班级：计硕2003 姓名：肖波 学号：2001842

**基于BERT的情感分析模型**

1. 基于Transformer的词向量表示

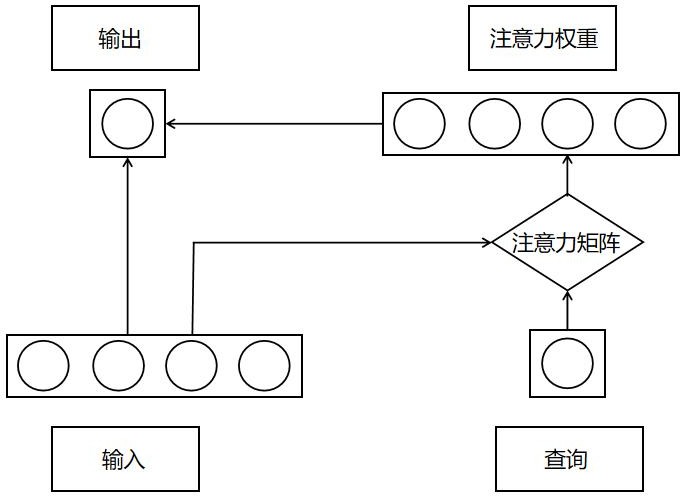
Word2vec 由词义的分布式假设出发，每一个单词被映射到一个唯一的稠密向量。这显然无法处理一词多义的问题：自然语言中每个词都有可能有多个不同的意思，那么如果需要用数值表示它的意思，或至少不应该是固定的某一向量。例如：

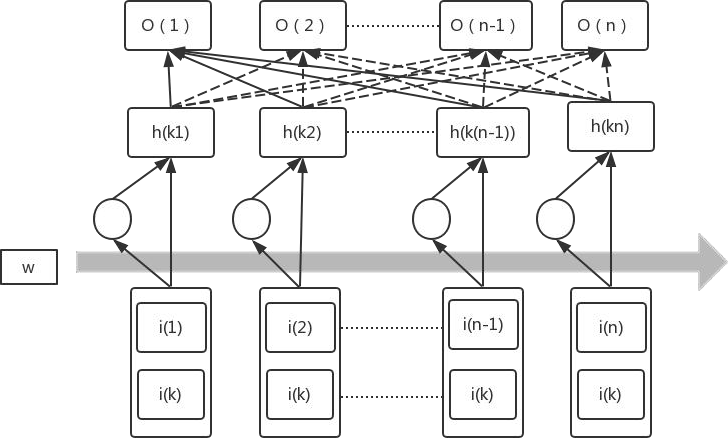
例 2： ①The animal didn’t cross the street because it was too tired.

②The animal didn’t cross the street because it was too narrow.

通过这两个句子可以看出，其中的 it 分别指代 animal 和 street，对于我们而言很容易区分其中的区别，然而对于计算机而言却并不容易。使用传统的 Word2vec 产生的词表示是静态的，不考虑上下文的，如果对两个 it 进行向量化表示，得到的词向量是一样的，这显然是有瑕疵的。

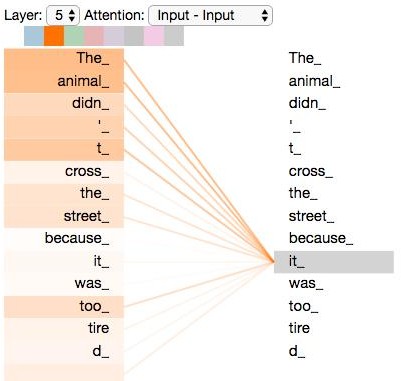
在介绍 Transformer 之前先来介绍一下注意力机制(Attention Mechanism)，注意力机制最早被应用于图像研究领域。它源于对人类大脑工作原理的分析，其本质上是注意力资源分配的模型。例如，当我们看图片时，我们的注意力肯定会集中在某个部分， 随着眼睛的移动，注意力又转移到图片的另一个部分。任何时刻我们的注意力分布是不一样的。Google 公司在 2017 年，将注意力机制定义为查询作用于键值的过程。注意力机制有三个主要元素：输入、查询和注意力矩阵。三者相结合，产生“注意力权重”，根据产生的权重就可以区分不同语境对目标词的重要性。文本情感分析领域内的注意力机制主要被归纳为三个模型：文本注意力、特征- 文本注意力与自注意力。这里主要介绍自注意力(self-attention)，self-attention 机制包含了不止一个特征，每个单词都依次与句子中所有的但那次相互作用。每一个单词都会通过这一系列的和计算得到一个其本身在该语境中的输出特征。所以 self-attention 的输出是多个单词的注意力特征的集合。



