**机器学习Project2聊天机器人项目文档**

**上海交通大学 软件学院**

**组员：胡昊源 王浩宇 许嘉琦**

**一、聊天机器人资料**

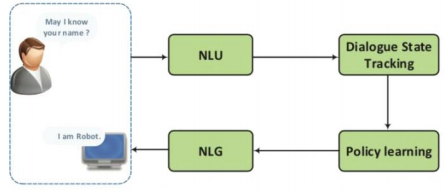
**1、背景简介**

现如今，我们可以很容易地访问网络上对话的 “大数据”，我们也许能够让机器学习如何回复，以及如何回复几乎任何的输入，这将极大地允许我们在人类和计算机之间建立数据驱动的、开放的对话系统。 另一方面，深度学习技术已经被证明是有效的，可以在大数据中捕获复杂的模式，并拥有大量的研究领域，如计算机视觉、自然语言处理和推荐系统等。具体来说，对话系统大致可分为两种:

1) 任务导向型 (task-oriented) 对话系统

2) 非任务导向型 (non-task-oriented) 对话系统 (也称为聊天机器人)。

面向任务的系统旨在帮助用户完成实际具体的任务，例如帮助用户找寻商品，预订酒店餐厅等。 非任务导向的对话系统与人类交互，提供合理的回复和娱乐消遣功能，通常情况下主 要集中在开放的领域与人交谈。 面向任务的系统的广泛应用的方法是将对话响应视为一条管道（pipeline)，如下图所示：



系统首先理解人类所传达的信息，将其作为一种内部状态，然后根据对话状态的策略采取一系列相应的行为，最后将动作转化为自然语言的表现形式。虽然语言理解是通过统计模型来处理的，但是大多数已经部署的对话系统仍然使用手 工的特性或手工制定的规则，用于状态和动作空间表示、意图检测和插槽填充。

非任务导向的系统似乎在进行聊天，但是它在许多实际应用程序中都发挥了作用 数据显示，在网上购物场景中，近80% 的话语是聊天信息，处理这些问题的方式与用户体验密切相关。

**2、非任务导向型对话系统**

一般来说，对于非任务导向型对话系统，目前用的主要是两种主要方法：

1. **生成方法**

例如序列到序列模型（seq2seq），在对话过程中产生合适的回复，生成型聊天机器人目前是研究界的一个热点，和检索型聊天机器人不同的是，它可以生成一种全新的回复，因此相对更为灵活，但它也有自身的缺点，比如有时候会出现语法错误，或者生成一些没有意义的回复；

