# 零

# 一、语言

## 1.1 python

## 1.2 C++

## 1.3 常用数据结构

# 二.AI

## 2.1 机器学习

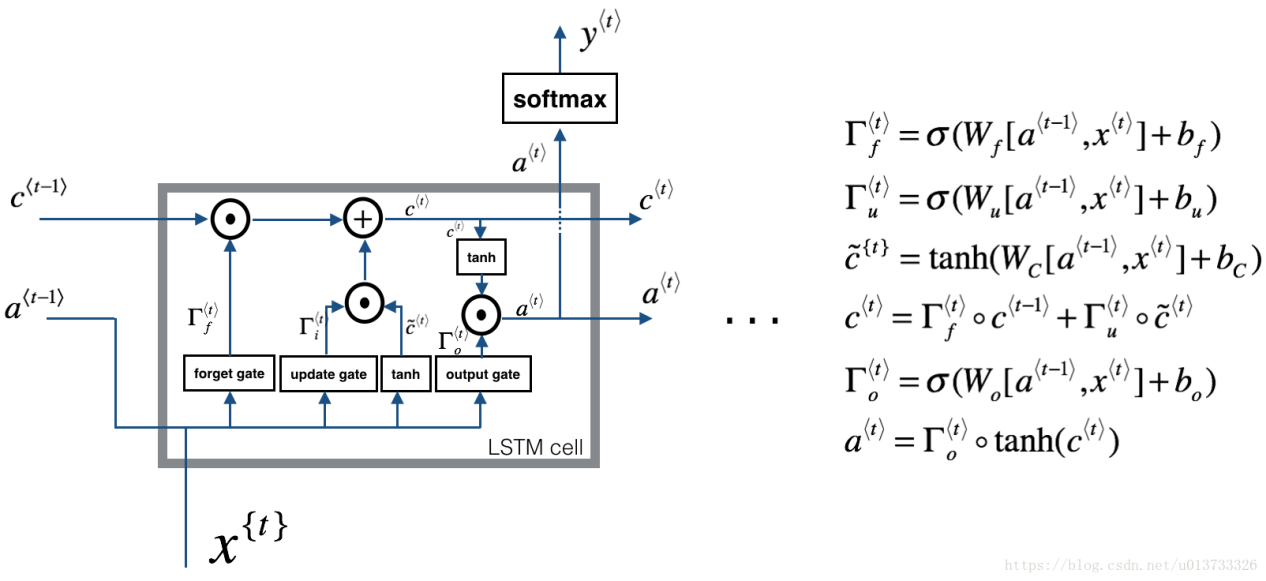
## 2.2、深度学习

### 2.2.1 基础模型

#### 2.2.1.1 循环神经网络

##### 2.2.1.1.1 RNN

##### 2.2.1.1.2 LSTM-三个门控单元



其中，空心点表示Hadamard乘积(同阶矩阵，对应元素相乘)，[]表示矩阵/向量相乘。

其实LSTM的思路很清晰：2条记忆线、一个输入、一个输出。

**历史信号**：进来后，与遗忘门相乘决定多大程度采用历史信息。

**记忆单元**：用双曲激活当前输入后，和更新门相乘再和记忆相加→Ct。

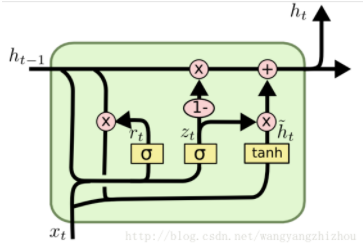
**输出信号**：将选择的历史信息和当前记忆单元求和，然后和输出门相乘决定最后的输出信号。3个门的输入都是a<t-1>和xt。

1、普通的RNN之所以不能捕获长期记忆，是因为W连乘问题导致梯度消失。

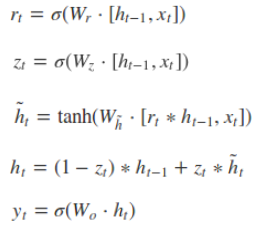
2、LSTM和普通RNN的相比之所以能缓解梯度问题，本质上讲在于信号在不同的时间步之间传播时，其数值不在取决于一个单一的矩阵，而是将数值问题的压力分散到更多的矩阵，抽象一层来讲就是由于门控机制的引入，使得理论上而言当遗忘门数值接近1的时候可以顺利的将历史信息引入并传递下去。

