5. 词嵌入与词向量

CONTENTS

词向量

文本数据是一种典型的非结构化数据。在对文本数据进行分析时，计算机需要对文字进行处理。由于计算机无法直接理解每个文字的内在含义，因此首先需要将文字转换成计算机能看懂的“语言”，也就是转化为结构化数据。**对于文本数据来说，通常需要将其转化成向量形式。**

词嵌入

传统的转化方式是将文本数据进行one-hot编码，这种编码方式将每一个词语对应一个维度，编码向量的长度即为词语的数量，每一个单词都有其自己的索引。随着机器学习、深度学习的发展，文本的**分布式表示**（distribution representation）方法逐渐发展，这种表示方法**考虑了词与词之间的关系，在将词向量的维度降低的同时也能够反映不同词语之间的关系**，这种方法通常被称为“词嵌入”（embedding）。

常用模型

目前使用最广泛的词嵌入方法是T.Mikolov等在2013年提出的**CBOW模型和Skip-gram模型**。这两种词嵌入模型都是**静态表示方法（static embeddings）**，即当模型训练后之后，会为每个词分配一个词向量，然后**利用向量在空间上的位置关系来表示对应词语在语义上的关联**。

2018年，Devlin等基于Transformer这一结构提出**BERT**（Bidirectional Encoder Representations from Transformers）模型。BERT模型是一种**动态嵌入（dynamic embeddings）**的方法，这种方法能够考虑到**同一个词语在不同的语境下可能存在的不同含义，是一种更加强有力的词嵌入方式**。

一、词的表示方法

离散表示（one-hot representation）

传统的基于规则或基于统计的自然语言处理方法将单个词语看作一个原子符号，这种表示方式被称为**离散表示**，常用的方法为**独热法（one-hot representation）**。简单来说，离散表示首先需要将所有待表示的词看做一个词典（也称为词表），假设词典中词的个数为V。每个词可以被表示为一个**长度为V的长向量**，该向量只有一个维度的值为1，即表示该词在词典中对应的位置，其余的维度均为0。这种方法生成的词向量往往维度很高，但是每一个词语对应的向量只有一个位置取值为1，其他维度取值均为0，因此也常被称为**稀疏的词向量（sparse vectors）**。

离散表示（one-hot representation）

独热法相当于给每个词分配一个索引，这种表示方法**简单易懂、操作方便**。然而缺点是无法反映词与词之间的关系。另外，如果词典中的词很多，即V很大时，**独热法将会导致特征空间非常大，从而产生维度灾难和储存危机。**

向量空间模型（vector space model）

中国人民大学
RENMIN UNIVERSITY OF CHINA

向量空间模型是建立在独热法基础上的一种文本表示方法，由Salton等人于20世纪70年代提出，并成功地应用于著名的SMART文本检索系统。向量空间模型的本质就是把每个文本看成**向量空间中的向量**，并以在**空间上的向量相似度**来表示语义的相似度。