注意力机制结构示意图

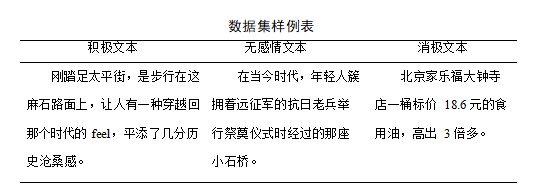
self-attention 结构示意图

Transformer 中引入了 self-attention 模块，让源序列和目标序列首先“自关联”起来，让计算机关注到更有价值的信息。Self-attention 机制简单来说就是一系列注意力权重系数，例如用 Transformer 对例 2 中①中的 it 进行编码，则需要对其上下文中的每一个词分配一个注意力权重值，计算时根据权重系数，就可以得到融合上下文语境的 it 编码。

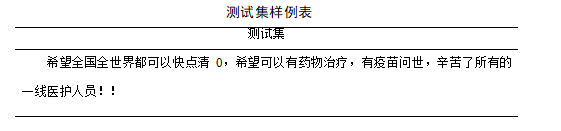


各个词对 it 编码影响程度示意图

1. 数据收集及预处理
2. 数据集

数据集为由山西大学提供的“拓尔思杯”中文隐式情感分析评测比赛数据集。其中训练数据集包括篇章 12664 篇，其中标注数据 14774 句，褒义、贬义隐式情感句分别为 3828、3957 句，不含情感句为 6989 句。验证集包括篇章 4391 篇， 其中标注数据 5143 句，褒义、贬义隐式情感句分别为 1232、1358 句，不含情感句为2553 句。测试数据集包括篇章 6380 篇，其中标注数据 3800 句，褒义、贬义隐式情感句为 919 和 979 句，不含情感句为 1902 句。

同时还在微博上爬取了关于“新冠疫情”等微博的一些评论 3973 条做为测试集。



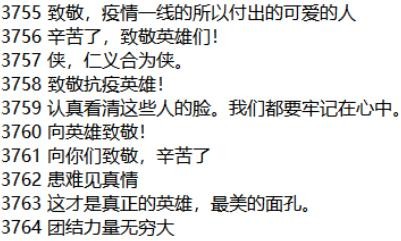
2.文本预处理

文本预处理是非常重要的一环，数据的质量对实验结果有着至关重要的影响。所以要通过文本预处理，对数据进行优化。文本数据预处理包括数据清洗、文本分词、过滤停用词等。

（1）数据清洗

数据清洗主要针对从微博上爬取的测试集。爬取的初始数据中包含许多特殊符号、网页解析符号等。利用Python 中的正则表达式来对测试集中的噪声进行清理。



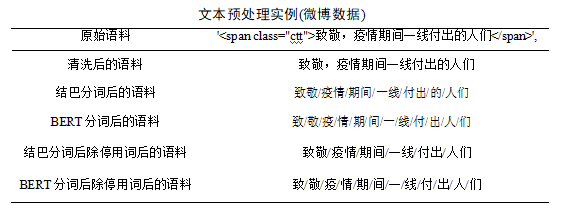


（2）文本分词

对于中文文本数据来说，每个字是最小的单元。同时，对文本的分词处理也十分重要。在本文中我们采取把两种分词方式，第一种是用结巴分词对一句话进行分词处理，把用结巴分词处理输入到 LSTM 网络中。在 BERT 模型中使用其自带的分词工具，把一句话的每个字拆分成一个单元。