PS：关于激活函数的选取：门控值采用sigmoid函数，候选记忆采用双曲正切。原因解释：为了符合门控的物理定义有必要采取“开关函数”，而选用tanh在于大多场景下特征分布以0为中心，而tanh在0附近有较大的梯度，有利于参数的快速收敛。

##### 2.2.1.1.3 GRU-两个门控单元



没有标明的地方，数据均是往右/下流动，重画#Todo



其中\*表示Hadamard乘积，.[]表示矩阵/向量乘积，关于GRU的维度：首先隐层会和r、z做Hadamard乘积，所以必定以隐层的维度为目标，故而在GRU内部，Tensor均转化为Hidden Size，理论上有3个矩阵，工程实现上只有两个矩阵，因为显然前两个可以合并。

对于GRU而言，input size namely the size of Xt don't have to equal to hidden size,cause matrix Wwill change the input size to hidden size

out\_put also have the same dimmention with hidden size

in short hidden size is everything

python 文件命名以字母开头，如果以数字开头的话无法import

对于单层GRU而言：

# 输入：seq\_len，batch\_size，input\_size

# 隐层：layers, batch\_size，hidden\_size

# 输出：seq\_len，batch\_size，hidden\_size

# 隐层：layers, batch\_size，hidden\_size

##### 2.2.1.1.4 Esim文本匹配利器

1、编码-通过Bi-Lstm

2、局部推断-attention

3、推断融合-再经过Bi-Lstm

4、池化推断-将各个时间步的向量进行平均池/最大池，然后拼接成一个向量。

5、预测

##### 2.2.1.1.5 Dssm

根据余弦相似度来做文本匹配问题。

输入--MLP--特征向量--余弦距离

#### 2.2.1.2 卷积神经网络

#### #### 归一化

##### ##### Batch Normnization

##### ##### Layer Normalization

##### ##### Drop Out

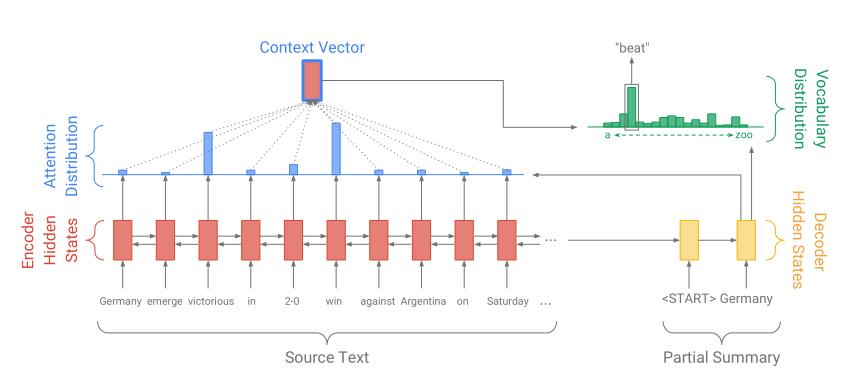
使得部分神经元参与参数的更新，相当于bagging的效果

测试的时候全部激活，但是计算值乘以衰减因子keep\_prob

BN与dropOut都具有正则化的效果，但是一般情况下，两者是不兼容的，一种解释是说网络状态在转换的时候会造成方差的偏移。

### ### NLP

#### #### 摘要模型



Baseline sequence-to-sequence model with attention

基线：encoder侧的隐层h作为k、v，然后decoder侧的隐层s作为q，基于weight sum求出语境向量context vector，将context与st concate之后送入全连接层，求出预测的单词。

