NLP

各种词向量预训练比较：https://zhuanlan.zhihu.com/p/75391062

word2vec

- 介绍下word2vec：1. 一类生成词向量的模型，2. 一个只含一个隐层的全连接神经网络，中间隐层是简单映射，无激活函数，输入层，中间一个无激活函数的隐含层，softmax输出层 3. 主要分为CBOW和skip-gram两种模型 4. 生成的向量具有语言信息。余弦相似度计算 5. 另外，由于softmax函数计算复杂度高，研究者提出两种加速方式：Negative Sampling和Hierarchical Softmax。
- skip gram和cbow的优缺点：1. CBOW模型中input是context（周围词）而output是中心词，训练过程中其实是在从output的loss学习周围词的信息也就是embedding，但是在中间层是average的，一共预测V(vocab size)次就够了 2. skipgram是用中心词预测周围词，预测的时候是一对word pair，等于对每一个中心词都有K个词作为output，对于一个词的预测有K次，所以能够更有效的从context中学习信息，但是总共预测K*V词。 3. skip gram的训练时间更长，CBOW更加的高效，但是对于一些出现频率不高的词，在CBOW中的学习效果就不如skipgram。总统来说skip gram可能好点，但差别不大。
- 负例采样和层次化softmax各自的优缺点：
- 介绍下Hierarchical Softmax，怎么更新参数：1. 词表中的全部词作为叶子节点，词频作为节点的权重，构建哈夫曼树，词频越高的词，距离根节点就越近。 2. 预测过程，从根节点出发，走到指定叶子节点 W 的过程，就是一个**进行 $L(w)-1$ 次二分类**的过程：路径上的每个非叶子节点都拥有两个孩子节点，从当前节点 n(w,j)向下走时共有两种选择，走到左孩子节点 $ch(n(w,j))$ 就定义为分类到了正类，走到右孩子节点就定义为分类到了负类。 3. 与普通Softmax相比，就是N 分类问题变成 log(N)次二分类。
- Negative Sampling：1. 把语料中的一个词串的中心词替换为别的词，构造语料 D 中不存在的词串作为负样本。然后优化目标变为最大化正样本的概率，同时最小化负样本的概率。
- 负采样具体怎么操作：
- word2vec的优缺点：优点1. 低维具有语义信息 2. 通用性比较强 缺点：1. 不能有效解决一次多义的问题，静态模型 2. 不能有效的解决未登录词 3.没有考虑全局的共现性，用到的context 较小
- 博文：https://zhuanlan.zhihu.com/p/33799633

Fasttext: 不错的博文，https://zhuanlan.zhihu.com/p/32965521

- 介绍fasttext：1.一个词向量计算和文本分类模型 2.模型结构与Cbow很像，输入词向量和n-gram字符特x征，叠加平均得到文档向量之后softmax分类出结果。
- 主要的创新点；1. 字符级n-gram的词特征向量输入，只用unigram的话会丢掉word order信息，所以通过加入N-gram features进行补充 ，用hashing来减少N-gram的存储2. 应用Hierarchical Softmax，类别数较多时，通过构建一个霍夫曼编码树来加速softmax layer的计算，和之前word2vec中的trick相同
- 与word2vec区别：1. word2vec的CBOW模型架构和fastText模型非常相似 2. 字符级n-gram可以很好解决未登录词，(计算subword的平均向量)，而word2vec无法解决未登录词

glove 博客https://zhuanlan.zhihu.com/p/42073620

- 介绍下glove：1.一种基于统计的词向量生成模型，计算共现矩阵到概率共现矩阵。2. 利用每三个词的概率共现的比率来计算（假设：任意三个词向量经过某种计算可以表现出和概率共现比率一样的的效果，让词向量之间蕴涵共现矩阵的信息）
- 与word2vec的区别。1. 损失函数的不同，一个交叉熵一个最小平方差损失。2. 一个是基于统计计数的词向量，一个是基于神经预测的词向量。3. 相比word2vec，收敛更快，训练时间更短。

N-Gram

- 介绍下N-gram：1. 是一种基于统计的语言模型，2. 计算方式符合马尔科夫假设，也就是每个词出现的概率只跟它前面的少数几个词有关，二阶马尔科夫假设只考虑前面两个词，相应的语言模型是三元模型（一次计算三个词）3. N-gram的应用有很多，包括评估语句是否合理，搜索引擎中预测用户搜索，输入法等等。在NLP中很多算法中用到n-gram作为句子序列的特征。