1. **基于检索的方法**

从事先定义好的索引中进行搜索，学习从当前对话中选择回复。检索型方法的缺点在于它过于依赖数据质量，如果选用的数据质量欠佳，那就很有可能前功尽弃。

**3、生成模型与检索模型**

关于神经生成模型，我们来看看Sequence-to-Sequence Models，

他的目标是：

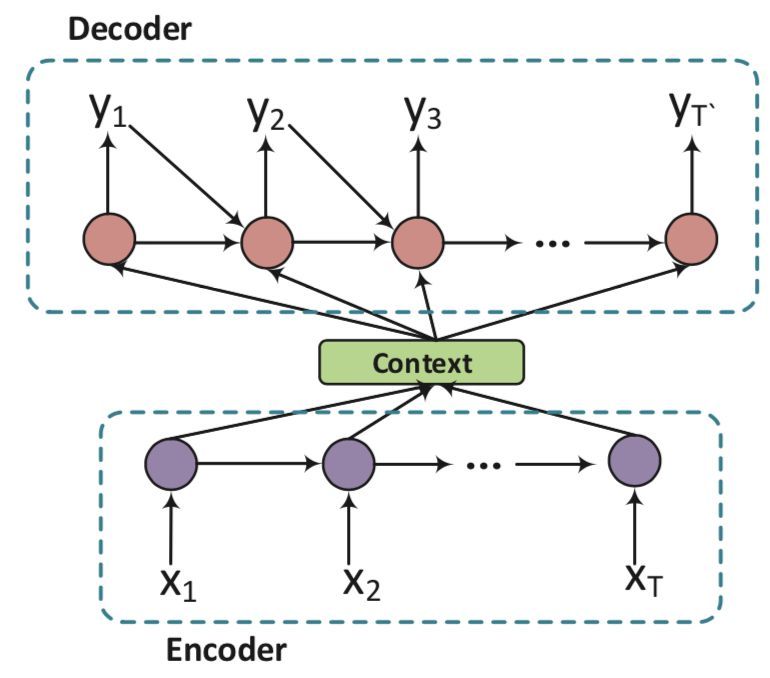
给定包含个词语的输入序列（message），

和长度为T的目标序列（response），

模型最大化Y在X下的条件概率。



具体来说，Seq2Seq模型是在encoder-decoder结构中,左图是这种结构的示意：



编码器将X逐字读入，并通过递归神经网络(RNN)将其表示为上下文向量c, 然后解码器将c 作为输入估计Y 的生成概率。

而检索模型所使用的回复数据通常是预先存储且事先定义的数据，而不像生成式模型那样可以创造出未知的回复内容。准确来说，检索式模型的输入是一段上下文内容，和一个可能作为回复的候选答案，模型的输出是对这个候选答案的打分。寻找最合适的回复内容的过程是：先对一堆候选答案进行打分及排序，最后选出分值最高的那个作为最终回复。

总的来说生成模型能够生成更合适的回复，而这些回复可能从来没有出现在语料库中，而基于检索的模型则具有信息充裕和响应流畅的优势。

**4、混合的方法**

将生成和检索方法结合起来能对系统性能起到显著的提升作用。

在集成模型中，被抽取的候选对象和原始消息一起被输入到基于RNN的回复生成器中。这种方法结合了检索和生成模型的优点，这在性能上具备很大的优势。

目前神经生成模型的热门研究课题大概有如下几个方面：

**A、对话上下文（Dialogue Context）**

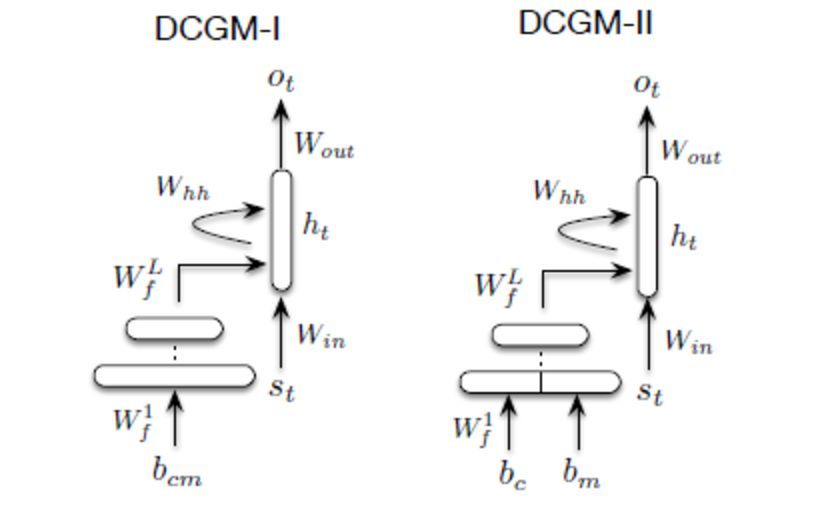
考虑对话的上下文信息的是构建对话系统的关键所在，它可以使对话保持连贯和增进用户体验。使用层次化的RNN模型，捕捉个体语句的意义，然后将其整合为完整的对话。

