字的语言，再从数字语言可以搞出来一个中心思想，或者说一段中文把它翻译成数字语言，再从数字语言搞成结构化的表格。这就是它的强大所在。



RXTHINKING 北京大数医达

GitChat

Seq2Seq 技术要素

- 用语言模型生成词向量
- 用于语义编辑的LSTM模型
- Attention机制
- 深度学习借力知识图谱
- 准确性评价函数 BLEU

8 / 18

要解释这件事情背后的成因，实际上它有若干个技术的要素，包括怎么去生成一个词向量，词向量是什么，给你一个字或词“知识”，你把它翻译成一个数字向量，这叫词向

量。这个词向量是产生的，这是第一件事情。这里面核心的问题是语言模型，用语言模型来生成词向量，这是咱们本文要讲的事情。

这个系列讲座我们还要谈其他话题，不仅有词向量，一个句子有很多词向量，你要把它再编辑，变成一个语义的表示。用什么东西来做编辑器，在 GoogleTranslate 里用了 LSTM。Facebook 前段时间发了一个论文，他用 CNN 来编辑。怎么去编辑语义，这是第二个话题。

第三个话题是 attention，解码的时候，你要把它翻译成德文，哪一个德文的词汇是最恰当词汇？这个里面用到一个 attention 机制，sequence to sequence 里面还有一个概念就是 attention 聚焦。

attention 之后再展开是什么？现在的 attention 只是从 sequence to sequence 表面字面的意思，没有先验的知识。能不能把知识图谱也融入 attention 里，也就是说我们会有一些先验的知识，这是第四个话题，怎么把符号主义的知识图谱和连接主义的仿生模型，把这两个完全独立的学派搞在一起。

目前为止，业界在做得最好的是 CMU Eric Xing 教授做的一个 student-teacher 模型，那天我跟俊哥讨论的时候，俊哥说他有个新方法。他说是在 attention 里面做一点花样，我听完之后茅塞顿开，我把那个东西称之为孙方法，很了不起的一件事情，超级简单但是超级有用的东西。我会在这里面谈，先谈 student-teacher model，然后再谈孙方法。

最后一个话题是评价函数，在我训练完之后还是不好、有瑕疵怎么办？我们需要一个评价的函数，大家现在通常用的是这个，但这个里面实际上有很多问题需要研究。整个 sequence to sequence 说完这五个大的技术要素，大家基本就明白了。

但是为什么大家会觉得听起来比较困难，主要问题是，现在谈的时候总是一下把五个话题全混在一起，大家自然就有点糊涂了。怎么办？各个击破。所以我们做一个系列，本文先讲第一个话题：**用语言模型生成词向量**。

用语言模型生成词向量



用语言模型生成词向量



徐伟



Yoshua Bengio



Tomas Mikolov

词向量的生成方法，业界现在最流行的方法是用语言模型（language model）来做的，提到这种方法得提三个人，第一个是徐伟，他是在98年最先提出语言模型怎么做，然后给了上图第二个人巨大的启示。

第二个人就是名列世界深度学习的 top 3，Yoshua Bengio。他写的一篇论文非常出名，基本奠定了怎么用语言模型和词向量的整个方法论，这篇论文发表以后，就出现了一个跨时代的巨大突破。

从工程上来讲，现在基本上词向量怎么做，大家已经觉得没什么争议了，最终把词向量这个事情封杀的是谷歌的一位叫 Tomas Mikolov 的工程师，大家都知道词向量就是他做的。

我们今天主要谈谈这个机制是怎么做的，以及业界对这个模型会有什么质疑。

A Neural Probabilistic Language Model

Yoshua Bengio
 Réjean Ducharme
 Pascal Vincent
 Christian Jauvin

Département d'Informatique et Recherche Opérationnelle
 Centre de Recherche Mathématiques
 Université de Montréal, Montréal, Québec, Canada

