

如何用 word2vec 计算两个句子之间的相似度？

看了下 word2vec，貌似只能计算词之间的相似度，不能计算 sentence 之间的相似度？有人说将 sentence 里的 word vector 直接相加然后归一化后的向量计算 cosine 就可以得到 sentence 之间的相似度，不知道有人试过吗，效果怎么样？

关注问题

写回答


1 条评论

分享

邀请回答

收起

24 个回答默认排序

董力

自然语言处理，北航NLSDE，爱丁堡大学ILCC，PhD student

186 人赞同了该回答

今年ICML-15上有个文章From Word Embeddings To Document Distances，是解决题主的问题。文章里面定义了Word Mover's Distance，词-词的相似度用word2vec结果算欧式距离，句子-句子的相似度通过求解一个transportation的优化问题得到。文章里面证明了词向量求平均算欧式距离是Word Mover's Distance的下界，这样在求解最近邻集合的时候，就可以先用词向量平均对候选集快速进行pruning。

我实现了一下，结果看着还有些道理，但感觉想计算句子-句子的相似度，还是要做sentence modeling才可以，而不是只从word level去算。

如果用word2vec结果作为一种soft matching score的话，其实可以把机器翻译里面的评价指标 BLEU改造一下，也可以计算句子-句子的相似度。

=====

如何定义句子的similarity其实是比较困难的，往往和具体应用也比较相关，到底需求是topic上的相关，还是说semantic上的相关，例如：

- I like this laptop.
- I do not like this laptop.

如果用不同的similarity定义方法，得出的结果也是不同的。这个和paraphrase的研究其实也有些关系，现在大多数工作感觉都是从similarity的角度去做，但其实按照严格定义应该是看双向的 entailment。（扯远了）

=====

只从词去比较的话，还有个比较困难的地方是similarity往往需要结合context去看，例如：

- hot girl（性感）
- hot water（温度高）
- hot dog（吃的热狗）

里面都有hot，但是similarity应该会比较低的。

编辑于 2015-07-25


186

38 条评论

分享

收藏

感谢

孙小宇

机器学习与计算机视觉，伯克利Ph.D.在读

137 人赞同了该回答

@董力 学长提到的Word Mover's Distance a.k.a. WMD，我是这篇ICML论文的第二作者。就算是推销一下自己的论文：)

针对你说的句子间距离的问题，我们的方法是我已知范围内的state of the art。我们测试的数据集里比较的大多是几十个词的段落，你用句子应该会更快，不需要用到董力说到的lower bound pruning。

具体方法很简单：假设我有句子A, B。A里的每个词是A\_i, B里的每个词是B\_j。

- 计算A,B句子里每两个词的距离 i.e.  $D = \text{dist}(A_i, B_j)$  over all  $i, j$ （这里用Euclidean distance b/t the word embeddings, from w2v）。
- 生成optimal transport（也叫earth mover's distance a.k.a. EMD）problem，给

下载知乎客户端  
与世界分享知识、经验和见解

相关问题

机器学习有很多关于核函数的说法，核函数的定义和作用是什么？50 个回答

「社交网络分析」是门怎样的学科？11 个回答

如何用简单易懂的例子解释隐马尔可夫模型？35 个回答

物理专业自学计算机应该学些什么？23 个回答

如何解读「量子计算应对大数据挑战：中国科大首次实现量子机器学习算法」？8 个回答

私家课 · Live 推荐

优惠券|比特币&区块链 25 讲  
共 26 节课

1024 程序员职场进阶必修课  
8 场 Live, 9214 次参与

刘看山 · 知乎指南 · 知乎协议 · 应用 · 工作  
侵权举报 · 网上有害信息举报专区  
违法和不良信息举报：010-82716601  
儿童色情信息举报专区  
联系我们 © 2017 知乎



solver ( 网上有很多各种语言的EMD solver )。输入是D, A所有词的词频(A\_BOW i.e. bag of words), B所有词的词频(B\_BOW)。EMD基本概念就是把两个句子看成两个probability distribution的histogram, A的是山, B的是坑, 用A的山填B的坑, 每两个histogram格之间搬运一个词频单元需要做的功是两词间的距离。

3. EMD返回的就是A,B的距离, 1, 2, 3对每两篇文章可以CPU平行。

论文链接:

[jmlr.org/proceedings/pa...](http://jmlr.org/proceedings/pa...)