同时，分别用字级别和句子级别的注意力方法扩展层次化的结构。

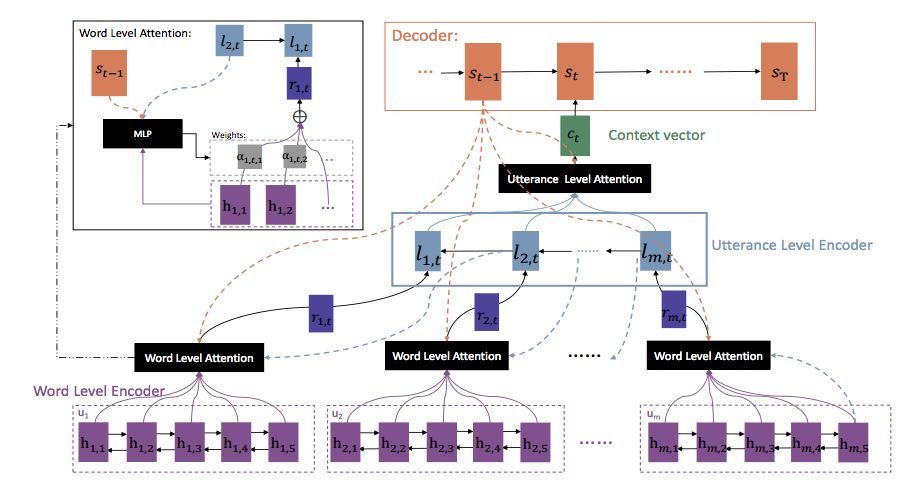
试验证明：

1) 层次化 RNNs的表现通常优于非层次化的RNNs；

2) 在考虑上下文相关的信息后，神经网络趋向于产生更长的、更有意义和多样性的回复。



在这个模型中，通过代表整个对话历史(包括当前的信息)，用连续的表示或嵌入单词和短语来解决上下文敏感回复生成的这一难题。

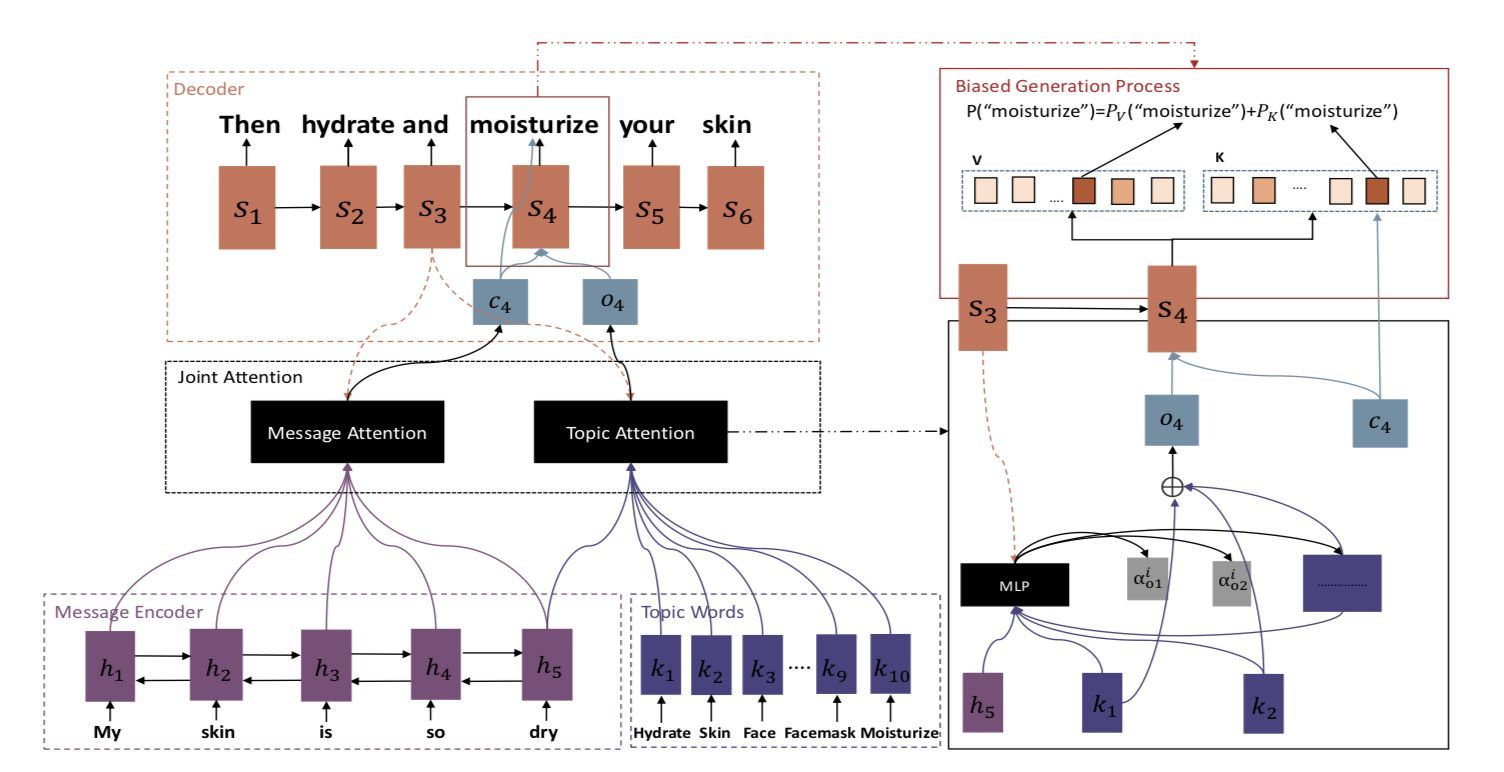


图中的结构中引入两个层次的Attention机制，让模型能够自动的学到词语与句子级别的重要度信息，从而更好的生成新一轮的对话。

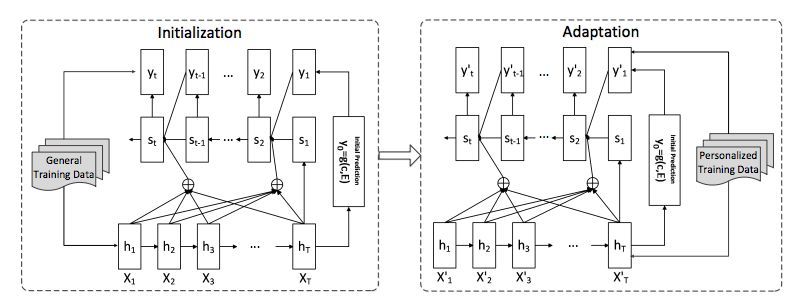