BENGIOY@IRO.UMONTREAL.CA
 DUCHARME@IRO.UMONTREAL.CA
 VINCENTP@IRO.UMONTREAL.CA
 JAUVIN@IRO.UMONTREAL.CA

Editors: Jaz Kandola, Thomas Hofmann, Tomaso Poggio and John Shawe-Taylor

Abstract

A goal of statistical language modeling is to learn the joint probability function of sequences of words in a language. This is intrinsically difficult because of the **curse of dimensionality**: a word sequence on which the model will be tested is likely to be different from all the word sequences seen during training. Traditional but very successful approaches based on n -grams obtain generalization by concatenating very short overlapping sequences seen in the training set. We propose to fight the curse of dimensionality by **learning a distributed representation for words** which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences. The model learns simultaneously **(1) a distributed representation for each word along with (2) the probability function for word sequences, expressed in terms of these representations**. Generalization is obtained because a sequence of words that has never been seen before gets high probability if it is made of words that are similar (in the sense of having a nearby representation) to words forming an already seen sentence. Training such large models (with millions of parameters) within a reasonable time is itself a significant challenge. We report on experiments using neural networks for the probability function, showing on two text corpora that the proposed approach significantly improves on state-of-the-art n -gram models, and that the proposed approach allows to take advantage of longer contexts.

Keywords: Statistical language modeling, artificial neural networks, distributed representation, curse of dimensionality

10 / 18

上图是 Bengio 那篇著名的论文，大家都读过很多论文，让人印象深刻的并不多，但这篇论文会让你印象非常深刻。论文讲了一个非常厉害的想法，但是整篇论文一口气能读下来，没什么障碍就读完了，非常畅快。

BENGIO, DUCHARME, VINCENT AND JAUVIN

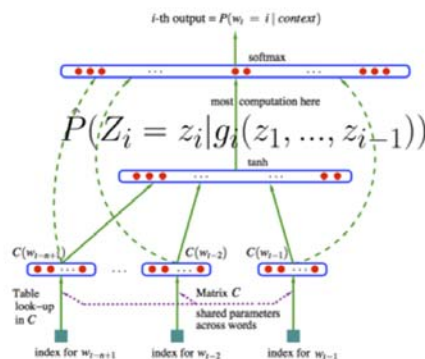


Figure 1: Neural architecture: $f(i, w_{t-1}, \dots, w_{t-n+1}) = g(i, C(w_{t-1}), \dots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i -th word feature vector.

parameters of the mapping C are simply the feature vectors themselves, represented by a $|V| \times m$ matrix C whose row i is the feature vector $C(i)$ for word i . The function g may be implemented by a feed-forward or recurrent neural network or another parametrized function, with parameters ω . The overall parameter set is $\theta = (C, \omega)$.

Training is achieved by looking for θ that maximizes the training corpus penalized log-likelihood:

$$L = \frac{1}{T} \sum \log f(w_t, w_{t-1}, \dots, w_{t-n+1}; \theta) + R(\theta),$$

where $R(\theta)$ is a regularization term. For example, in our experiments, R is a weight decay penalty applied only to the weights of the neural network and to the C matrix, not to the biases.³

12 / 18

如果能看懂上面这张图，这篇论文基本上就看懂了。论文讲的第一件事情是语言模型，这个语言模型很简单，就是给你一篇文章，由若干个句子构成，每个句子有若干个词按顺序构成。



RXTHINKING 北京大数医达

语言模型

$$\hat{P}(z_1, \dots, z_n) = \prod_{i=1}^n \hat{P}(Z_i = z_i | g_i(z_1, \dots, z_{i-1}))$$

人工智能医生帮助人类医生提高临床效率。

$P(\text{全句}) = P(\text{智能}|\text{人工}) * \dots * P(\text{效率} | \text{人工智能}..\text{临床})$

GitChat

11 / 18

现在问题是，我怎么预测下一个词出现的应该是哪个词？它的概率是多少？这就是语言模型。

比方说给你这么一句话，“人工智能医生帮助人类医生提高临床效率”。假如我先告诉你“人工智能”，你推测下个可能出现什么词？下一个可能会出现很多词，什么词都有可能，也许是“医生”，也许是“帮助”等。这里，我先给你一个前面的词，我想猜一猜后续出现“医生”的概率是多少？