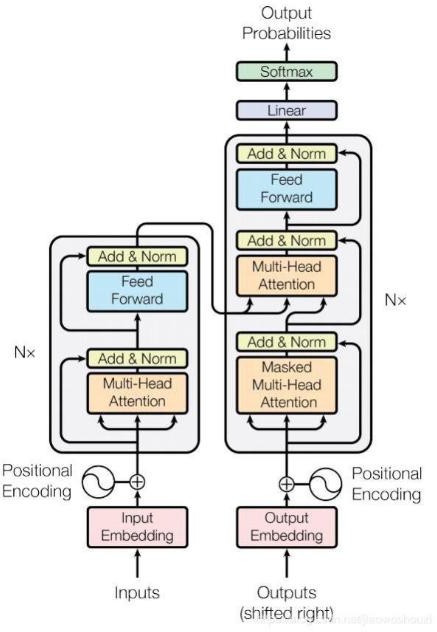
（3）过滤停用词

停用词指的是在文本中，没有明显意义的词。但是由于语言的习惯，经常在句子或短语中重复出现，常见的是一些连接词、介词、语气词等，如“哦”、“啊”、“额”等等。研究表明，情感模型的效果会受这些停用词的干扰。所以我们要把这些词经常出现，但是却没有什么实际含义停用词进行剔除。如表 3.3 所示，为本文测试集微博数据清晰的一个实例。



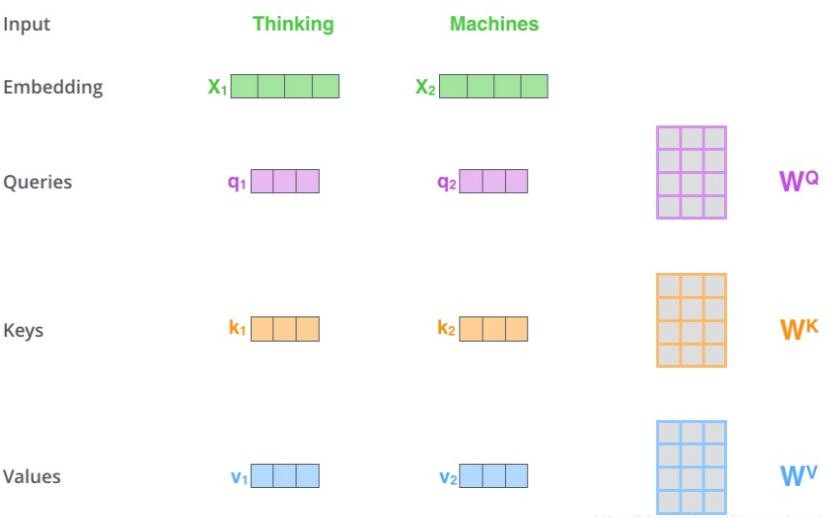
1. 基于BERT的情感分析模型

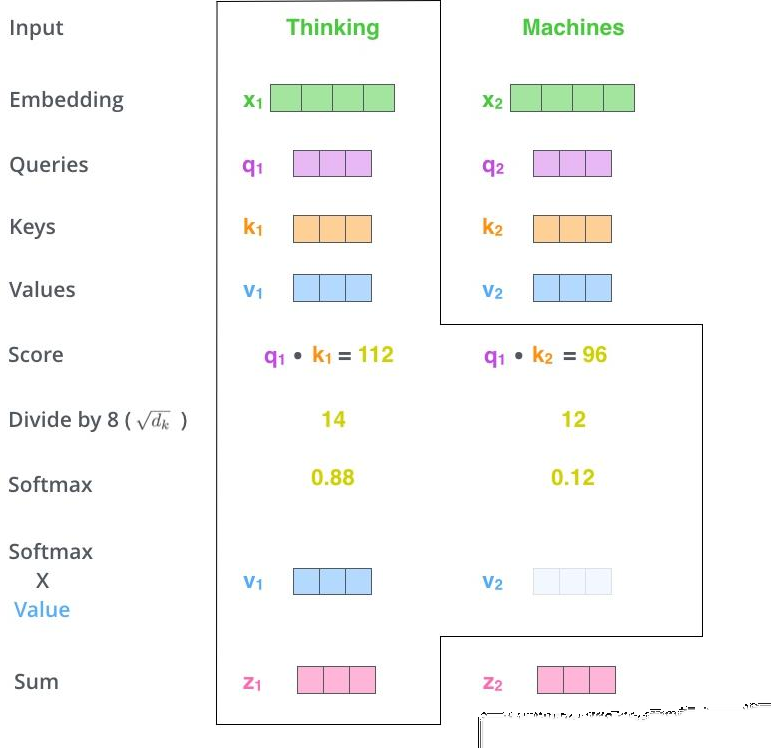
BERT 模型使用的是多层 Transformer 结构，其最大的特点是抛弃了传统的 CNN和 RNN，通过 Attention 机制有效的解决了文本处理中长期依赖问题。