主题模型

- 介绍下LDA算法：1. 简单来说，LDA的目的就是要识别主题，即把文档—词汇矩阵变成文档—主题矩阵（分布）和主题—词汇矩阵（分布），2. 是一种文档主题生成模型，也称为一个三层贝叶斯概率模型，包含词、主题和文档三层结构。

seq2seq&attention

- 介绍下attention： 1. 假设当前时刻t下，我们有一个query向量，和一段key向量，这里query可以理解为一个包含比较多信息的全局向量，利用query对所有key向量相似度计算和对齐，然后对key对应的value值进行加权求和，权重就是相似度，学习到一个更合适的新向量去做分类或者预测等任务。2. attention主要有三种相似度计算方式：dot，矩阵乘，拼接tanh
- 介绍下不同种类的attention：1. hard 和soft的 2. local和global 3. 单头self-attention和muti-head attention 4. 基于memory的attention
- 博文：https://zhuanlan.zhihu.com/p/31547842
- **<手写attention的公式>**

参考Attention is All You Need中的说法，假设当前时刻t下，我们有一个query向量，和一段key向量，这里query可以理解为一个包含比较多信息的全局向量，我们利用这个query对所有key向量进行加权求和，学习到一个更合适的新向量去做分类或者预测等任务。

假设q_t就是时刻t下的query向量，K是key矩阵，k_s是其中一个key向量，V是value矩阵，我们先对q_t和每个key进行相似度计算得到一个非归一化的score分数：

$$s(q_t, k_s) = \frac{< q_t, k_s >}{\sqrt{d_k}}$$

这里用到是最简单的点乘，分母是为了调节内积结果，使得内积不那么大。

然后对score进行softmax归一化，作为attention概率权重：

$$a(q_t, k_s) = \frac{exp(s(q_t, k_s))}{\sum_{i=1}^{N} exp(s(q_t, k_i))}$$

最后我们对每个位置key所对应的权重和value进行加权求和，得到最终的输出向量：

$$Attention(q_t, K, V) = \sum_{s=1}^{m} a(q_t, k_s) v_s$$

可以看到，对同一段key，不同query会有不同的输出向量。

对应到具体任务上，可能会更加清晰一点：

在机器翻译任务中，query可以定义成decoder中某一步的hidden state，key是encoder中每一步的hidden state，我们用每一个query对所有key都做一个对齐，decoder每一步都会得到一个不一样的对齐向量。

在文本分类任务中，query可以定义成一个可学的随机变量（参数），key就是输入文本每一步的hidden state，通过加权求和得到句向量，再进行分类。

transformer

不错的博文：https://zhuanlan.zhihu.com/p/48508221；https://zhuanlan.zhihu.com/p/54743941

transformer问题整理：https://www.nowcoder.com/discuss/258321?type=post&order=time&pos=&page=1

- 介绍transformer：1. encoder-decoder的形式，encoder和decoder都是多层结构；encoder包括多头self-attention，位置编码，Add & Norm，全连接；decoedr也一样。2.
- 介绍多头self-attention的具体做法和作用：1. 得到不同的KVQ进行attention后全部拼接 2.考虑不同层次的信息，比如全局和局部
- 介绍位置编码：1.引入self-attention不足的不存在位置信息 2. 使用的是三角正弦和余弦函数编码相对位置
- 介绍Add & Norm：1. 残差链接和LayerNorm 2. 就是普通正规化的作用

- Transformer相比于RNN/LSTM/CNN，有什么优势？1. 最主要的相对于RNN，Transformer具有并行性，更加的高效 2. 相对CNN和RNN，一般来说语义抽取能力都要强一些
- 对Transformer-XL了解吗：1. 针对Transformer无法建模超过固定长度的依赖关系，对长文本编码效果差和Transformer把要处理的文本分割成等长的片段，通常不考虑句子（语义）边界，导致**上下文碎片化(context fragmentation)**的问题，Transformer-XL主要目的是赋予编码器捕获长距离依赖的能力。2. 实现方式，为了长距离依赖的建模和片段之间产生交互，解决上下文碎片化问题，提出片段级递归机制，引入一个**记忆(memory)**模块（类似于cache或cell），循环用来建模片段之间的联系。3. 提出**相对位置编码机制(relative position embedding scheme)**，代替绝对位置编码，在memory的循环计算过程中，避免时序混淆 4. 总的来说，片段级递归机制为了解决编码长距离依赖和上下文碎片化，相对位置编码机制为了实现片段级递归机制而提出，解决可能出现的时序混淆问题。