这个语言模型要怎么做？其实道理很简单。举例说：我先给图上的句子做分词：“人工”，“智能”，“医生”，“帮助”.....，分好词后做计数，统计每次出现“人工”时，后面出现的“智能”的有多少词？这我们可以统计出来。所以后面提出一个叫 N-gram 的方法。比如说，一句话前面是“人工”，后面是“智能”，“智能”出现的概率远远大于“人工化肥”等其他带“人工”的词，那么我们就认为“人工智能”可能是一个词组，这两个词是经常在一起的，这种方法就叫 N-gram。

它完全是依靠统计，里面有没有透露出“人工”是什么语义，“智能”是什么语义，但是它是一个语言模型。现在问题是，Bengio 对语言模型做了两个地方的改动。

$$\begin{aligned}
 P(z_1 \dots z_n) &= \prod_{i=1}^n P(z_i = z_i | z_1 \dots z_{i-1}) \\
 &= \prod_{i=1}^n P(z_i = z_i | z_{i-t} \dots z_{i-1}) \quad N\text{-gram} \\
 &\quad \downarrow \qquad \qquad \downarrow \\
 &= \prod_{i=1}^n P(z_i = z_i | g(z_{i-t}) \dots g(z_{i-1})) \quad \text{词向量} \\
 &\quad \downarrow \\
 &= \prod_{i=1}^n f(z_i = z_i | g(z_{i-t}) \dots g(z_{i-1})) \quad \text{神经网络}
 \end{aligned}$$

根据上图公式，第一行首先是最原生态的语言模型，给定一个句子中间前一句的词，现在要猜测下面一个位置出现这个词的概率是多少。举个例子，前一句词“人工智能医生”，现在想猜测“医生”后面出现“帮助”个词的概率是多少，这是它原生态的语言模型。如果句子从第一个词开始一直到你马上要预测的下一个位置， i 是不断增长的。

第二行开始，做一点改造，每次给一个定长的窗口 t ，前续词倒推，如果我把语言模型限定为定长的窗口，那么会降低一点点精度。这个给定长的窗口 t 一个学术名称，就是 N -gram，在我们这里面 N 等于 T 。

这个事情和 Bengio 的工作没有任何关系。Bengio、徐伟等人做了什么事情？

第一个事情，从第三行起，前续的词不再是原来的那个 z 符号，它先把它翻译成一个数字向量，我用 g 来表示，这是一个100维或者200维的或者256维或者512维的数字向量，从这边转换到这边，变成一个词向量。

第二件事情，你的条件概率用什么函数来模拟？在 N -gram 里面没什么，它就是一个统计概率。从第四行看，在 Bengio 的工作里，他用了一个很简单的神经网络，来模拟条件概率。

所以他做两件事情，第一件事情，从 N -gram 里面把原来的词、符号变成一个数字向量，第二件事情把这个概率用一个神经网络来模拟。

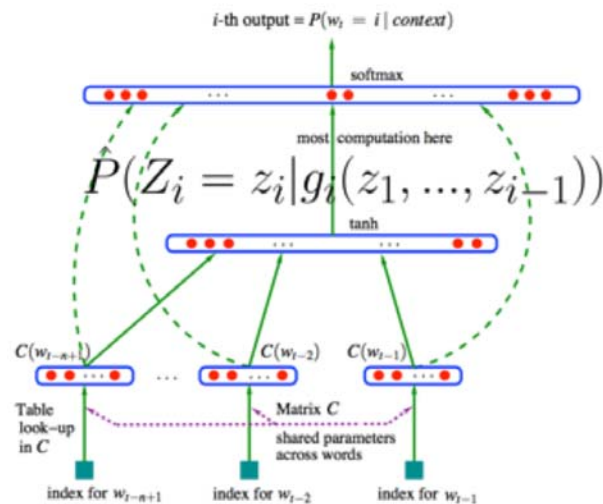


Figure 1: Neural architecture: $f(i, w_{t-1}, \dots, w_{t-n+1}) = g(i, C(w_{t-1}), \dots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i -th word feature vector.

parameters of the mapping C are simply the feature vectors themselves, represented by a $|V| \times m$ matrix C whose row i is the feature vector $C(i)$ for word i . The function g may be implemented by a feed-forward or recurrent neural network or another parametrized function, with parameters ω . The overall parameter set is $\theta = (C, \omega)$.