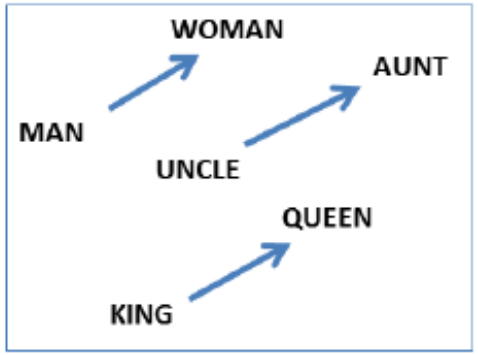
向量空间模型（vector space model）

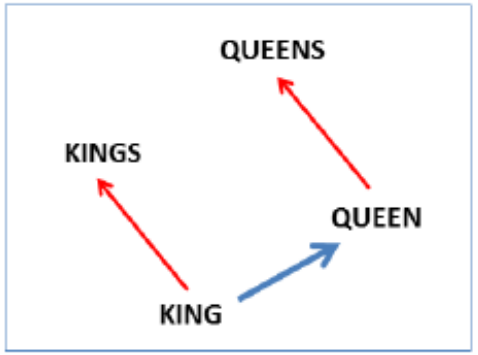
在向量空间模型中，**一个词语或者一段文本均可由向量表示。**如果是词语，其表示方法和独热法一致；**如果是一段文本，其同样由一个长度为V的向量表示，该向量中非0的位置表示这段文本中出现的词语频数。**值得注意的是，向量空间模型中非0的位置除了可以用每个词出现的频数表示外，**还可以计算频率，或者是其他的标准化方法，比如TF-IDF。**Jia et al. (2017)系统的讨论过向量空间模型中的标准化方法。**使用向量空间模型表示后，文本数据就转换成了结构化数据，进而为后续进行文本分析奠定了基础。**

分布式表示

中国人民大学
RENMIN UNIVERSITY OF CHINA

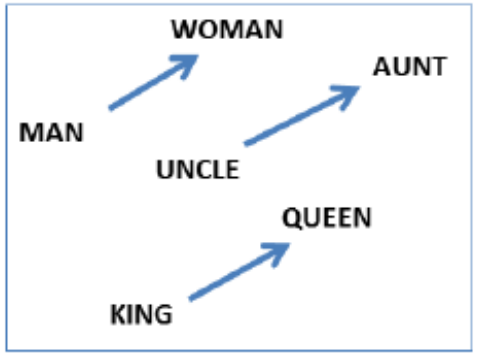
文本的分布式表示是伴随着深度学习兴起的一种文本表示方法，近年来获得了广泛应用。之前提到独热法的一个缺点就是当词典很大时，表示词的向量维度会很高，从而很容易产生维度灾难。分布式表示即将词语表示为某个**低维连续的稠密向量**，通常称为“词向量”（word embedding），从而可以有效降低文本的表示维度。例如，下图中的每个英文单词都被表示为了**二维空间中的一个向量**，每个词语的位置即由它背后的词向量决定。

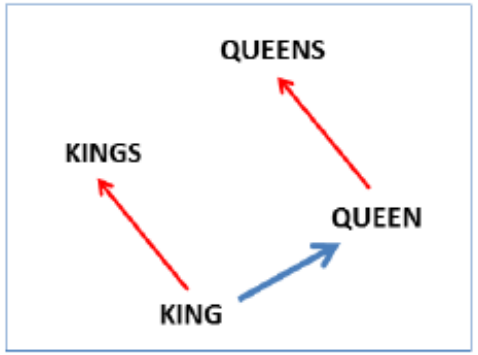




分布式表示

值得注意的是，文本的分布式表示还可以反映**词和词之间的语义相关性**。例如，**MAN的词向量和WOMAN词向量之间的距离，等于UNCLE和AUNT两个词向量之间的距离，也等于KING和QUEEN两个词向量之间的距离，因为上述每对单词之间的差别就在于性别。**正是由于词向量之间的距离可以表示语义之间的关联，因此可以通过词向量的加减构造新的词语，例如，KING的词向量减去MAN的词向量加上WOMAN的词向量就等于QUEEN的词向量，即**vector(“QUEEN”) = vector(“KING”) - vector(“MAN”) + vector(“WOMAN”)。**





二、静态词嵌入生成模型

分布式表示

由词嵌入方法生成的词向量里面，**各个维度的取值不再是0或1，而是一个连续的数字，而且词向量的维度也相较于独热法大大降低。这种方法生成的词向量常称为稠密的词向量（dense vectors）**，用以与独热法生成的稀疏词向量形成对照。在获得词语的词向量表示以后，通常可以使用两向量之间的余弦距离来度量两个向量之间的语义相似度，假设V和W表示两个词语的词向量，则它们之间的距离可以表示为：$\cos(V,W)=(V \cdot W)/|V||W|$ 。**两个词语对应的词向量之间的余弦值越高，表示两个词语之间的语义相似度就越高。**