关于一词多用以及similarity的定义这两个问题: 这是NLP公认的难题, 我们上面的论文很明显无法解决一词多用 (任何embedding完全基于w2v的模型都不能), 对similarity的定义也是单一的。归根结底, WMD是完全不需要训练的, 而如果想不用人工设计解决上述两个问题, 必须有training data。我们新投稿到NIPS的论文改进了WMD, 用Neighborhood Component Analysis来学习词距, 实验中已经能比较好的区分词义, 由此也能影响similarity的定义。如果NIPS中了, 会在这里贴出链接。

MATLAB核心代码, 抱歉复制来的没有仔细comment。用的solver是emd\_mex。

```
% copyright 2015 Yu Sun
disp('getting pairwise EMD...');
dists = zeros(n_rows, n_cols);
matlabpool open 8
parfor i = 1 : n_rows
    disp(strcat('working on i=', num2str(i)));
    for j = 1: n_cols
        i_BOW = A_BOW{i}./sum(A_BOW{i});
        j_BOW = B_BOW{j}./sum(B_BOW{j});
        DE = sqrt(distance(A_vecs{i}, B_vecs{j})); % L2 norm b/t word vectors
        DE(DE < 0) = 0; % numerical instability may cause negative dists
        [emd, flow] = emd_mex(i_BOW, j_BOW, DE);
        dists(i, j) = emd;
    end
end
matlabpool close
dists(dists < 0) = 0; % numerical instability may cause negative dists
```

编辑于 2015-07-24

▲ 137 ▼

● 56 条评论

➦ 分享

★ 收藏

♥ 感谢

收起 ^



鲁灵犀

浙大数学系, 机器学习, 计算机视觉

108 人赞同了该回答

之前在bat做过这个, 和大家分享一下。

先说一个还是从词的角度出发考虑的, 最后的效果非常好, 就是怎么样从词的向量得到句子的向量, 首先选出一个词库, 比如说10万个词, 然后用w2v跑出所有词的向量, 然后对于每一个句子, 构造一个10万维的向量, 向量的每一维是该维对应的词和该句子中每一个词的相似度的最大值。这种方法实际上是bag of words的一个扩展, 比如说对于 我喜欢用苹果手机 这么一句话, 对应的向量, 会在三星, 诺基亚, 小米, 电脑等词上也会有比较高的得分。这种做法对于bag of words的稀疏性问题效果非常好。

还做过一个直接训练句子的相似度的一个query2vec模型, 效果也不错, 就不细说了。

发布于 2015-07-16

▲ 108 ▼

● 31 条评论

➦ 分享

★ 收藏

♥ 感谢



知乎用户

一切的爱好者 | AI领域

40 人赞同了该回答

2016.7.30

看了知友们的答案, 我再补充一个结合word2vec和RNN来构造**无监督式**的句子特征学习模型。

ps: 本科毕业论文的一个子模块 (E-LSTM-S), 根据文献题目和文献摘要来衡量文本相似度。

在这里，首先给出模型在CiteUlike(文献分享数据集)上的结果。  
用E-LSTM-S得到每篇文献的文本特征后，可以简单地采用欧式距离计算文本相似度，示例如下：

表4-1 E-LSTM-S挖掘到的文本相似性示例

doc_id	doc_title
20 => 6	Spectra of random graph with given expected degrees
	Random graph with arbitrary degree distributions
100 => 5040	EDUTELLA: A PP Networking Infrastructure Based on RDF
	The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration
200 => 6091	Learning to cluster web search results
	Web Document Clustering: A Feasibility Demonstration
6091 => 5282	Web Document Clustering: A Feasibility Demonstration
	A personalized search engine based on web-snippet hierarchical clustering
800 => 971	Ariadne: a secure on-demand routing protocol for ad hoc networks
	A review of current routing protocols for ad hoc mobile wireless networks
300 => 6990	Accurate multiplex gene synthesis from programmable DNA microchips
	Complementary DNA sequencing: expressed sequence tags and human genome project
6000 => 1489	Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data.
	An empirical Bayes approach to inferring large-scale gene association networks