在句子级别的信息中，是反向学习的，即认为下一句的信息中更能够包含上一句的信息，所以从总体上来看，其对于对话的学习是逆向使用每一轮对话的内容的。

**B、主题和个性化（Topic and Personality）**

明确对话的内在属性是提高对话多样性和保证一致性的另一种方法。在不同的属性中，主题和个性被广泛地进行研究探讨。



在这个模型中，原作者注意到人们经常把他们的对话与主题相关的概念联系起来，并根据这些概念做出他们的回复。他们使用Twitter LDA模型来获取输入的主题，将主题信息和输入表示输入到一个联合注意模块中，并生成与主题相关的响应。



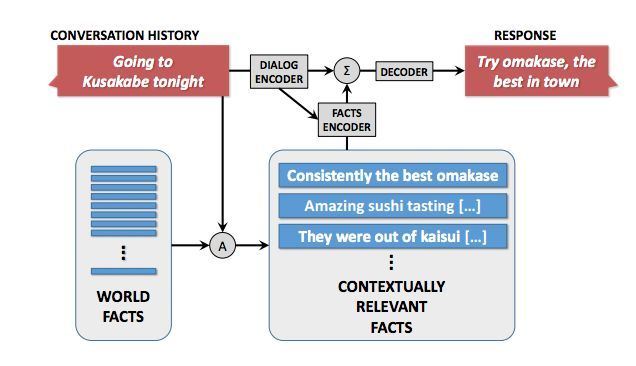
这个模型提出了一种两阶段的训练方法，使用大规模数据对模型进行初始化，然后对模型进行微调，生成个性化响应。