词嵌入的概念

词嵌入旨在通过训练，将词语转化为维度有限的词向量，这一过程称为“嵌入”（embedding）。**一般经过词嵌入生成的向量，维度在50到1000之间。**这种稠密的表示方法在自然语言处理中，相较于独热法往往能取得更好的效果。下面介绍T.Mikolov在2013年提出的两种词嵌入经典方法，即CBOW（continuous bag-of-words）模型和Skip-gram模型。这两种模型训练起来相对简单，而且能够获得较好的效果，自两种方法提出以来就成为使用最多的词嵌入模型之一。

词嵌入的原理

为了说明两个模型的原理，假设有如下句子：

```
... lemon,   a [tablespoon of apricot jam,       a] pinch ...
                 c1          c2    w     c3         c4
```

在该句子中，定义目标词为W=apricot，目标词也称为中心词。假设词W对应的上下文的预测窗口为2，即上下文中包含的单词为c1=tablespoon，c2=of，c3=jam，c4=a。**CBOW模型旨在通过上下文的词（即c1，c2，c3，c4）来预测中心词W，而Skip-gram模型旨在通过中心词W来预测其上下文的词。**下面将具体介绍两种模型。

01

CBOW模 型

CBOW模型

假设整个词典中所有词所构成的空间为V，|V|表示的是所有词语的个数。假设某句话中第t个词用x_t表示。在初始化时，每个x_t通过one-hot进行编码，即每一个x_t都是一个$|V| \times 1$的向量。由于每一个词语都有两个身份，即中心词和背景词，其中背景词表示该词出现在其他中心词的上下文中。**CBOW模型希望根据上下文来预测中心词**，因此假设每个词语x_t都有两个向量——输入向量v和输出向量u，其中输入向量即x_t作为背景词对应的向量，而输出向量即x_t作为中心词对应的向量。设置n为词向量的维度，因此这两个向量均为$n \times 1$的向量，n的取值一般在50到1000之间。将词典中所有词对应的向量排列起来，就构成了**输入矩阵**$\boldsymbol{v}_{n\times|\mathbf{V}|}$**和输出矩阵**$\boldsymbol{u}_{|\mathbf{V}|\times n}$。

CBOW模型

CBOW模型中需要根据上下文中的词来预测中心词。**假设在一段话中的中心词为x_t，上下文的预测窗口长度为m，如上例中$m=2$，因此模型的输入其实是$2m$个词对应的向量（包括中心词上文m个词和下文m个词）**，具体表示为：$(x_{t-m},\dots,x_{t-1},x_{t+1},\dots,x_{t+m})$。输入层中每个词对应的one-hot向量与整个输入矩阵$\mathcal{V}_{n\times|V|}$相乘，即得到输入层中每个词的词向量，即$(v_{t-m},\dots v_{t+m})$。进一步将这些输入的词向量求平均，获得**投影层向量**：

$$\hat{v}_t=\frac{v_{t-m}+\cdots+v_{t+m}}{2m}\in\mathbb{R}^n$$

将投影层向量左乘输出矩阵$\mathcal{U}_{|V|\times n}$，获得**得分向量**$z=\mathcal{U}\hat{v}\in\mathbb{R}^{|V|}$。假设$z=(z_1,z_2,\dots,z_{|V|})'$。最后对向量$z$进行softmax变换，即$\hat{y}_i=e^{z_i}/\sum_{k=1}^{|V|}e^{z_k}$，最终得到**输出向量**$\hat{y}=(\hat{y}_1,\hat{y}_2,\dots,\hat{y}_{|V|})'$，该输出向量各个维度求和为1，**这个输出向量表示在输入为$(x_{t-m},\dots,x_{t-1},x_{t+1},\dots,x_{t+m})$的情况下，各个词语出现在其中心位置的概率。CBOW模型希望根据$\hat{y}$可以准确预测出真实的中心词x_t。**