接下来逐个分析其组成部分。

* + - 1. 自注意力(Self-Attention)：首先，自注意力计算三个新的向量，这三个向量分别称为 *Query*，*Key* 和 *Value*。这三个向量是通过将自动偏移向量和随机初始化矩阵相乘而产生的，并在反向传播过程中不断更新。接下来，计算自注意的分数值，该分数值可通过 *Query* 和 *Key* 向量计算得出。然后用所得结果除以一个常数。然后对所有得分进行 softmax 计算，结果表示当前位置单词和每个单词的相关性程度。最后，将softmax 所得结果与 *Value* 向量相乘并求和。结果是当前节点处的自注意值。

*Query*、*Key*、*Value* 矩阵示意图



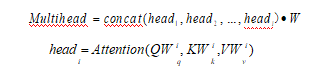
*Query*、*Key*、*Value* 计算过程示意图

*Query*、*Key*、*Value* 计算过程示意图

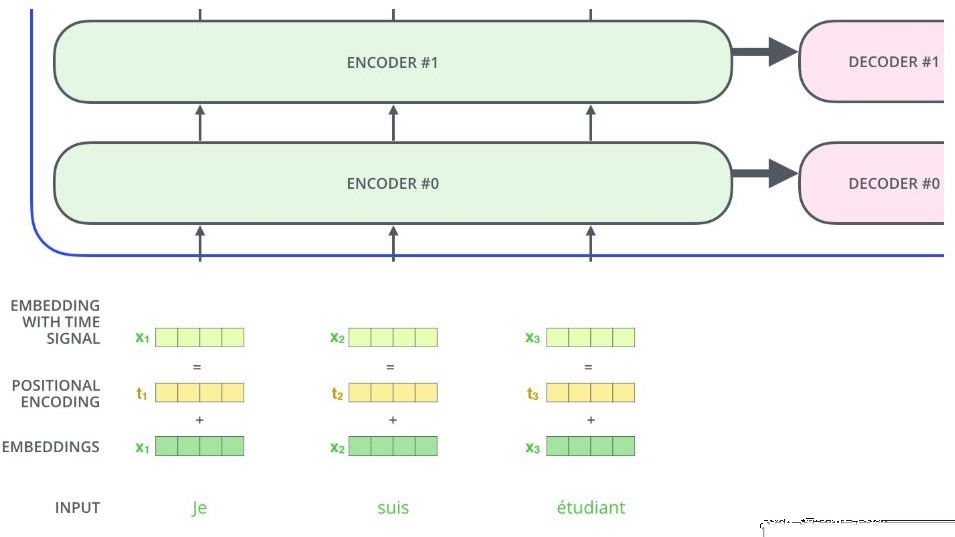
计算中 dk 等于 64，为惩罚因子，为保证 Q 和 K 的内积不至于过大。



* + - 1. 多头注意力机制：BERT 还给自注意力机制加入了另外一个机制，被称为“多头注意力机制”该机制就是不仅仅只初始化一组 *Q*、*K*、*V* 的矩阵，而是初始化多组， 用来获取句子级别的语义信息。