其中A=>B表示跟A文献最为语义相关的是B文献，可以看到：

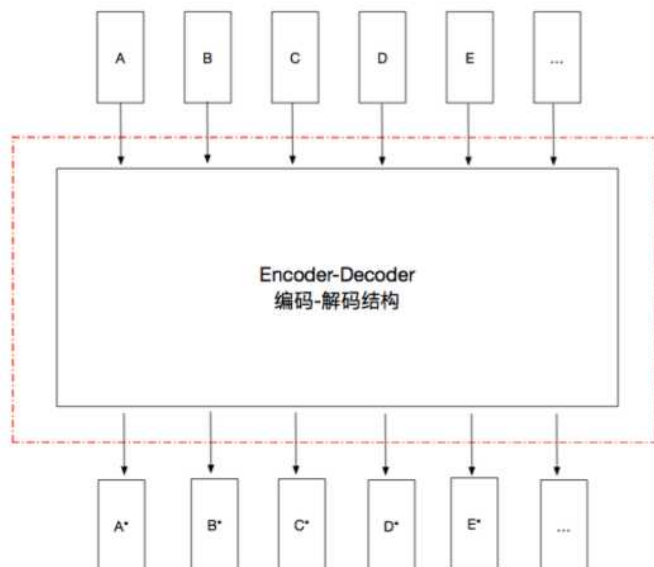
1. 200=>6091表明《Learning to cluster web search results》和《Web Document Clustering: A Feasibility Demonstration》两篇文献具有相近的语义，从标题上也可以直观地判断它们都跟 “web clustering” 相关。

2. 6091=>5282表明《Web Document Clustering: A Feasibility Demonstration》和《A personalized search engine based on web-snippet hierarchical clustering》两篇文献也具有相近的表达，同样跟 “web clustering” 相关。

3. 200=>6091、6091=>5282这个组合表明，跟 “web clustering” 语义相关的文档将构成一类(类似聚类)。

那么，实现的过程如何呢？(这里需要先了解RNN的结构以及如何使用词嵌入embedding-layer)

模型框架如下：

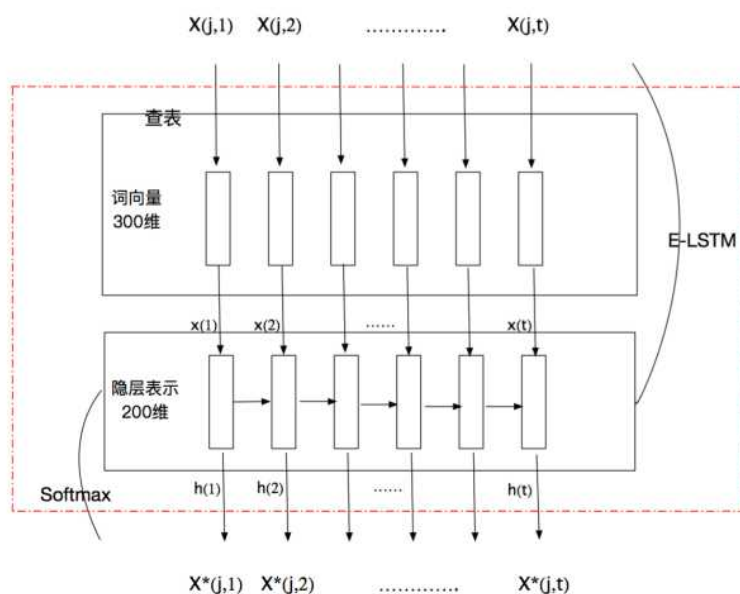


其中，ABCDE...为(词序列构成的)句子， $A^*B^*C^*D^*E^*$ ...同样为相同句子。**模型的核心就是通过编码 - 中间表示 - 解码的形式来学习每个句子的向量表示**，从而可以用距离来衡量句子相似性。

需要说明的是，有两种方式可以用来表示每个词的词向量(实验时选1)：

- 1.采用one-hot vector，定义embedding layer，word2vec作为其初始参数，训练时进行fine-tune。
- 2.直接采用word2vec作为词向量，即在训练过程中不改变每个词的词向量。

接下来，就涉及到了红色虚线处的Encoder-Decoder结构了，细节如下：



其中，上半部分是Embedding，下半部分则是RNN(LSTM)。具体应用时，将词向量表示依次输入RNN的每一时间步，**综合词语义和词顺序的影响**，将其编码成中间表示，再用Softmax解码。

当然，最好采用End-to-End（端到端）的方式直接生成句子的中间特征表示(E-LSTM-S是对每一时间步的隐层输出取平均)，可以参考用于机器翻译、邮件回复的seq2seq模型。这里我给出几篇文献供大家阅读。