CBOW模型

CBOW模型中需要根据上下文中的词来预测中心词。**假设在一段话中的中心词为x_t，上下文的预测窗口长度为m，如上例中$m=2$，因此模型的输入其实是$2m$个词对应的向量（包括中心词上文m个词和下文m个词）**，具体表示为：$(x_{t-m},\dots,x_{t-1},x_{t+1},\dots,x_{t+m})$。输入层中每个词对应的one-hot向量与整个输入矩阵$\mathcal{V}_{n\times|V|}$相乘，即得到输入层中每个词的词向量，即$(v_{t-m},\dots v_{t+m})$。进一步将这些输入的词向量求平均，获得**投影层向量**：

$$\hat{v}_t=\frac{v_{t-m}+\cdots+v_{t+m}}{2m}\in\mathbb{R}^n$$

将投影层向量左乘输出矩阵$\mathcal{U}_{|V|\times n}$，获得**得分向量**$z=\mathcal{U}\hat{v}\in\mathbb{R}^{|V|}$。假设$z=(z_1,z_2,\dots,z_{|V|})'$。最后对向量$z$进行softmax变换，即$\hat{y}_i=e^{z_i}/\sum_{k=1}^{|V|}e^{z_k}$，最终得到**输出向量**$\hat{y}=(\hat{y}_1,\hat{y}_2,\dots,\hat{y}_{|V|})'$，该输出向量各个维度求和为1，**这个输出向量表示在输入为$(x_{t-m},\dots,x_{t-1},x_{t+1},\dots,x_{t+m})$的情况下，各个词语出现在其中心位置的概率。CBOW模型希望根据$\hat{y}$可以准确预测出真实的中心词x_t。**

CBOW模型

上文讨论了CBOW模型的原理，那么如何进行模型估计呢？如果已知输入矩阵$\mathcal{V}_{n\times|V|}$和输出矩阵$\mathcal{U}_{|V|\times n}$，那么$\hat{y}\in\mathbb{R}^{|V|}$便可以很顺利的通过输入$(x_{t-m},\dots,x_{t-1},x_{t+1},\dots,x_{t+m})$计算出来。于是，**如何得到$\mathcal{V}_{n\times|V|}$和$\mathcal{U}_{|V|\times n}$就是模型训练的重点**。假设$y$表示模型输出，即中心词$x_t$对应的真实的one-hot向量。**因此CBOW模型的目标是使$\hat{y}$尽可能与真实情况y接近，深度学习模型中经常使用交叉熵来作为损失函数**，即：

$$\mathrm{H}(\hat{y},y)=-\sum_{j=1}^{|V|}y_j\ln(\hat{y}_j)$$

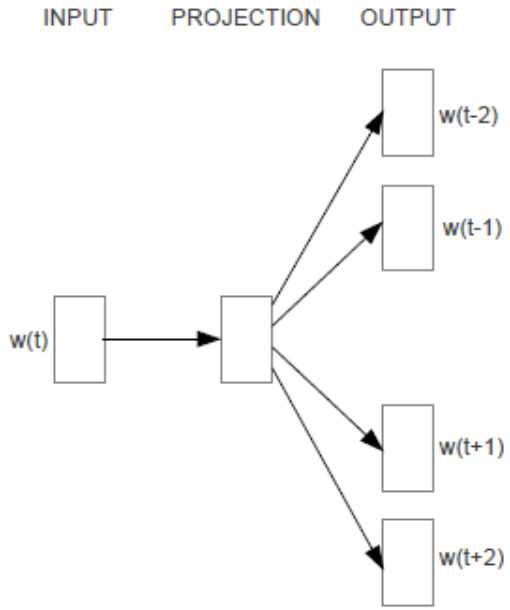
由于y是一个one-hot向量，于是上述交叉熵可以简化为$\mathrm{H}(\hat{y},y)=-\ln(\hat{y}_c)$，其中c表示真实中心词$x_t$在词典中的位置，即$x_t$对应的one-hot向量中取值为1的地方。模型训练时通过不断优化交叉熵损失来更新$\mathcal{V}_{n\times|V|}$和$\mathcal{U}_{|V|\times n}$。