* + - 1. 位置编码：目前为止，变换器模型中输入数据的单词顺序问题还没有解决。正常来说，两个相关距离相近的词关联性一般比相对较远的大，所以把位置编码融入到词向量的计算中是非常必要的。因此，在编码器和解码器层的输入端添加了矢量位置编码。

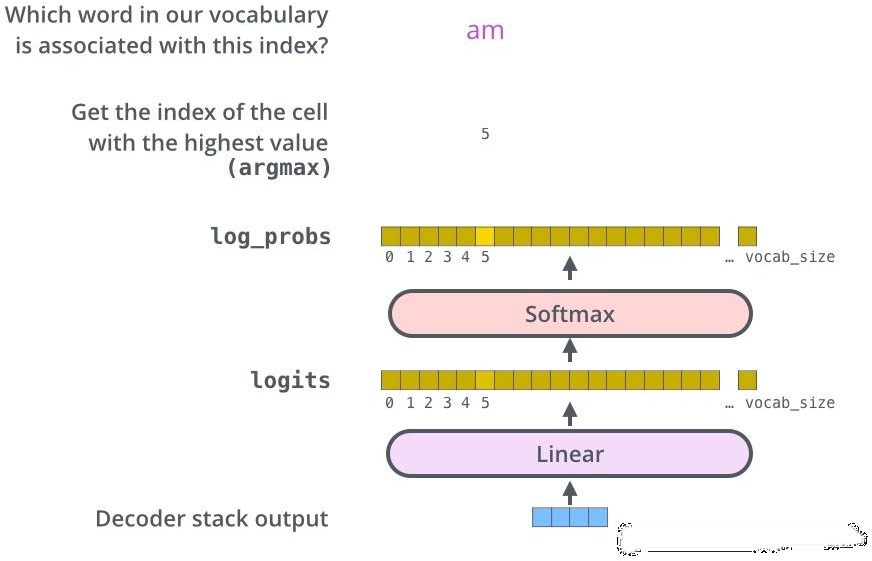


矢量位置编码示意图

* + - 1. 残差模块：在 transformer 中，为了使模型速度更快，每一个子层之后都会接一个残差模块，并且有一个 Layer normalization。同时为了增加模型的可表达能力， 会添加一个 Feed Forward 层。



* + - 1. 解码器：通过观察变换器的结构图可以发现解码器结构和编码器结构非常相似。其中 *Mask* 表示屏蔽某些值的掩码，*Mask* 为 0，则在更新参数时它们不起作用。
      2. 输出层：在解码层之后，在最后添加一个全连接层和一个 softmax 层，将词汇表中的单词分配给产生的词向量。然后，softmax 层将得分值转换为概率（所有正数，所有加起来都为 1.0）。

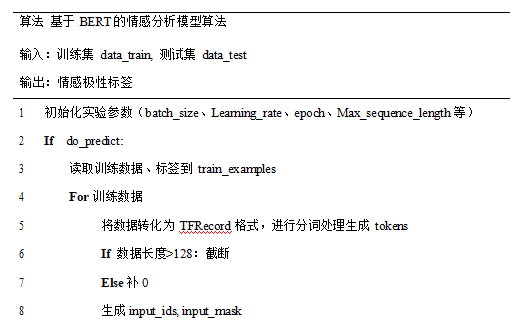
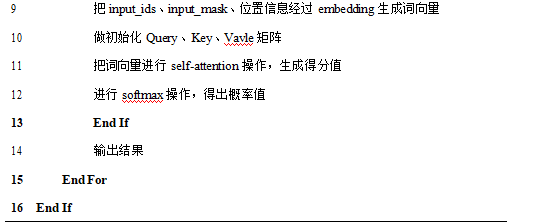


输出层示意图

在基于 BERT 的情感分析模型中，首先要对数据进行预处理。算法中第 1 行至第 8 行做的都是预处理工作。首先把数据转化为 TFRecord 格式，这样有利于 TensorFlow 进行计算。*tokens* 是分词处理之后的结果，BERT 模型中选择把每个字作为一个单位进行分词。因为在计算中要保持数据输入长度一致，所以对长度超过128 的句子进行截断，对不满 128 长度的句子进行补零。*inputs\_ids* 表示每个词对应的数字 id，这是根据词汇表来进行对应的，把文字转化为数字 id 有利计算。当我们补 0 时，在进行self-attention 操作时，我们是希望补的0 不参与运算的，所以其中*input\_mask* 的作用就是表示长度为 128 的数据之中有多少个数据是用有用的，补的 0 对应的*input\_mask* 就是 0，是不参与运算的。