**C、外部知识库（Outside Knowledge Base）**

人类对话与对话系统之间的一个重要区别是它是否与现实相结合。结合外部知识库 (KB)是一种很有前途的方法，可以弥补背景知识之间的差距，即对话系统和人之间的差距。

**D、记忆网络（Memory Network）**

记忆网络（Memory Network）是一种以知识库处理问题的经典方法。因此，它非常直接的用在对话生成中。实际研究表明，所提出的模型能够通过参考知识库中的事实来生成对问题的自然和正确答案。



这个模型是一个完全数据驱动的带有知识的对话模型。其中的 World Facts是一个集合，收集一些经过权威认证过的句子或者不准确的句子，作为知识库。

当个定一个输入S和历史，需要在 Fact 集合里面检索相关的facts，这里采用的IR引擎进行检索，然后经过 Fact Encoder 进行 fact injection。

这个模型提出了一种全新的、完全由数据驱动的、基于知识的神经对话模型，目的是在没有槽位的情况下产生更多的内容。作者归纳了广泛使用的SEQ2SEQ方法，通过对会话历史和外部“事实”的响应 。

**5、面临的问题**

深度学习已成为对话系统的一项基本技术。研究人员将神经网络应用于传统任务导向型对话系统的不同组成部分，包括自然语言理解、自然语言生成、对话状态跟踪。近年来，端到端的框架不仅在非面向任务的聊天对话系统中流行，而且在面向任务的对话系统中逐步流行起来。深度学习能够利用大量的数据，从而模糊了任务导向型对话系统和非任务导向型对话系统之间的界限。值得注意的是，目前的端到端模型仍然远非完美。尽管取得了上述成就，但仍然有不少问题需要克服：

1. **快速适应**

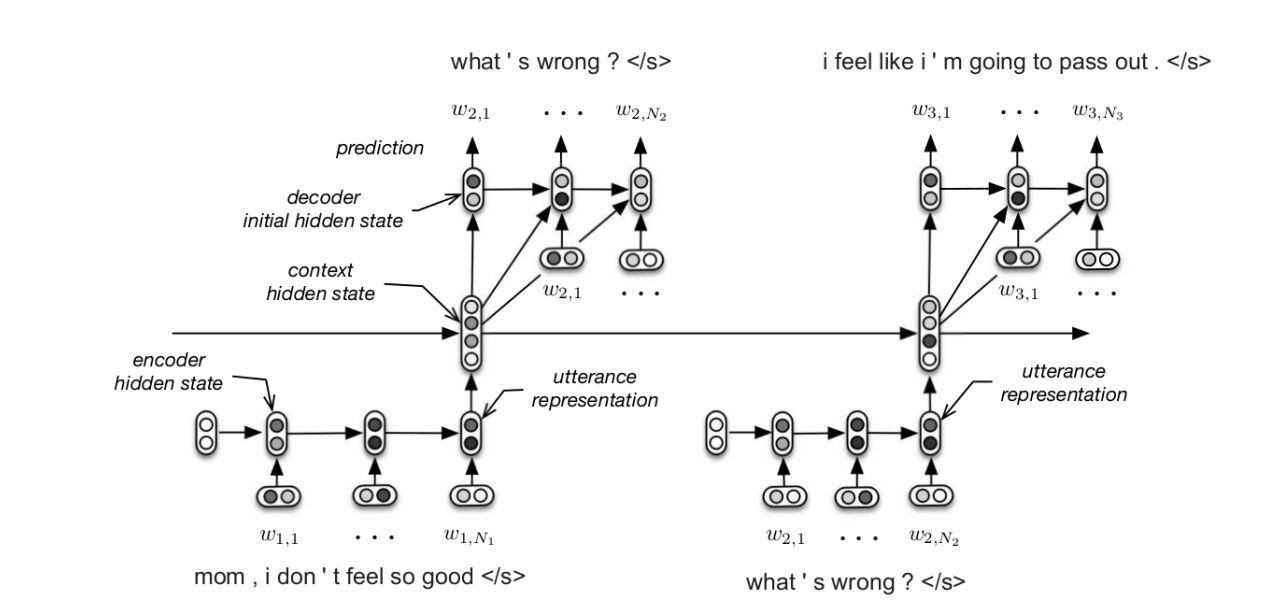
在一些新的领域，特定领域对话数据的收集和对话系统的构建是比较困难的。另外，有限的离线训练数据和真实的在线测试数据之间存在数据分布的差异性，会导致系统在线表现不佳，和离线测试效果不匹配，这种差异性主要来自于未见的用户行为（例如：新槽值、新意图、复杂句等）。例如，训练数据中从未出现过“喂，内容是什么？我忘了反馈什么了” 类似的用户语句，一旦真实上线，模型很容易预测出错误的结果，从而影响用户体验。尤其是当数据量少的时候，模型对于在线数据的鲁棒性会进一步下降。因此如何解决线上线下数据不匹配，提高端到端对话模型的鲁棒性是另一大挑战。通过调研我们发现，应对数据少的问题的常用方法有元学习、数据增强等，应对训练和测试对话数据不一致的问题的常用方法有人机协同、在线学习等。未来的趋势应该是对话模型有能力从与人的交互中主动去学习。