<1> [Sequence to Sequence Learning with Neural Networks](#)

<2> [Effective Approaches to Attention-based Neural Machine Translation](#)

<3> [Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation](#)

此外，也可尝试使用CNN替换以上的RNN，目前CNN在文本处理上也有不少研究了。

<4> [A Convolutional Neural Network for Modelling Sentences](#)

<5> [Convolutional Neural Networks for Sentence Classification](#)

<6> [A Sensitivity Analysis of \(and Practitioners' Guide to\) Convolutional Neural Networks for Sentence Classification](#)

-----

ps：不严谨之处，请NLP大牛勿见怪

编辑于 2016-07-31

▲ 40

▼

● 13 条评论

➦ 分享

★ 收藏

♥ 感谢

收起 ^



**山同气**  
深度学习/自然语义应用/图像、视觉相关/预测问题/图模型

32 人赞同了该回答

先回答问题，最简单的方法可以用doc2vec from gensim

=====

为啥要用doc2vec 换句话说，为什么doc2vec是性价比最高的选择。注意我一直强调简单或性价比最高，没说这是唯一解决方案。

我们先看看其他的解决方法：

- 1.用word2vec做：如果word能表示成向量，最直观的思路，对于phrase和sentence，可以将组成它们的所有word向量加起来（简单相加不做任何处理），作为短语向量，句向量。首先这个方法确实有效，但只对15个字以内的短句子比较有效，这种方案常用在搜索时做query比对。如果简单向量相加，丢掉了很多词与词相关意思,也对句子与句子的模式、结构等照成负面影响，无法更精细的表达句子与句子之间的关系。
- 2.用LSI或LSA做：LSI是处理相似度的，而基于的是SVD分解方式，SVD分解方式主要用于特征降维，LSI求解出来的相似度跟topic相关性更强，而句子结构等信息较少。顺便说下，句子中词的顺序是不会影响LSI相似度结果的，可想而知了。

而对于word2vec和doc2vec的模型异同点，以下文章提到了：

引用自：[语义分析的一些方法\(中篇\)](#)

Le和Mikolov在文章《Distributed Representations of Sentences and Documents》里介绍了sentence vector。

先看c-bow方法，相比于word2vec的c-bow模型，区别点有：

- 训练过程中新增了paragraph id，即训练语料中每个句子都有一个唯一的id。paragraph id和普通的word一样，也是先映射成一个向量，即paragraph vector。paragraph vector与word vector的维数虽一样，但是来自于两个不同的向量空间。在之后的计算里，paragraph vector和word vector累加或者连接起来，作为输出层softmax的输入。在一个句子或者文档的训练过程中，paragraph id保持不变，共享着同一个paragraph vector，相当于每次在预测单词的概率时，都利用了整个句子的语义。
- 在预测阶段，给待预测的句子新分配一个paragraph id，词向量和输出层softmax的参数保持训练阶段得到的参数不变，重新利用梯度下降训练待预测的句子。待收敛后，即得到待预测句子的paragraph vector。

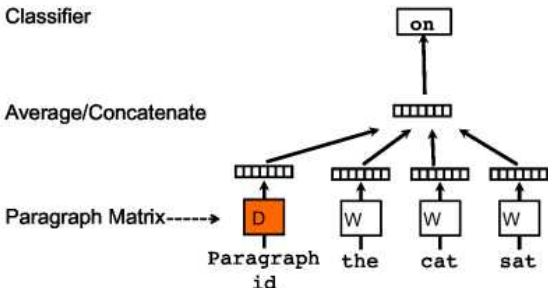
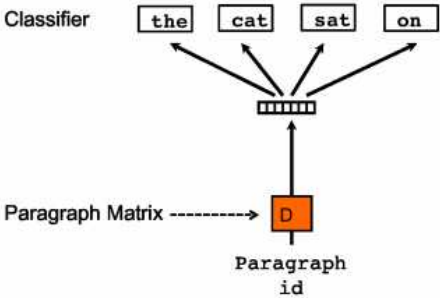


图14. sentence2vec cBow算法

sentence2vec相比于word2vec的skip-gram模型，区别点为：在sentence2vec里，输入都是 paragraph vector，输出是该paragraph中随机抽样的词。



引用完毕。

我们看看gensim文档中出现了一个dbow参数，而且是这么解释的，

dbow\_words if set to 1 trains word-vectors (in skip-gram fashion) simultaneous with DBOW doc-vector training; default is 0 (faster training of doc-vectors only).