使用了 Google 发布的预训练模型“chinese\_L-12\_H-768\_A-12”。该模型包括 12 层 Transformer，隐藏维度为 768，Multi-head Attention 的参数为 12，模型的参数总大小为 393MB。加载此预训练模型后，输入 *inputs\_ids* 即可获得训练好的 768 维词向量*word embedding*。此外，BERT 模型之中还添加了维度为768 的位置信息*position embedding*。*word embedding* 与 *position embedding* 相加，就得到了含有位置信息的词向量 *embedding*，维度为[8,128,768]，其中 8 表示每次处理 8 条文本，128 表示每条文本的最大长度。

接下来对词向量进行self-attention 操作，把前面得到的[8,128,768]维的 *embedding* 作为输入，分别构建维度为[8,128,768]的 Q、K、V 矩阵，对其进行公式(4.11)的计算得到一个新的[8,128,768]维的矩阵 *embedding2*，*embedding2* 作为下一层的输入，重复计算过程。因为本文的 BERT 模型含有 12 层 Transformer 结构，所以此过程重复 12 次，最后得到的即为自注意的分数值，对所有的分数值进行 softmax 操作，即可求出每个单词与当前单词的相关性矩阵 R。相关性矩阵 R 和 V 做内积，即可获得经过self-attention 操作、含有上下文语义的维度为[8,768]特征向量，然后采用两个全连接层对其进行加权，将特征矩阵的数据大小变为 8×3，最后做一次 softmax 操作，得出文本属于每种情感极性的概率，输出结果。



1. 实验

1.评价指标

对于文本情感分析任务，正常的评价指标是正确率 Accuracy，正确率是预测正确的样本数和总的预测样本数的比值。但是单一的 Accuracy 有时也并不准确，例如做一个二分类任务，第一类和第二类数据的比值是 95:5，这时候神经网络预测的结果全为第一类，正确率也有 95%，但是此时神经网络不具备分类功能。所以评价指标不能光看正确率率。本文的评价指标选取正确率、召回率、F1 值。F1 值是正确率和召回率的加权调和。具体的计算方法如下：

1. *TP*、*FP*、*T*、*F*、*TN*、*FN*：*TP* 表示真实为正情感倾向，预测结果正确。*FP* 表示真实为正情感倾向，预测结果错误。*T* 表示真实为无情感倾向，预测结果正确。*F* 表示真实为无情感倾向，预测结果错误。*TN* 表示真实为负情感倾向，预测结果正确。*FN* 表示真实为负情感倾向，预测结果错误。

三分类混淆矩阵

|  |  |  |  |
| --- | --- | --- | --- |
| 预测  真实 | 预测为正情感 | 预测为无情感 | 预测为负情感 |
| 正情感 | *TP* | *FP* | *FP* |
| 无情感 | *F* | *T* | *F* |
| 负情感 | *FN* | *FN* | *TN* |

1. 正确率(Precision):预测正确的样本数和总样本数的比值。

Pr *ecision* 

*TP*  *T*  *TN*

*TP*  *FP*  *T*  *F*  *TN*  *FN*

1. 召回率(Recall):预测正确的样本数和该类样本数的比值。

*R*ecall 

*TP TP*  *FP*

1. F1 值：F1 是正确率和召回率的加权计算调和，我们是期望正确率和召回率都是越高越好，但是二者是此消彼长的关系。所以我们要计算 F1 的值，来找到一个最优的评价方式。

2.实验结果

利用正向情感、无情感、负向情感倾向性1万多语料训练语言模型，迭代3次。在剩余的3000多测试集上进行验证。准确率达到81.2%，召回率为76%，F1值为78.5%。