1. **深度理解**

现阶段基于神经网络的对话系统极大地依赖于大量标注好的数据，结构化的知识库以及对话语料数据。在某种意义上产生的回复仍然缺乏多样性，有时并没有太多的意义，因此对话系统必须能够更加有效地深度理解语言和真实世界。

解决这类问题的一个很有效的方法是找到并设置一个更好的目标函数。除此之外，解决此类问题的一个方法是增加模型的复杂度。

论文《Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models》使用了 latent variable 来解决无意义回复这个问题。



作者训练了分层递归编码器-解码器（HRED）网络来生成对话框。 “较低”级别将单词序列编码为一个“通”向量，而较高级别的编码器则使用这些思想向量来构建上下文的表示形式。作者使用困惑度量 MoviesTriples 数据集上评估了他们的模型，并获得了比纯RNN和DCGM模型更好的结果。使用大型问题解答语料库进行预培训可以大大减少困惑。

1. **评价困难**

评价生成回复的质量是对话系统的一个重要方面。任务导向型的对话系统可以基于人工生成的监督信号进行评估，例如任务完成测试或用户满意度评分等。然而，由于高回复的多样性，自动评估非任务导向的对话系统所产生的响应的质量仍然是一个悬而未决的问题。目前的方法有以下几种：

1）计算 BLEU 值，也就是直接计算 word overlap、ground truth和你生成的回复。由于一句话可能存在多种回复，因此从某些方面来看，BLEU 可能不太适用于对话评测。

2）计算 embedding的距离，这类方法分三种情况：直接相加求平均、先取绝对值再求平均和贪婪匹配。

3）衡量多样性，主要取决于 distinct-ngram 的数量和 entropy 值的大小。

4）进行图灵测试，用 retrieval 的 discriminator 来评价回复生成。

**D、 隐私保护**

目前广泛应用的对话系统服务于越来越多的人。很有必要注意到的事实是我们使用的是同一个对话助手。通过互动、理解和推理的学习能力，对话助手可以无意中隐蔽地存储一些较为敏感的信息。因此，在构建更好的对话机制时，保护用户的隐私是非常重要的。

**6、文献来源与总结**

我们主要翻看了两篇文献。

**A、第一篇《Attention is all you need》**

创新点在于抛弃了之前传统的encoder-decoder模型必须结合cnn或者rnn的固有模式，只用attention，可谓大道至简。文章的主要目的是在减少计算量和提高并行效率的同时不损害最终的实验结果，创新之处在于提出了两个新的Attention机制，分别叫做 Scaled Dot-Product Attention 和 Multi-Head Attention.