也就是要快还是要效果你是可以控制的，这就是体现性价比的地方。

=====

=====

以上，所以推荐doc2vec from gensim

编辑于 2016-02-18

▲ 32 ▼


● 16 条评论

➦ 分享

★ 收藏

♥ 感谢

收起 ^

 **Eastdog**  
资深红烧肉爱好者

9 人赞同了该回答

已经有人提到了，如何定义句子相似度是这个问题的关键。一种理解方式是一个句子是一个word sequence，句子之间的相似度是topic层面的东西；一种理解方式是一个句子表达了一个 meaning，句子之间相似度是衡量两个句子在何种程度上可以paraphrase。

我猜题主想问的是第一种理解方式。从word2vec出发最自然的处理方式，上面有人也提到了，就是向量叠加平均；之后Mikolov等还提出了doc2vec，对词向量进行处理（DM/DBOW），不过基本思路没有差很远。

我在实际工作尝试使用了这些方法，可能在其实用效果上有一些发言权，抛砖引玉。word2vec和doc2vec我都用的gensim的实现，训练在自己的语料（几万文档）上，语料不大跑得快所以各种参数试了很多。因为训练出来的模型没有一个太好的evaluation benchmark（我觉得word analog不太有普遍意义），所以模型的评判更主观一点儿。中文语境下，通过适当的调参（对结果影响比较显著的，我觉得有window size和min count），word2vec的结果还比较能看，doc2vec的结果一直都比较差，尤其是比较长一点儿的句子/文章。

在文档级别上，我觉得doc2vec的robust程度还不足以支撑一个产品，稳健性程度不如LSI，或者简单的tf-idf。@董力 提到的paper我打算看一下，听起来是一个可能可行的方法。

关于句子相似度的另一种理解方式，@董力说的给我很大启发。Textual Entailment角度来看这个问题应该会比较准确，能更深层次地model问题，打算重新温习一下传统方法，看看有没有能跟word embedding结合的点。

发布于 2015-07-10

▲ 9 ▼

● 2 条评论

➦ 分享

★ 收藏

♥ 感谢



**且听风雨**

蚂蚁金服 坐标成都, 持续招聘

12 人赞同了该回答

谢邀。。。这里面牛人太多, 我就说下自己尝试过方法一点看法。

首先, 目前的模型对于长句子绝大部分不能work。Mikolov在google group里面也回答这个模型很难训练句子词向量。

不过对于短句子或者词组效果还是很明显的。

1、项目中想对评价里抽取的属性情感词进行聚类。我将评价里面抽取属性情感词组合成一个词放入语料中, 然后用word2vec训练, 效果非常好, 已经在实际项目中使用。

2、对于短文本如query, 我用word2vec的变种DM/DBOW训练句子向量, 效果也挺不错。不过这个也只是有一定的尝试, 其实编辑距离也能ko绝大部分这种问题。。。

总体上感觉word2vec模型还是过于粗暴, 一个词的意思基本固定不变, 最多也就两三种意思。用word2vec这种基于共现的方法还是不错的。

不过对于句子, 这个就精细很多, 即使绝大部分词语相同, 整句话的意思也很有可能南辕北辙。句子级别的训练, 句法分析感觉绝对少不了, 个人比较看好Istm这种模型。

附上一篇牛人刚写的文章, 很有意思。感受就是很多时候不work其实不是模型的问题, 而是语料和参数。。。

《How to Generate a Good Word Embedding?》导读 [《How to Generate a Good Word Embedding?》导读](#)

发布于 2015-07-23

▲ 12 ▼

● 3 条评论

➦ 分享

★ 收藏

♥ 感谢

**知乎用户**

梅吹, 老司机。

1 人赞同了该回答

[如何通过词向量技术来计算2个文档的相似度? - 知乎用户的回答](#)

引用一下自己的答案~

发布于 2015-08-17

▲ 1 ▼

● 添加评论

➦ 分享

★ 收藏

♥ 感谢

**Andy Huang**

锐利, 嘲讽, 特立独行

1 人赞同了该回答

[Paraphrase Identification \(State of the art\)](#)