02

Skip-gram模 型

Skip-gram模型

CBOW模型通过输入上下文的词向量，来预测中心词。而Skip-gram模型通过输入中心词，来预测其上下文可能出现的词语。Skip-gram模型的原理由下图所示，假设w(t)为中心词，预测窗口为2，那么**该模型是通过输入w(t)对应的向量来预测上下文的词，即w(t-2)，w(t-1)，w(t+1)，w(t+2)**。



Skip-gram模型

Skip-gram模型与CBOW模型的结构和参数设置基本相同，下面介绍其具体的建模步骤：

1) 在Skip-gram模型中，输入层为中心词$x_t \in \mathbb{R}^{|V|}$，采用one-hot编码。

2) x_t通过左乘输入矩阵$\mathcal{V}_{n\times|V|}$，获得输入向量$v_t = \mathcal{V}x_t \in \mathbb{R}^n$。

3) 将该输入向量左乘输出矩阵$\mathcal{U}_{|V|\times n}$，获得得分向量$z_t = \mathcal{U}v_t \in \mathbb{R}^{|V|}$。

4) 将该得分向量进行softmax变换得到$\hat{y}$，然后基于$\hat{y}$去估计中心词x_t对应的前后m个词。

Skip-gram模型

假设中心词x_t在词典中的位置为c，因此对应的输入词向量为v_c。给定中心词以后，**假设其前后m个词语之间是彼此不相关的，因此训练时的目标函数可以写成$2m$个交叉熵的求和，即：$f = \sum_{j=0,j\neq m}^{2m} H(\hat{y}, y_{c-m+j})$**。通过优化这一目标损失函数进行更新迭代。

值得注意的是，在Skip-gram模型的目标函数中需要对|V|个词的数值进行求和。由于|V|通常很大，在进行softmax计算的时候会导致**计算量非常大**。

Skip-gram模型

为了解决这一问题，T.Mikolov等（2013）在论文《Distributed Representations of Words and Phrases and their Compositionality》中详细介绍了基于Skip-gram模型提出的**“负采样”方法。负采样的核心在于构造一个新的目标函数，该目标函数使用了“抽样”的思想，即从真实存在的词语组合和不存在的词语组合中进行抽样，然后再进行训练**。由于负采样不需要进行传统的softmax变化，而只需要选取几个不在语料库中的词语组合作为“负样本”，从而减小了计算的成本。关于负采样的详细介绍，感兴趣的读者可以参阅T.Mikolov等（2013）。

Skip-gram模型

中国人民大学
RENMIN UNIVERSITY OF CHINA

CBOW模型和Skip-gram的模型差异主要在于**Skip-gram的训练时间更久，但是其往往具有更高的预测准确率，且对于生僻词会更加敏感**，使用者可以根据实际需求选择合适的模型。

三、BERT模型

BERT模型介绍

Vaswani等在2017年第31届NIPS会议的论文《Attention Is All You Need》上提出了一个不同于CNN和RNN（CNN和RNN详见第九章）的全新结构——**Transformer**，并提出了**注意力机制（Attention Mechanisms）**。注意力机制在机器翻译领域取得了非常好的效果。**Devlin等在2018年基于Transformer这一结构提出BERT模型，这被认为是深度学习模型在文本分析上的一个重大进展。**同时，**BERT模型也是对于词嵌入（Word2Vec）方法的进一步发展。使用BERT模型可以完成文本分类、文本问答、阅读理解等任务，且都显著提高了训练的精度，因此对BERT模型的应用占据了各类自然语言处理任务的榜首。**下面将简要介绍BERT模型。