Model ： Scaled Dot-Product Attention 
Scaled Dot-Product Attention 
首 先 计 算 q 和 k 的 点 乘 ， 然 后 除 以 《 ， 经 过 些 巨 圣 得 到 v 上 的 
权 重 分 布 ， 最 后 通 过 点 乘 计 算 v 的 加 权 值 。 
在 实 际 中 为 了 并 行 计 算 ， 可 以 在 一 组 queries 上 计 算 注 意 力 函 
数 ， 将 多 个 qu “ y 堆 叠 成 同 理 keys 和 “ es 也 被 堆 叠 成 K 和 V ， 
通 过 下 曲 的 公 式 来 计 算 矩 阵 输 出 ： 
Attention(Q ， K00 = softmax( 

Model: Multi-Head Attention 
Scaled Attention 
Multi-HeadAttention 
论 文 提 出 对 queri ， keys 和 
“ ] ues 做 h 次 不 同 的 投 影 ， 映 射 的 
维 度 都 是 和 尹 ， 然 后 都 经 过 
Scaled Dot-Product Attention ， 
将 结 果 拼 接 在 一 起 ， 最 后 通 过 一 
个 线 性 映 射 输 出 ， 通 过 多 头 汴 意 
力 ， 模 型 能 够 获 得 不 同 子 空 间 下 
的 位 置 信 息 。 如 上 冬 右 边 所 示 ， 
公 式 如 下 ： 
A"凵眉艹@， 人 飞 0 = 

**B、第二篇《A Structured Self-attentive Sentence Embedding》**

文章的主要创新之处是使用矩阵来表示句子的embedding，矩阵中的每一行通过self-attention机制来表示提取句子不同的关键信息。目前word emebdding在词表示上已经非常的成功，但是在表示短语或者句子上，通常的做法是使用RNN的最后一个输出或者在CNN和RNN的中间输出上使用max和mean pooling操作。在一些任务中也可以使用attention机制通过引入额外的信息来指导句子embedding的生成。然而在一些任务上，比如语义情感分类，这类任务并没有可以利用的额外信息，模型的整体输入只有单一的句子，在这种情况下似乎只能通过上面所说的方法来获取句子的embedding。这篇文章提出一种self-attention的方法来替换max和mean pooling操作，文章中最让人眼前一亮的是通过attention将句子转化为多个vector来提取句子中不同的部分，用一个matrix来表示句子embedding。

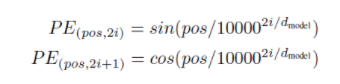
XPUJJJOS 
(q) 

**二、实现方法与细节**

**1、词向量化**

在 data/dailydialog 中已提供相应的词向量的映射，我们并不需要特别的实现，但究其原理，即将句子中的每一个词映射为一个向量，两者之间一一对应，比如一个长度为 512 的行向量。对于一个含有n个单词的句子，就是一个 n x 512 的矩阵，为了后续的计算方便，我们将n设为训练集中最长句子的长度。

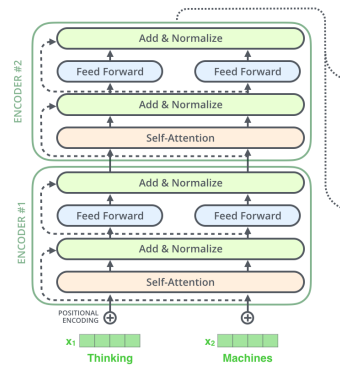
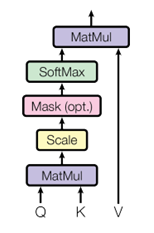
一个单词除了它本身的含义外，它在句子中的位置也同样会对翻译造成影响，而 attention 机制却没有办法直接考虑到单词之间的顺序（这也是 attention 可以并行的原因），如果对于一句话，改变其单词顺序对 Transformer 来说不会产生什么影响的话，这样显然是不行的，因此需要在词向量中加入位置信息，在这里采用的是位置编码的方法，计算方式如下，其中pos代表单词在句子中的位置，i代表该词向量的第i个维度，dmodel为512，与词向量的维度相同。



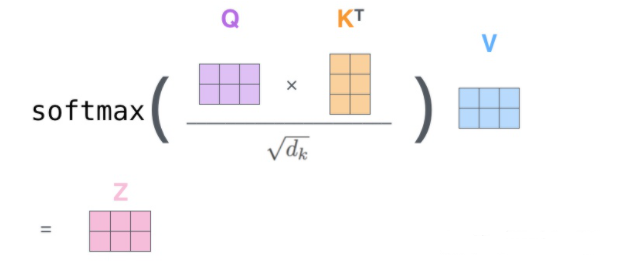
这一样就得到了关于位置信息的矩阵PE，可以看出来，该矩阵与具体的单词无关，对于相同长度的句子，其PE矩阵一样。将PE矩阵加到上述的矩阵中，就可以得到包含位置信息的句子矩阵。