##### #### 文献阅读

###### ###### 2019 综述：《基于序列到序列模型的生成式文本摘要研究综述》-安徽财经大学

基线：ABS attention based summarization

改进方案：

**编码侧：--隐向量的处理**

**选择性编码**：对每个step的隐层向量乘以一个衰减系数，系数由语境向量和h向量经过sigmoid生成，其实也是一种attention，只不多不是用解码查询，而是用语境向量查询。

**全局编码**：类似于ESIM的局部推断。

**层级编码**：又可以分为：句子、篇章、段落

句子：用句子的权重分布 更新单词的权重分布

段落：更精细的编码方案

**抽取辅助**：获取关键句的编码信息、基于ROUGE强化训练、关键词抽取[textRank]，然后获取关键词的向量，添加到attention机制中求注意力分布/**直接拼接用于预测**。

**事实抽取**：利用开源信息抽取工具OpenIE和句法句法分析工具[dependency parser]抽取原文的事实，然后分别抽取原文编码和事实编码。

**卷积编码**：ConvS2S 模型多层CNN，+ 残差 + 编解码attention

其他：添加语义特征：NER/POS/TFIDF

**解码侧：**

拷贝机制

重复控制

额外信息：变分模型获取隐层向量(类似于结构/摘要)、抽取实体构成主题向量用于辅助解码。

beam搜索

**训练：**

由最大化词项概率-->最小化摘要距离

基于ROUHE构建强化训练，结果是分数高了，但是可读性降低，所以后来又混合。

focalLoss

**总结：**

一般使用gru/LSTM取代RNN，gru结构更加简单，但是lstm的表达能力更强，所以一般短文本用GRU，长文本用LSTM。

卷积/多层LSTM多层网络可以构建原文本的分布特征、捕捉长距离依赖，但是训练时间更长。所以目前还是单层的比较多。

长文本摘要中，1/3采用了抽取辅助，添加知识驱动、先验知识可以帮助模型更好的理解文本。

未来方向：注意力、先验知识、辅助信息

###### ###### 2017 Get To The Point: Summarization with Pointer-Generator Networks-Stanford University-指针网络

**论文摘要：**

当前sequence2sequence modle 存在的问题与解决办法：

问题1:复制原文不准确的信息

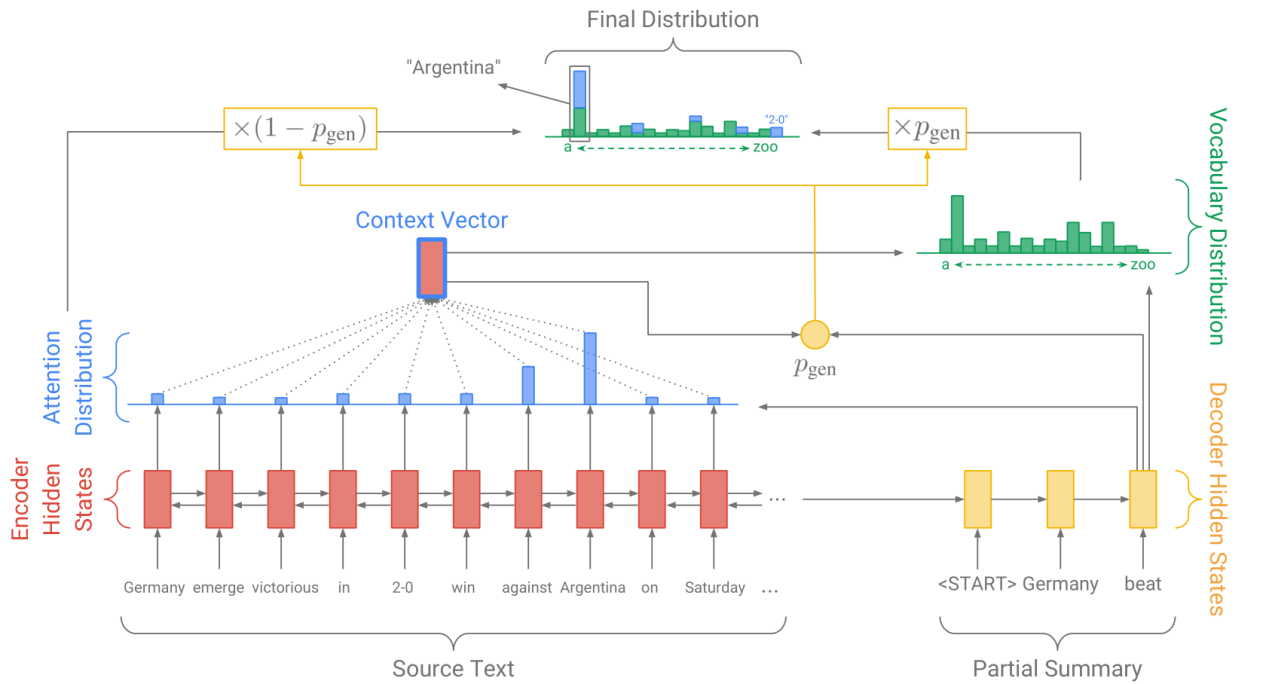
问题2:生成词自己重复

解决方案：pointer-generator network

1、pointer net：从原文复制使得能够准确的复现原文关键信息；另外generate net使得网络具有生成新词的能力；

2、利用coverage追踪已经生成的词语来缓解摘要重复问题。(从机器翻译引入)