发布于 2016-07-20

▲ 1 ▼

● 1 条评论

➦ 分享

★ 收藏

♥ 感谢

**li Eta** ★

机器学习 话题的优秀回答者

5 人赞同了该回答

原始的word2vec只对words之间的关系进行建模。题主要做句子相似度计算, 首先需要定义清楚一个问题: “句子相似” 是什么, “句子不相似” 是什么? , 别看这两个定义naive, 仔细考虑一下会发现, 这两个定义不同, 则句子相似度的建模方法差别也会很大。

如果, 我们采用最简单的定义: “句子相似” = “句子中的词语相似”, 则用word2vec得到的词向量相加, 然后归一化一下, 是可以较好地代表句子的。然而, 如果希望model到词语顺序和语法结构恐怕就不能用这种方法。

此外, 对句子进行建模的paper是有的, 只是大量都不是为了得到句子的embedding, 而是为了处理一些传统NLP任务, 比如分词(使用HMM, CTR)等等。

BTW. 这篇 Distributed Representations of Sentences and Documents 参考意义不大, 只要还停留在word2vec的框架内, 就无法model到语法结构对句子相似度的影响。

编辑于 2015-07-09

▲ 5 ▼

● 24 条评论

➦ 分享

★ 收藏

♥ 感谢



**Serena Gao**  
努力PhD毕业.

12 人赞同了该回答

有不少童鞋问到具体的实现方法，更新一下...

这篇paper的方法并没有具体介绍太多细节，感觉他们着重在表示很简单的pca就能得到很好的效果。另一篇相关文章会对理解这个方法的具体实现有很多帮助，来自同样的作者：

Arora, Sanjeev, et al. "A latent variable model approach to pmi-based word embeddings." Transactions of the Association for Computational Linguistics 4 (2016): 385-399.

附上这篇paper的github repo: [PrincetonML/SIF](#)

<如果原作者的下载量暴增他是不是该感谢我哈哈哈哈>

=====

找草稿找了半天，打不开还是来重新写吧。

上面有人提到sentence2vec，最近有篇很有用的paper叫 "A simple but tough-to-beat baseline for sentence embeddings." Arora, Sanjeev, Yingyu Liang, and Tengyu Ma. 2017. 正如其名，算法简单但performance很好。直接上图：

简而言之就是weighted average word embedding = sentence embedding。其中weights =  $a / (a + p(w))$ ，然后经过pca。

真的很简单吧？我最近在用自己的dataset来跑word2vec和这篇paper, performance真的好很多 (make sense很多)，当然因为没有benchmark所以没法提供一个evaluation metrics.

这篇paper另一个比较convincing的地方是，他们的试验是基于sts task,

其中专门有sentence similarity的task。



个人觉得word2vec就是很虚，因为大家都在用所以拿来用。从syntax意义上来说，它或许能很好的做出representation,可是semantic representation呢？我不觉得word embedding能准确capture。上面几个回答几乎都提到了怎么定义sentence similarity。如果trivial task，或许word embedding就够，performance肯定会比word overlap之类的要好得多，如果真的想要准确表现semantic level, 那就很难说，不觉得是能靠调参数提高performance就能解决的问题。

另一种方法是基于LDA + RNN的lm。还没试过，不过感觉基于lda的话，既能得到distributed representation又有topical semantics。有人能尝试一下就好了。

编辑于 2017-08-16

▲ 12 ▼

● 14 条评论

↗ 分享

★ 收藏

♥ 感谢

收起 ^



裴斐晏

转机器学习神经网络指南

4 人赞同了该回答

效果还行，句子一长效果就差了；

用lstm做句子encoding吧，不需要word2vec这种，根据句子上下文做训练；

不行的话用denoising Autoencoder，这种效果最好哦；

也可以试试seq2seq的最终向量，输入输出都是句子本身；完全无监督；

实在不行只能用那个啥variational Autoencoder了，用kl divergence计算句子的距离；

编辑于 2017-06-07

▲ 4 ▼

● 6 条评论

↗ 分享

★ 收藏

♥ 感谢



马进

不要忘记阶级斗争。

3 人赞同了该回答

怎样利用word2vec计算两篇文档的相似度？有没有大神给我一些启发？ - 机器学习

Mikolov后来又发了一篇：[arxiv.org/abs/1405.4053](https://arxiv.org/abs/1405.4053)，完全符合你的需求。但是实际应用中这篇paper的效果并不好。