BERT与WORD2VEC比较

- Word2Vec模型和BERT模型都是自然语言处理发展进程中的里程碑式的模型。两个模型的差异在于**从“静态”到“动态”**的提升以及充分考虑了词语的多层特性。2013年提出的**Word2Vec模型实际上是一个词袋模型（Bag-of-words model），并不能考虑词语所在句子中的位置等特征**。在模型训练完成以后，每一个词语的词向量就固定了（context free）。但是事实上，**词语在不同的位置、不同的语境下，其应该有不同的意义，所以用同一个词向量表示有失偏颇。**

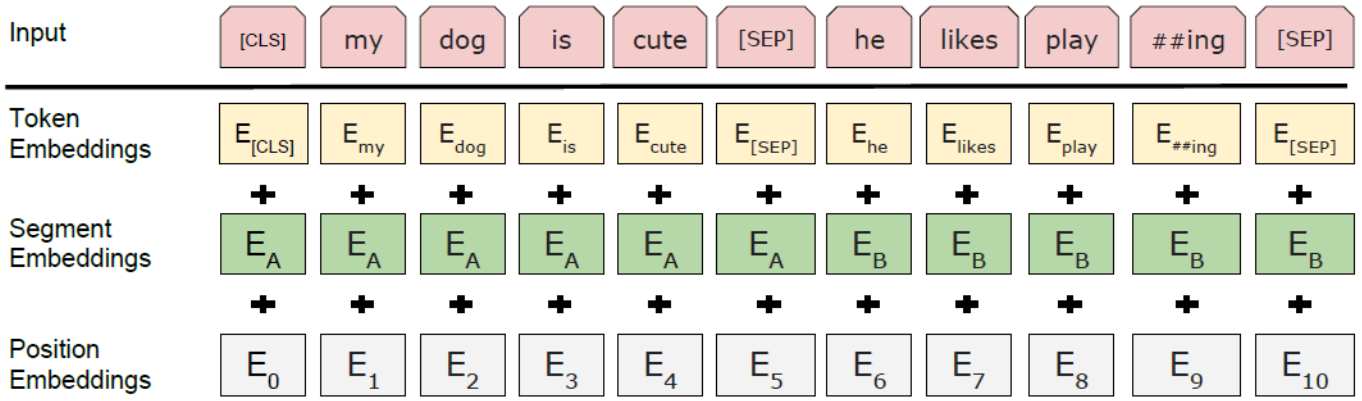
BERT模型特点

BERT模型是**一种与上下文有关的文本向量化方法**（contextual method）。

BERT模型与普通词嵌入模型的区别在于：普通词嵌入模型在训练好以后，输入一个词语就可以获得其对应的向量；但是BERT模型是需要给训练好的模型中输入目标语句，然后BERT模型基于这个句子的含义，给出这个特定句子下各个词语所对应的向量，由此可见，**BERT的词嵌入是与上下文相关的。**

BERT模型结构

BERT模型具体的嵌入层的结构如下图所示。从图中可以看出，嵌入层有**token embeddings**——表示的是原始文本中的词语对应的embedding；**segment embeddings**——表示的是这次词语所在句子对应的embedding；**position embeddings**——表示的是这个词语所在位置对应的embedding。通过这三个嵌入层的信息相加就实现了词嵌入与上下文相关。



BERT训练目标

BERT在预训练时主要有两个训练目标——

- 第一个是**MLM（Masked Language Model）训练**，具体训练方法如下：随机遮住15%的词语，将这一个序列输入Transformer的Encoder结构中，预测被遮住的15%的单词，这一个过程类似于四六级考试中的完形填空任务。通过不断训练增加预测的准确度。在这一个学习过程中，模型需要不断的学习某一个单词周围的语境信息。
- 第二个任务是**NSP任务（Next Sentense Prediction）**，用于判断两个语句是否互为上下文（二分类问题）这一任务。该任务用于发掘句子层面的逻辑关系。这一训练过程需要大量的语料和大量的内存来运行，因此一般使用的是已经预训练好的BERT模型。