Training is achieved by looking for θ that maximizes the training corpus penalized log-likelihood:

$$L = \frac{1}{T} \sum \log f(w_t, w_{t-1}, \dots, w_{t-n+1}; \theta) + R(\theta),$$

where $R(\theta)$ is a regularization term. For example, in our experiments, R is a weight decay penalty applied only to the weights of the neural network and to the C matrix, not to the biases.³

虽然这看上去很简单，但确实是很牛的，看上图，有这么几条。

第一条，首先，要去训练这个神经网络，训练词向量的转化函数，这时会需要很多参数，要用大量的训练语料，这些训练语言模型的训练语料从哪里来？非常好找，任何一篇文章都可以作为训练语料，数量几乎是无限，这是它的第一个优点。

那么问题来了，这种训练是无监督学习还是有监督学习？这个训练确实不需要标注它是个无监督学习，但是你看从它训练的输入和输出看，又是个有监督学习，所以它的界限非常宽，你可以说它是无监督，也可以说它是有监督。

有监督学习是从标签化训练数据集中推断出函数的机器学习任务。


无监督学习是根据类别未知(没有被标记)的训练样本解决模式识别中的各种问题。

第二个优点，它整体是如何训练的？词到词向量转换得非常准确，前提假设是：

- 假设一，转换得非常准确。
- 假设二，假设说我们的句子中都有某种予以的连贯性，以至于说你从前面几个词的词向量能够猜测出来下一个词应该出现什么。

那么整个函数 g 加起来拟合的会非常贴切。但是在开始时候你不知道函数 g 是多少，假定函数 g 开始的时候就给它随便设一个随机词（值），但是应该知道它贴合哪个词（值），这两个词（值）中间会有一个距离，这时调词向量的参数和神经网络的参数，不停地调，直到调到两者贴得很近。这就是整个语言模型训练的过程。

所以只要知道这个神经网络怎么训练的，就没那么复杂。好在说它的训练语料几乎是无限的，只要你有足够的计算资源在上面你就可以摆，所以模型非常简单。这就是 Bengio 讲的怎么用自然语言模型来训练词向量。

 RXTHINKING 北京大数医达

arXiv

Natural Language Processing (almost) from Scratch

Ronan Collobert
NEC Labs America, Princeton NJ.

Jason Weston
Google, New York, NY.

Léon Bottou

Michael Karlen

Koray Kavukcuoglu[†]

Pavel Kuksa[‡]
NEC Labs America, Princeton NJ.

**这标题，
太嚣张**

RONAN@COLLOBERT.COM

JWESTON@GOOGLE.COM

LEON@BOTTOU.ORG

MICHAEL.KARLEN@GMAIL.COM

KORAY@CS.NYU.EDU

PKUKSA@CS.RUTGERS.EDU

0398v1 [cs.LG] 2 Mar 2011

Abstract
We propose a unified neural network architecture and learning algorithm that can be applied to various natural language processing tasks including: part-of-speech tagging, chunking, named entity recognition, and semantic role labeling. This versatility is achieved by trying to avoid task-specific engineering and therefore disregarding a lot of prior knowledge. Instead of exploiting man-made input features carefully optimized for each task, our system learns internal representations on the basis of vast amounts of mostly unlabeled training data. This work is then used as a basis for building a freely available tagging system with good performance and minimal computational requirements.
Keywords: Natural Language Processing, Neural Networks

13 / 18

这篇论文出来之后立刻引起冲动，比如说有人写了这么一篇文章，《自然语言（几乎）重起炉灶》。重起炉灶的意思是说，以前的办法不好，需要重来。为什么不是修修补补，而是重来？**因为方法论变了。**