**2、编码流程**

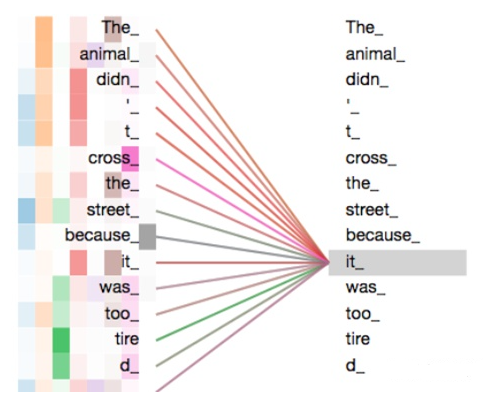
得到句子矩阵之后，我们将其送入编码器 Encoder 进行编码。这里，我们会用到 attention 策略，编码器 Encoder 整体流程如下：



将Query看作查询向量，Key和Value看作键值对，对于一个单词x1，想找到句子和它相关性比较大的单词，通过查询向量q1计算数据库中与其相关性较大的键值k，再以相关性大小作为权重对相应的Value求和，最后得到的结果就应该是对于x1而言影响最大的，这样做就使得原向量不再只包含单词本身，还包含了该单词与上下文之间的关系，这样通过多层编码就可以提取出该单词包含上下文后的特征。



不过我们还可以通过Multi-Head Attention来进一步提高性能：把上述的计算过程是看作一个head，那么可以同时构造出多个head，它们之间相互独立（Q、K和V均不同），最后输出的结果也是独立的，如下图所示。这样做的好处就是能够得到词与词之间的更多的关系，从而令翻译更加准确。



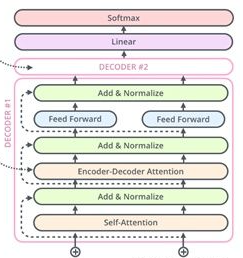
每一个head都输出一个结果矩阵，将这些矩阵拼接成一个矩阵，再乘以一个权重矩阵WO，使得最终的矩阵大小与一个head的结果矩阵大小一致。

**3、解码流程**

解码层的输入来自于两处，如下图所示，一部分输入来自于编码组件的输出，需要注意的是这部分是直接输入到解码器中的编码-解码attention子层中的，编码-解码attention中的Key和Value就来自于这部分输入；另一部分则是将模型已经翻译出来的部分句子重新输入到解码层的self-attention中，这被叫做auto-regressive性质，也因为这样的性质，解码器一次只能生成一个位置的输出。

对于self-attention子层，它与编码器中的self-attention基本一致，唯一不同的是解码器中的是Masked Multi-Head attention，这是为了防止在翻译中出现未来信息。在训练过程中，解码组件输入的是已知的输出（目标语言句子），为了防止未来的翻译结果对当前翻译的影响，模型在attention中使用了masking，将后面位置的单词与当前位置单词的相关性变为0，从而避免了未来翻译对当前单词的影响。这样当前翻译就只能够attention到已翻译的前文。

对于编码-解码attention子层，它和self-attention有所不同，它的Query来自于之前的解码层，而Key和Value则来自于编码层的输出，这样就可以使得解码器中的每一个位置都能attention到初始句子中所有位置的单词，使得模型能够得到当前翻译与编码器提取出的特征向量之间的关系。



**4、输出预测值**

通过解码组件可以得到一个向量，经过线性层将它投影成为一个长度很大的向量，这个向量的长度就是词典的大小，向量的每一个维度都代表目标语言单词库中的一个单词，对向量做softmax后，向量中该维度的值越大，代表对应单词的概率越大，输出概率最大的单词作为翻译结果。