BERT预训练

中国人民大學
RENMIN UNIVERSITY OF CHINA

BERT模型首先需要研究者完成预训练，例如Python包中就有各类语言预训练好的模型。网址https://github.com/google-research/bert上也公开了在多种语言（包括中文）上预训练好的BERT模型参数，研究者可以在此基础上进行进一步的训练 。在平时使用时，**只需要调用预训练好的BERT模型，输入语句，模型即可返回词语向量化的结果。**对于某一种特定的文本分析任务，研究者也可以在其特定的研究主题上对BERT模型进行再次训练。

BERT实际运用

在具体的使用中，BERT模型往往是作为一个大型自然语言处理任务的向量化预处理部分，它可以用于语言理解、翻译、情感分析等领域。在大部分时候表现都明显优于Word2Vec模型，在各个自然语言处理大赛中，拔得头筹的处理方法往往都会出现BERT及其衍生模型的身影。

四、代码实现

01

R代码实现

Word2Vec包

使用R进行词嵌入的训练需要使用**R包word2vec**，该包的核心函数为**word2vec**，该函数的具体形式为：word2vec(x, type=c("cbow", "skip-gram"), dim=50, stopwords=character(),split=c())。其中，x表示输入文本，type可选项为cbow和skip-gram，分别对应两种词嵌入方法；dim为向量化以后向量的维度，默认值为50，stopwords是指一些不纳入建模的停用词；split是标识分割两个词语的标志，常用的有空格、斜杠等等。在英文中常用空格分隔，而在中文中往往需要进行分词，然后再分词结果中用斜杠作为分隔符。此外，参数hs表示模型训练的方法。**需要注意的是，在R中没有专门实现BERT的程序包，因此BERT的实现将在Python中进行介绍。**

Word2Vec包

下面对和讯网新闻数据集进行词嵌入建模，选择词向量的维度是50。首先将中文文本进行分词，然后使用CBOW和Skip-gram方法训练词嵌入模型，具体代码如下：

```
# 承接第三章的内容，定义res为使用jiebaR分词包进行分词之后的结果。下面需要将分词结果进行重新组织
# 首先，初始化一系列的字符串，用于对jiebaR分词以后的结果进行重新组织
strs <- rep("", length(res))
# 使用循环语句来处理jiebaR分词以后的结果
for(i in 1:length(res)) {
  # 在每个循环中，初始化一个字符串，用于承接每一段话的分词结果
  str <- ""
  for (j in 1:length(res[[i]])) {
    # 词语与词语之间用空格分隔
    str <- paste(str, res[[i]][j], sep = " ")
  }
  # 将这一段话重新组织的结果赋值给strs变量
  strs[i] <- str
}
```

处理分词结果

下面对和讯网新闻数据集进行词嵌入建模，选择词向量的维度是50。首先将中文文本进行分词，然后使用CBOW和Skip-gram方法训练词嵌入模型，具体代码如下：

```
# 承接第三章的内容，定义res为使用jiebaR分词包进行分词之后的结果。下面需要将分词结果进行重新组织
# 首先，初始化一系列的字符串，用于对jiebaR分词以后的结果进行重新组织
strs <- rep("", length(res))
# 使用循环语句来处理jiebaR分词以后的结果
for(i in 1:length(res)) {
  # 在每个循环中，初始化一个字符串，用于承接每一段话的分词结果
  str <- ""
  for (j in 1:length(res[[i]])) {
    # 词语与词语之间用空格分隔
    str <- paste(str, res[[i]][j], sep = " ")
  }
  # 将这一段话重新组织的结果赋值给strs变量
  strs[i] <- str
}
```

Word2Vec包使用代码

中国人民大学
RENMIN UNIVERSITY OF CHINA

```
# 使用word2vec函数进行训练
# 其中，type表示选择的模型，可供选择的为"cbow"和"skip-gram"。
# dim表示的是词向量的维度。encoding表示文字的编码格式，
# 这里我们使用UTF-8对文字进行编码
model <- word2vec(strs, type = "skip-gram", dim = 50, encoding = "UTF-8")

# 通过训练好的模型来查看某一个词语最临近的若干个词语
# 其中，summary(model)[26001]就是“社会”这个词语
# top_n表示的是需要查看的词语数，这里我们查看与“社会”相似度最高的10个词语
predict(model, summary(model)[26001], top_n = 10)
```