为什么是方法论变了呢？回到最开始的一段话，因为找到了一种“跨自然语言的超级语言”，一种“基因语言”，一种“数字语言”，而且它是连续的可微分的可训练的。

他不仅是说一个一个词可以算它的词向量，把一个句子里面的每一个词都变成词向量，整个句子就变成一组词向量了，而且还可以对这一组词向量不停的编辑，编辑出它的中心思想，编辑出它的语义语法结构。

这个就是说整个方法论变了，以至于会有人写一篇论文说我们得重起炉灶吧。事实证明这篇论文里面所预言的几件事情，几年之后全部变成现实了。最出名的证据就是谷歌翻译，谷歌翻译基本上证明了跨自然语言、可微分、可编辑，这三个事完全成立。

虽然如此，但是会有人质疑，会是哪些质疑？



对词向量的质疑

怎么处理多义词？

“苹果员工爱吃苹果”

如何证明词向量蕴含了语义？

N-gram 也能预测下一个词呀

14 / 18

第一个质疑，同义词。

单纯从这个语言模型来讲，每一个词对应的当前一个词向量，单一词向量你怎么表示同义词。

方法一，每一个词并不一定要对应一个词向量，它可以对应好几个词向量，每一个词向量对应一个不同的语义。

方法二，先在语料中做搜索，查一查“苹果”出现在哪些场景，然后我把有歧义的语料给扔掉，在这一堆语料中间，苹果只是说那个公司。在那一堆语料中间当且仅当，它只是说水果。这样一来，训练出来的“苹果”会有两个完全不同的词向量。

方法可以有很多，所以第一个质疑比较容易解决。

第二个质疑，刚才说它的数字语言是一种超级语言，是一种基因语言，代表了它蕴含的语义，这只是个猜想，如何证明？

比如说苹果翻译成了一个256维的数字向量，请问你前面8个 bytes 代表什么语义，现在不知道，它就像基因一样，给了你一大段核糖核酸，代表了生命中间什么东西不知道，所以大家去研究那事。我们现在问题是说，我确实算出来词向量了，词向量中间每一个 bytes 是什么含义，目前为止不知道。Bengio 的论文也没说到底为什么是这样，只是说好像是。

倒是 Tomas Mikolov，他的贡献主要是他把词向量的工程细节做得很完美了，但在他的论文里他提出一个证明，**证明词向量中间的确是蕴含着语义的。**



Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

这是他的著名论文，这篇论文现在所有做词向量基本上都是按这个规矩来做的。这篇论文里面在它的结论部分有一个旁证，它并不能说这个词向量前面几个 bytes 到底是什么语义，但是它会给你一些证据，说好像这几个 bytes 真的包含着一些语义的意思。



这是一个旁证

Table 1: Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

比如给出两词，“雅典”、“希腊”，雅典是希腊的首都，对应着，“奥斯陆”、“挪威”，奥斯陆是挪威的首都，“雅典”有一个词向量，“希腊”有一个词向量，“奥斯陆”也有一个词向量，“挪威”也有词向量。我把这两个词向量相减，首都的词向量减掉国家的词向量，“雅典”的词向量减去“希腊”的词向量，差不多等于“奥斯陆”的词向量减去“挪威”的词向量，这个很神奇。

Tomas Mikolov 说了，我通过这么一个简单的词向量的减法实验，从侧面反映了雅典的词向量和希腊的词向量中间确实隐含着首都和国家之减的某种关系。同样的减法的思路，也能表达州与州府的关系。

另外，brother 和 sister 这两个词向量相减等于孙子和孙女相减。然后他又说，从形容词和副词的关系，也能够用词向量表达，不仅仅词向量可以反映出语义之间的某种，而且还可以反映出某种语法上的关系，很神奇的一件事情，但是这只是一个旁证。

最后还有一个更强的证据，就是前面提到的翻译。翻译的难度比做这两个词之间有某种语义关系要难得多，从中文翻译成英文，翻得准不准确，大家一眼就知道。如果现在大家觉得翻得很准了，从中文翻译成数字语言，这事听起来有点靠谱。但翻译也还是一个旁证，可是比前面那个词减词的旁证更强大，更确凿的证据。

GitChat