NET:name entity typing，命名实体分类，把mention分类，为后续的实体链接啥的减少了candidate（只有这个类别的entity会是候选集，别的类别不考虑了）

OTyper: A Neural Architecturefor Open Named Entity Typing

Neural architectures for fine-grained entity type classification.这个论文的扩展

以下是和之前工作比这个论文的好处：

实体分类的粒度细（types很多很多，fine-grained）

没有的类别也能分，这个类别之前没定义，也能分（Zero shot learning

，判断一个实体是不是属于一个没出现过的type，看意思不是一点label都没有，也是得有点label）

可以给一个实体打多个标签（分到多个类别）

上下文敏感，能利用上下文信息分Chicago是城市还是乐队这种

主要思路：

把mention的embedding和type的embedding映射到同一个空间，mention和对应的类别应该离得近，然后这个模型里没label的也是这样，就把没label的分出来了。

Mention embedding用这个词的embedding加上上下文信息，上下文用bi-LSTM（这个之前见过）办法找embedding；

Type embedding和mention的不含上下文的embedding一样，每个词的embedding平均（这个都是假设我找到对应的entity了，直接略了消歧这种步骤）

问题：

第一个，这里的mention已经预设找到它的entity了，然而这是个过程啊，mention如何找到它的entity，这是一个点。

第二个，咋映射到一个空间？这个变换可做的文章很多，这里乘了俩矩阵，说是叫恒等变换，可以做文章

第三个，没利用知识库，不知道好处还是坏处……

Improving Neural Fine-GrainedEntity Typing with Knowledge Attention

和前人的比好处：

之前mention和context word是单独embedding的，这里用了两者的关联信息（找context种哪些词是和这个mention相关性大的）；

之前的工作没有用KB种关系信息的（有用type信息的），这里应用了KB中的关系信息

还创新了下加了一点消岐

主要思路：

首先，根据mention的每个词的embedding求平均得到mention的embedding（这个过程可以不用存疑了，有消岐啥的加在别的地方，只就mention的基础的embedding就是这么搞，这是基础操作，单词的embedding可以找已有数据）；

之后用各种attention+双向LSTM去学习这个mention的context word的embedding；

得到这两个embedding之后用这个去预测type。

这里面的各种attention：

Semantic attention：不考虑context word和mention的关系，那个公式里只有双向LSTM的输出，就是只考虑context word本身的attention

Mention attention：考虑了mention和context word的相关性，就是那个公式里又有双向LSTM的输出又有mention的embedding

重要：knowledge attention：考虑了知识库中的关系信息，就是那个公式里又有双向LSTM的输出又有mention在知识库中对应实体的embedding（这个实体的embedding是transE学出来的embedding。用TransE的方法做的embedding就是带了关系信息的）。这里训练集中，是已知mention和KB中entity的对应关系的，可以直接找到对应entity的embedding直接用；但是测试集这个对应关系是不知道的。然后他们想了一个办法，在训练的时候，用mention和上下文信息训练一个这个mention的embedding（），想办法让这个embedding和已知KB中对应的entity的embedding(e)尽可能近，然后测试集就可以用这样得到的embedding（）代替KB中entity的embedding。为了效果更好，在test端加了一个小小的消岐：用mention的表面的名字的信息弄出来一个KB中和其对应的entity的候选实体list，然后算这个list的entity的embedding(e)和这个mention的embedding（）算距离，如果这个距离最小的小于一个阈值，就可以用e，否则就用（）

一个问题：

我还是觉得不理解，这个mention的embedding咋就加和求平均得到了，不用消岐吗，不用吗，不用吗？（这个解决了，word2vec有个word embedding的库，就是直接用单词的embedding求和平均得到的）

还有这个attention，我总看到论文有，但是我就是看不懂这个attention咋用的，这个问题得整整

一些我觉得可以创新的思路：

利用KB中的关系的信息这个现在还处于“蓝海”，这里面用的是transE得到的知识库中的entity的embedding，那我能不能用别的？就是如何更好的学习KB的embedding，然后用于attention？（最简单的，用transR啥啥啥）

训练集假定了mention和entity是已经链接上的了，所以knowledge attention部分就直接拿过来用了，如果都像测试集一样预先不知道呢？type工作可以说是链接工作的前期工作，所以我觉得假设预先完全不知道链接关系是合理的，这样的场景怎么办？（已知实体链接的前提下去做实体分类，我觉得有点耍赖，实体分类应该是为链接服务的，我觉得……）