两个模型下R代码结果

在上述代码中，变化word2vec函数中的type=“cbow”,即可得到CBOW模型的结果。下表展示了CBOW和Skip-gram两个模型下，和“社会”相似性最高的词语情况：

CBOW模型

term1	term2	similarity	rank
社会	深远	0.7397	1
社会	迫切需要	0.7363	2
社会	公信力	0.7353	3
社会	更多地	0.7352	4
社会	议题	0.7267	5
社会	良性循环	0.7124	6
社会	劳动	0.7119	7
社会	各行各业	0.7099	8
社会	各族群众	0.7088	9
社会	方方面面	0.7080	10

Skip-gram模型

term1	term2	similarity	rank
社会	全面进步	0.8563	1
社会	国民经济	0.8554	2
社会	慈善	0.8541	3
社会	生存权	0.8477	4
社会	弱势群体	0.8400	5
社会	迫切需要	0.8367	6
社会	清尘	0.8323	7
社会	基本权利	0.8282	8
社会	慈善事业	0.8275	9
社会	社会工作	0.8203	10

02

BERT的实现

借助Python工具

中国人民大学
RENMIN UNIVERSITY OF CHINA

BERT模型获得词向量实例

在深度学习领域，通过BERT模型进行词嵌入往往是文本分析的第一步，在获得这一词嵌入结果以后，研究者将会使用词向量来完成一系列下游任务。

使用Python中的**transformers包**可以调用训练好的BERT模型完成词嵌入。首先要调用相关程序包，从transformers包中导入BertTokenizer和BertModel两个函数，前者是一个针对BERT模型的分词器，或者是已经训练好的BERT模型。

借助Python工具

中国人民大学
RENMIN UNIVERSITY OF CHINA

BERT模型获得词向量实例

BertTokenizer与普通的分词器略有不同：**使用BertTokenizer的分词结果都是包含在训练好的BERT模型中的词语，对于BERT模型中不存在的词语，BertTokenizer会以字符[UNK]代替，即Unknown。**另外，在英文词语的分词中，**BERT模型会拆解词语**，例如将“strawberry”拆解为“straw-”和“berry”。这是因为在进行BERT的预训练中，英文词汇量过多，故而通过分解复合词、分拆前后缀的方式能够有效的减少训练的时间。**同时在分词时，BertTokenizer会在两个句子之间添加字符[SEP]，作为两个句子的分隔。对于不同的句子，BERT模型会给出不同标签，从而实现基于上下文的动态嵌入。**

借助Python工具

BERT模型获得词向量实例

对和讯网的数据进行分析时，由于数据是中文的，可以在BertTokenizer和BertModel中使用bert-base-chinese参数。输入想要进行嵌入的句子，然后通过outputs查看结果。

```
# 导入程序包，torch包用于张量运算等
import torch
# 从transformers包中导入Bert分词器（BertTokenizer）和预训练好的Bert语言模型（BertModel）
from transformers import BertTokenizer, BertModel

# 初始化分词器（导入的是已经预训练过的BERT中文模型）
tokenizer = BertTokenizer.from_pretrained("bert-base-chinese")
# 初始化BERT模型（导入的是已经预训练过的BERT中文模型）
model = BertModel.from_pretrained("bert-base-chinese")

# 输入想要预处理的句子，返回的张量类型为pt（return_tensors="pt"）
inputs = tokenizer("词嵌入非常有意思", return_tensors = "pt")

# 输出，可以查看到每一个词语所对应的向量
## **是python的一种特殊运算字符，指的是讲字典数据（input）解开成为独立的元素作为形参
outputs = model(**inputs)
outputs
```

END