**模型：Pointer-Generator-Model**



Pointer-generator model

attention参数和语境向量的计算和baseline一样，开关函数的公式为：



其中ht\*表示语境向量，st表示解码器输出，xt表示解码器输入，其他参数可以学习。每一个词语的生成概率公式为：



Pvocab表示全连接层输出的词表中每一个词的概率，ai为每个单词的attention得分。所以未登录词也会有得分。

覆盖机制：为了缓解解码时候的自我复制问题，借鉴MT(机器翻译)模型中的coverage机制，具体而言，在每一个解码时间步，维护一个覆盖权重：



attention机制也修改为：



其中h仍然是step的隐层向量，st为解码侧隐层向量，ct为覆盖向量。最后的分布权重仍然为at=softmax(et)。

损失函数修改为：



ct与at分别为历史总和与当前值，作为损失函数那么就希望覆盖向量变小，但是在每一个时间步at的和为softmax求和为定值1，所以对已经有很多注意力的位置不利。

**数据集**

CNN/Daily Mail Dataset：新闻内容平均781个词语，摘要平均56个词语，

287,226(28万)training pairs, 13,368(1万) validation pairs and 11,490(1万)test pairs

**实验**

1、并不是一开始就加上***覆盖机制***。一开始就假如覆盖机制的话会降低模型的表现。

2、只把覆盖向量添加到attention机制中，而不添加到损失函数中的话作用不大。

3、使用128维度的词向量，但是不是从头开始训练的，对比模型是从头开始训练的。

数据集：https://github.com/abisee/cnn-dailymail

python2代码：https://github.com/abisee/pointer-generator

python3代码：https://github.com/becxer/pointer-generator/

###### ###### 2019《基于改进Sequence-to-Sequence模型的文本摘要生成方法》-北京林业大学.周健

**背景：**

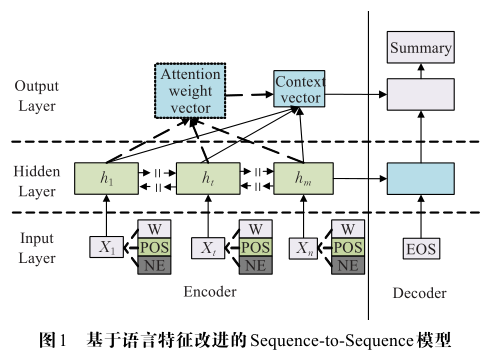
问题1：模型不能充分利用文本的语言特征信息； 对策1：加入语言特征

问题2：生成未登录词，影响阅读；对策2：利用copyNet缓解该问题

**什么是语言特征？**

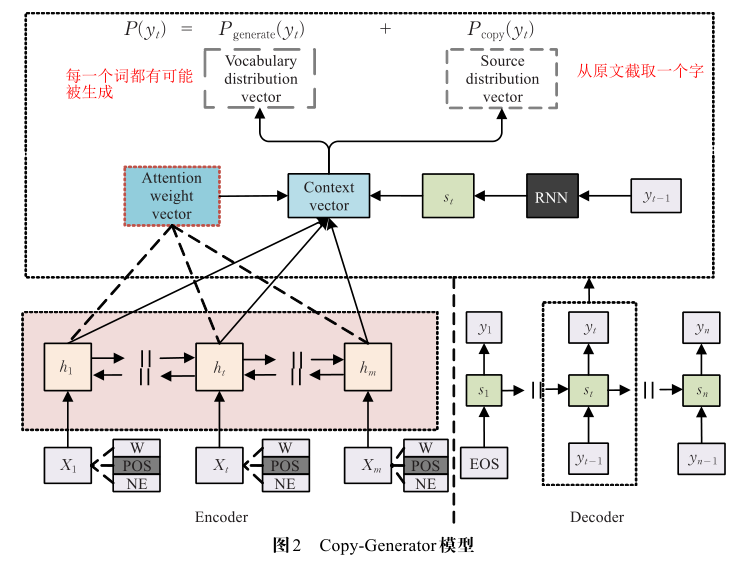
语言特征包括：词性、命名实体等，PS文本还具有统计特征包括tfidf、语义特征等等