Transformer+bert+elmo问题整理：https://www.nowcoder.com/profile/353192/myDiscussPost

BERT

- 介绍下BERT：1. BERT是一个谷歌在18年提出的自编码语言模型 2. 模型的结构是只用到了Transformer的Encoder部分，作者分别用了12层和24层的模型 3. 模型在预训练阶段设计了两个任务，分别是采用MaskLM的方式来训练语言模型和句子级别的连续性预测任务 ，也就是输入嵌入分别是token embeddings, segmentation embeddings 和position embeddings 的总和。 输出的是mask的词和预测结果 4. 在训练好预训练模型之后可以进行fine-tuning或者得到动态的词向量或者句子向量。
- 为什么BERT比ELMo效果好？ELMo和BERT的区别是什么？1. BERT用的是Transformer而ELMo用的是lstm，而Transformer的特征抽取能力要好于LSTM 2. BERT在训练模型的深度和训练数据的量都多于ELMo，这点也很大程度上提升了BERT的效果。3. ELMo一般都是动态得到词向量给下游任务，而BERT基本在fine-tuning的时候效果更好一些。
- 讲讲BERT的优点和缺点：优点：1. 考虑到双向上下文信息，2.可以得到动态的词向量或者句子向量缺点1. 在预训练的时候mask，在fine-tuning的时候没有mask，训练数据和测试训练不匹配，造成误差。 2. 缺乏生成能力 3.对mask的tokens没有考虑到他们的相关性。也就是假设mask的词之前是独立的，类似朴素贝叶斯。
- BERT的mask相对于CBOW有什么异同点？ 1. 本质上，两者都是给定上下文，根据它的上文 Context-Before 和下文 Context-after 去预测input word。 2.在CBOW中，每个单词都会成为input word，但BERT不是，因为太耗时，3. CBOW中的输入数据只有待预测单词的上下文，而BERT的输入是带有[MASK] token的“完整”句子，也就是说BERT在输入端将待预测的input word用[MASK] token 4. CBOW模型训练后，每个单词的word embedding是唯一的，因此并不能很好的处理一词多义的问题，而BERT模型得到的word embedding(token embedding)融合了上下文的信息，就算是同一个单词，在不同的上下文环境下，得到的word embedding是不一样的。
- BERT的两个预训练任务对应的损失函数是什么(用公式形式展示)？1. 第一部分是来自 Mask-LM 的**单词级别分类任务**，另一部分是**句子级别的分类任务。** 2.两个损失函数都是log分类损失，最后是两个损失函数相加得到最后损失 3.BERT 还利用了一系列策略，使得模型更易于训练，比如对于学习率的 warm-up 策略，使用的激活函数不再是普通的 ReLu，而是 GeLu，也使用了 dropout 等常见的训练技巧。（**<手写出两个损失函数>**）

  > 参考：https://www.nowcoder.com/discuss/351902

ALBERT

- 介绍下ALBERT：1. 貌似减少的大头是embedding参数 2.参数量降维的大杀器来了 3.NSP

ELMO

- ELMo的基本原理是什么？1. 利用语言模型进行预训练；第二个阶段是在做下游任务时，从预训练网络中提取对应单词的网络各层的Word Embedding作为新特征补充到下游任务中。2. 预训练的语言模型的网络结构采用了双层双向LSTM，任务目标是根据单词x的上下文去正确预测单词x 3. 先将句子X作为预训练好的ELMO网络的输入，这样句子X中每个单词在ELMO网络中都能获得对应的三个Embedding。之后给予这三个Embedding中的每一个Embedding一个权重a，这个权重可以学习得来，根据各自权重累加求和，将三个Embedding整合成一个。这个Embedding作为下游任务的输入。
- ELMo为什么能够达到区分多义词的效果？1. 在两层LSTM得到的embedding是动态改变的，会受到上下文单词的影响，周围单词的上下文不同应该会强化某种语义，弱化其它语义，这样就达到区分多义词的效果。

GPT-2

- 介绍下GPT和GPT-2：1. GPT-2与GPT一样，都使用的是单向语言模型 2. Bert基本就是GPT 1.0的结构，除了预训练阶段采取的是"双向语言模型"之外，它们并没什么本质差异 3.GPT 2.0用更多的训练数据来做预训练，更大的模型，更多的参数。