可以把词向量作为输入，用LSTM或者CNN来对句子建模。

知乎大神太多，匿了。

发布于 2015-04-30

▲ 3 ▼

● 4 条评论

↗ 分享

★ 收藏

♥ 感谢



vinsin

2 人赞同了该回答

有多种选择，你可以使用similarity.docsim doc2vec，也可以使用lsh，前面两种属于gensim，后面一种属于scikit-learn，具体的例子可以参见 用docsim/doc2vec/LSH比较两个文档之间的相似度

发布于 2016-08-23

▲ 2 ▼

● 1 条评论

↗ 分享

★ 收藏

♥ 感谢



张宏伦

数据爱好者

2 人赞同了该回答

在NLP里面和sentence modeling、semantic matching相关的模型很多，分别是对单个语句进行建模、对两个语句之间的语义关联进行建模。

将一句话所有词的词向量加在一起求平均，将所得结果作为句向量也是一种简单的解决方案，所以由word2vec衍生出了seq2vec和doc2vec之类。

当然模型弄得很复杂，计算成本也会高很多，原理也没那么好理解，最后同样是在一些通用的测试数据集上刷成绩，提高一两个百分点。所以很多模型仅限于研究，在实际应用中还是需要结合模型的性能和复杂度进行选择。

发布于 2017-01-02

▲ 2 ▼

1 条评论

分享

★ 收藏

♥ 感谢



杰瑞朱

1 人赞同了该回答

我用gensim实验了wmd距离，觉得效果还是不错的，但是速度比向量平均后求余弦要慢很多的。但是可以用求余弦的方法来进行快速判断，然后用wmd来精细化判断。下面是我用100万对句子数据画的散点图，用余弦距离做快速筛选还是很靠谱的。

编辑于 2017-09-02


▲ 1 ▼

1 条评论

分享

★ 收藏

♥ 感谢



琦在江湖飘

占个坑

编辑于 2017-11-12

▲ 0 ▼

添加评论

分享

★ 收藏

♥ 感谢



Lapis-Hong

机器学习 深度学习 SJTU

最近也在研究这个方面。

文档相似度首先得看文档长度，如果是短文本，如句子层面，传统方法tf-idf，lsi，lda，余弦相似度以及交并集相似度等方法基本只从词频出发，无法体现相近的词义。效果一般，不过也看具体应用，如果只是词形上的相似而基本不涉及到语义上的相似，那么可用。反之，则需要词向量等方式来解决。

如果是长文本，可以用doc2vec，效果一般，或者用word2vec，然后用wmd度量相似度，但gensim中的实现貌似不能含有没有出现过的词。。。

编辑于 2017-10-10

▲ 0 ▼

添加评论

分享

★ 收藏

♥ 感谢

zhtsky



Life is short, use Python!

在Stack Overflow上看到一个回答

[How to calculate the sentence similarity using word2vec model of gensim with python](#)

还有人说分词后加载word2vec模型直接用 `gensim.models.word2vec.n_similarity(ws1, ws2)` 计算两个词序列之间的相似度, 不知道好不好用

发布于 2017-04-07

▲ 0



● 1 条评论

➦ 分享

★ 收藏

♥ 感谢



知乎用户

3 人赞同了该回答

stackoverflow上有一个相似的提问,

[How to calculate the sentence similarity using word2vec model of gensim with python](#)

按照这个帖子, 对于并不是非常长的句子, 简单的向量叠加可能会起作用。但最高票回答者也承认这是一个非常有挑战性的问题。

This is actually a pretty challenging problem that you are asking. Computing sentence similarity requires building a grammatical model of the sentence, understanding equivalent structures (e.g. "he walked to the store yesterday" and "yesterday, he walked to the store"), finding similarity not just in the pronouns and verbs but also in the proper nouns, finding statistical co-occurrences / relationships in lots of real textual examples, etc.

The simplest thing you could try -- though I don't know how well this would perform and it would certainly not give you the optimal results -- would be to first remove all "stop" words (words like "the", "an", etc. that don't add much meaning to the sentence) and then run word2vec on the words in both sentences, sum up the vectors in the one sentence, sum up the vectors in the other sentence, and then find the difference between the sums. By summing them up instead of doing a word-wise difference, you'll at least not be subject to word order. That being said, this will fail in lots of ways and isn't a good solution by any means (though good solutions to this problem almost always involve some amount of NLP, machine learning, and other cleverness).

So, short answer is, no, there's no easy way to do this (at least not to do it well).

发布于 2015-07-09

▲ 3



● 1 条评论

➦ 分享

★ 收藏

♥ 感谢