**基本模型结构：**



Encoder和Decoder均采用RNN实现。首先在编码阶段，将词向量、词性类别向量、NE类别向量concate、然后就出每一个step的隐层向量。解码的时候，基本的做法是将encoder最后一个step作为隐层输入，然后逐个的生成摘要内容。引入attention的做法是：在每一个解码时间步，将解码隐层作为q，每一个编码隐层作为k、v，然后将两个向量concate过全连接求单词。

**copy-generator模型：**

****

在解码阶段，每个时间步的预测词yt可由两种方式产生：生成模式和拷贝模式。

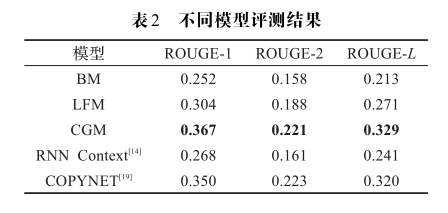
生成模式：基于encoder的隐藏状态和当前decoder步的隐层状态来求出预测词。

拷贝模式：利用两侧的注意力机制求内积，分数高的词被选中。

利用开关机制来决定模式选择：开关量的取值由编码隐向量、解码意向量

实验结果如何？和谁比较有提升？

数据集采用LCSTS，实验结果如下：其中BM表示编码和解码+attention模型，LFM表示BM+语言特征；CGM表示copy-generate model，CopyNet为经典模型。