> https://zhuanlan.zhihu.com/p/56865533

XLNET

- 介绍下AutoRegression自回归模型和AutoEncoder：2. AutoEncoder，代表的是BERT融入双向语言模型，同时考虑上下文，但是训练和测试数据不匹配，存在mask的tokens之前独立的假设1. AutoRegression，传统语言模型，可以理解就是从左到右的语言模型，具有生成能力，考虑了词与词之间的关系，缺点就是只考虑单向信息。3. ELMO可以看成双向AutoRegression，然后拼接看上去能够解决单向问题，因为融合模式过于简单，所以效果其实并不是太好。
- 介绍下XLNet：1. 融合自回归LM和DAE LM两者的优点。就是说如果站在自回归LM的角度，如何引入和双向语言模型等价的效果 2. XLNet有几个主要贡献分别是，XLnet看上去仍然是个自回归的从左到右的语言模型，但是XLNet引入Permutation Language Model的训练目标，通过对句子中单词排列组合，把一部分Ti下文的单词排到Ti的上文位置中，于是，就看到了上文和下文，但是形式上看上去仍然是从左到右在预测后一个单词。3. 双流注意力，为了Fine-tuning阶段和训练的时候输入一致，xlnet用内部打乱方式，实现方法就是双流注意力，一个是内容流自注意力，其实就是标准的Transformer的计算过程，第二个是Query流自注意力，代替Bert的那个[Mask]标记的，通过Query流自注意力忽略掉x3输入了，只保留这个位置信息 4. 引入Transformer-XL， 5.最后总的来说，XLNet就是Bert、GPT 2.0和Transformer XL的综合体变身，首先，它通过PLM预训练目标，吸收了Bert的双向语言模型；然后，GPT2.0的核心其实是更多更高质量的预训练数据，这个明显也被XLNet吸收进来了；再然后，Transformer XL的主要思想也被吸收进来，它的主要目标是解决Transformer对于长文档NLP应用不够友好的问题。

> https://zhuanlan.zhihu.com/p/70257427和https://zhuanlan.zhihu.com/p/71916499

短文本分类及情感分类

- 介绍下你所了解的文本分类的几种算法：1. 词嵌入向量化：word2vec, FastText等等 2. 卷积神经网络特征提取：Text-CNN, Char-CNN等等 3. 上下文机制：Text-RNN， BiRNN， RCNN等等 4. 记忆存储机制：EntNet， DMN等等 5. 注意力机制：HAN等等

  > 参考：https://zhuanlan.zhihu.com/p/34212945

- 说一下nlp的文本分类流程： 1. 预处理，主要是对训练集和测试集的语料库进行处理。 2. 分词，分词是文本分类的核心，这里一般是基于CRF的分词 3. 结构化表示，也就是提取特征 4. 设计分类器进行分类。

  > 参考：https://blog.csdn.net/weixin_40240670/article/details/80612813

对话和问答

- 怎么设计一个对话系统：1. **首先是确定场景边界**（确定需求）。比如我要做一个差旅机器人，我们需要知道这个聊天机器人的场景边界在哪里，换句话说功能有哪些，是只能定机票，酒店，火车票的一种或者几种，还是所有的任务都能完成。以及要确定机器人是否支持闲聊功能。2. **第二步要梳理业务要素和知识库**，同样用机票酒店预订机器人举例子，我们需要知道具体的机票酒店信息，包括时间地点等等。在后续章节中我们会继续的详细介绍，这些都是要梳理到的业务要素。3. **要撰写故事线**，这就想一个编剧编写剧本一样，需要把场景中每个人说的话通过一个完整的故事线体现出来。 4. **第四部分叫抽取对话流程**，换句话说，画出对话**流程图**。

> 参考https://zhuanlan.zhihu.com/p/54157527

NLP其他问题

- CNN，RNN和Transformer在NLP的应用。1. RNN存在的问题主要是，特征提取能力不如CNN和Transformer，然后是RNN并行计算能力比较差 2. CNN并行计算能力强，单层CNN无法抽取远距离特征，需要捕获到长距离的特征有两个方式，分别是Dilated 卷积和加深度

  > 参考：https://zhuanlan.zhihu.com/p/54743941

- 如何在海量数据中快速查找最相似的文本：1. Simhash是google用来处理海量文本去重的算法，将相似的字符串hash得到相似的hash值。这样就可以以接近线性的时间去解决相似性判断和去重问题。2. Minhash算法的基本思想是使用一个随机的hash函数h(x)对集合A和B中的每个元素进行hash。用hmin(A)、hmin(B)分别表示hash后集合A和集合B的最小值，那么有结论：P(hmin(A)==hmin(B)) = Jaccard(A, B)，这是minhash算法的核心。

  > 参考：https://www.zhihu.com/question/32107594/answer/559325508

- 介绍下beam search：1.对于MLE算法训练的模型，beam search只在预测的时候需要。训练的时候因为知道正确答案，并不需要再进行这个搜索。 2.

  > 参考：https://www.zhihu.com/question/54356960

- 介绍一下nltk：1. NLTK是一个主要面向英文的然语言处理工具 2.