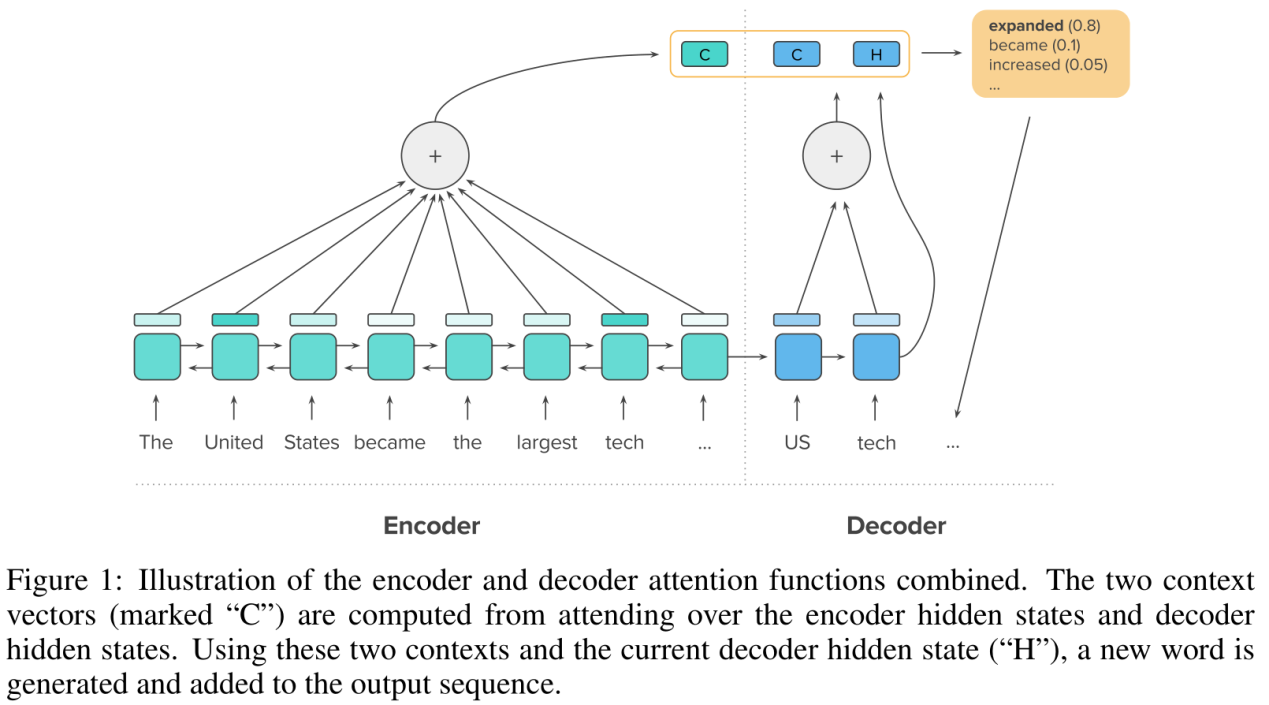
词向量表示方法？

word2vec向量，非关键词用特殊标记代替：（1）NAME\_EN — 英文标记；

（2）NUM — 数字标记；（3）DATE — 日期标志；（4）UNK — 未登录词；

（5）EOS — 文本结束标记

###### ###### 2017 A DEEP REINFORCED MODEL FOR ABSTRACTIVE SUMMARIZATION



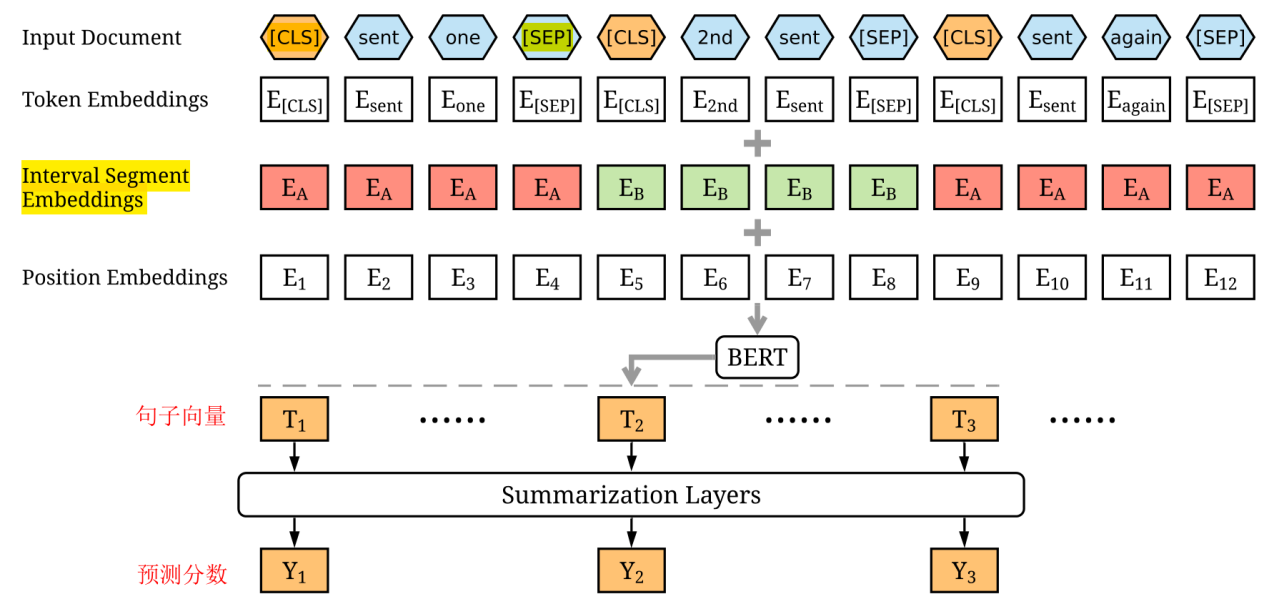
内注意力：除了解码对编码求attention，不同解码时间步之间求attention，然后连接两侧的atention以及当前step的h预测单词。

###### ###### 2019《Fine-tune BERT for Extractive Summarization》

摘要：我们要基于BERT模型做 抽取式摘要 模型，在CNN和NYT(纽约时报)两个数据集上评测结果显示效果最好[比较对象没有基于bert做的，要么是前三句，要么是指针]所以实验结果没有什么说服力。但是可以看看思路模型。

方法：将文章拆分为若干句，然后给每个句子打分为0/1，1表示要抽取这句话。

模型:

 **Encoding Multiple Sentences:**首先把文本分割为若干句话，并且用CLS和SEP框起来；

**Token Embeddings**：获取token 预训练embedding；

**Interval Segment Embeddings**：根据句子的奇偶序，分配为EA或者EB；

**Position Embeddings**：位置编码

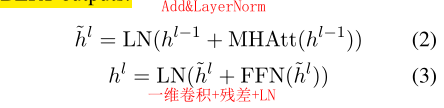
**Ti：**CLS位置对应的语境向量，有几句话就有几个语境向量；

**Yi：**sentence i的分数。

文中采用了3中方法求Y分数：直接对Ti向量打分、先过Transformer的编码结构[所谓的summarization layer]再打分、先过LSTM再打分。

疑问：segment直接标记为0或者1吗，为什么要用A、B标识？？？

**着重看下用transformer编码做摘要层的结构，因为这个效果最好：**

****

h0：就是bert输出，然后加上位置编码；

接着过两层编码(内部就是 多头attention、残差、LN、一维卷积)；

然后出来应该shape：句子个数，d\_model；

接着用sigmoid对每一句话打分。

**数据集：**由于数据集不是为抽取式模型设计的，所以label没办法直接使用，首先首贪心的方法，从原文的句子中选出和标准摘要Gouge分数最高的集合，以此为label。不太明白到底是怎么生成的。

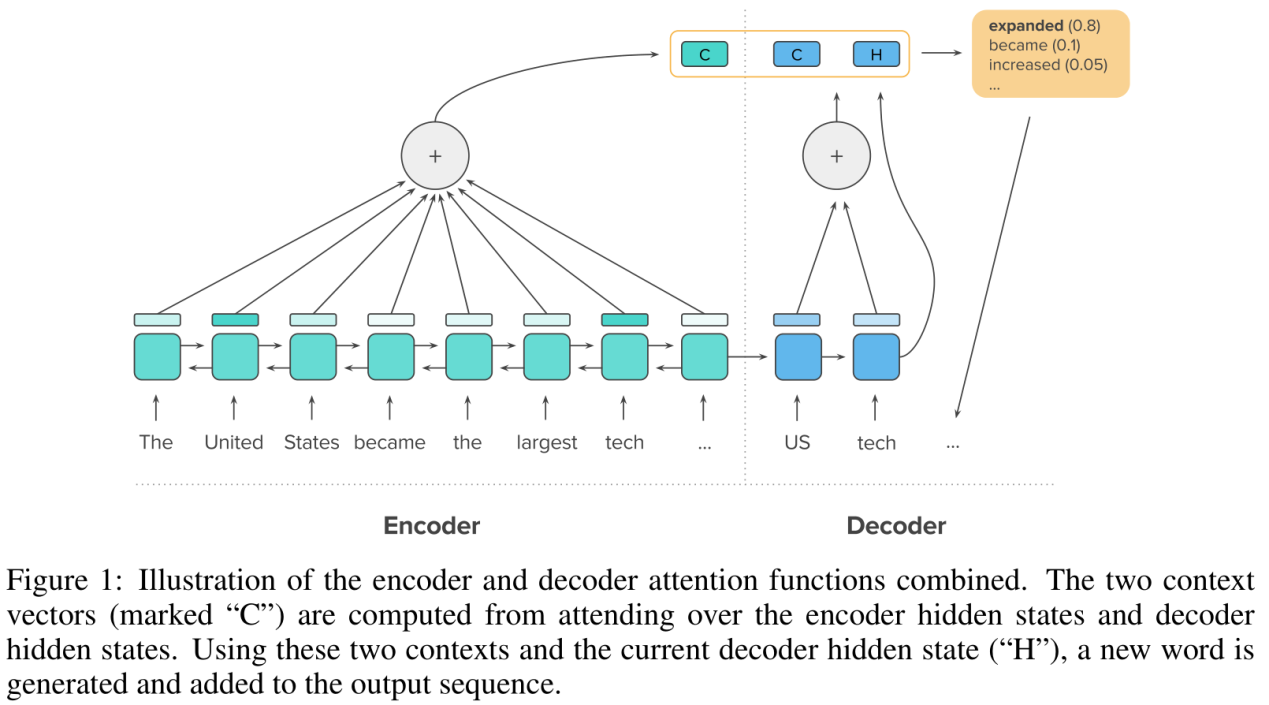
**优点**：为bert式摘要提供baseline

**缺点**:没有控制变量，应该和bert系列的照耀模型做对比，不然没有说服力。

问题：序列化函数是否必要?

#### #### 摘要创新point

更加精细的attention机制：正交分解/胶囊网络/句子编码用于解码attention。



强化学习

注意力、先验知识、辅助信息

POS分析、NER

熵？

2.3框架与常用工具包

### 2.3.1 pytorch

### 2.3.2 tensorflow

### 2.3.3 pandas

### 2.3.4 numpy

### 2.3.5 matplotlib