ggnn的代码是选择图中的一个节点，所以我尝试将一句话的情感分类（5类）任务转化为节点选择任务。

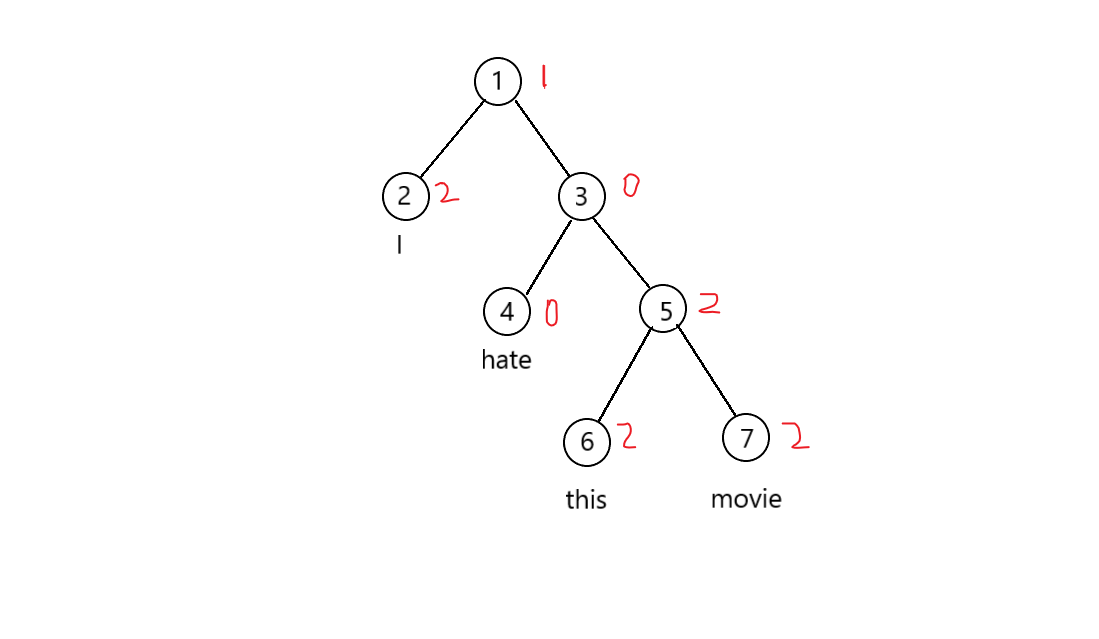
Sentiment treebank的原始数据在treebank/treebankRaw文件夹中，以树的形式给出了一句话，以及每个节点的情感分类。以下是一个例子：

**(1 (2 I) (0 (0 hate) (2 (2 this) (2 movie))))**

每对”()”中就是树的一个节点，包括该节点的分类，子节点（如果是叶子节点就是该节点的单词），其中单词可以忽略，只记录它的分类。

首先将这棵树的每个节点标号（按dfs序），连边。

如图，圆圈内是节点序号，红色数字是分类



为了分类需要，我把图中每个节点拆为5个节点，对应五个分类。

第i个节点拆为( (i-1)\*5, (i-1)\*5+4 )五个节点

而节点之间的边对应拆为25条边，将这两组节点两两连接，这25条边也分为25类

这些边都是指向父节点的有向边。

每个节点的annotation默认为[0,0]，如果此节点是叶子节点，且节点分类对应于此节点，则第一个0改为1。如果是要预测分类的节点，则第二个0改为1。

例如，现在要预测图中1节点的分类，则:

//1节点

annotation[0]=[0,1]

annotation[1]=[0,1]

annotation[2]=[0,1]

annotation[3]=[0,1]

annotation[4]=[0,1]

//2节点

annotation[5]=[0,0]

annotation[6]=[0,0]

annotation[7]=[1,0]

annotation[8]=[0,0]

annotation[9]=[0,0]

//3节点

annotation[10]=[0,0]

annotation[11]=[0,0]

annotation[12]=[0,0]

annotation[13]=[0,0]

annotation[14]=[0,0]

//4节点

annotation[15]=[1,0]

annotation[16]=[0,0]

annotation[17]=[0,0]

annotation[18]=[0,0]

annotation[19]=[0,0]

//5节点

annotation[20]=[0,0]

annotation[21]=[0,0]

annotation[22]=[0,0]

annotation[23]=[0,0]

annotation[24]=[0,0]

//6节点

annotation[25]=[0,0]

annotation[26]=[0,0]

annotation[27]=[1,0]

annotation[28]=[0,0]

annotation[29]=[0,0]

//7节点

annotation[30]=[0,0]

annotation[31]=[0,0]

annotation[32]=[1,0]

annotation[33]=[0,0]

annotation[34]=[0,0]

图中1节点的分类为1，所以ggnn应该选择1节点拆开后的第二个节点，下标为1。

如果预测5节点的分类，则正确答案是下标为22的节点。

对于每棵树，我对每个节点的分类都进行了询问。

treebank/change\_data.cpp和utils/data/dataset.py实现了上述的数据转换， change\_data.cpp转换后节点下标是从1开始的，而不是0，在utils/data/dataset.py代码中，会将下标-1。

treebank/treebank\_graph.txt是转换后的图，询问了所有节点。

treebank/treebank\_graph\_OnlyRoot.txt是转换后的图，只询问了根节点。

treebank/treebank\_graph\_small.txt是缩小版的treebank\_graph.txt，对每种分类的节点有等数量的询问。

以treebank\_graph\_small.txt的图进行训练后的结果是，对于所有叶子节点都能100%预测正确（这个很简单，annotation一开始就给清楚了），然后其他的节点会统一预测同一种分类，不像我预想的那样可以将叶子节点的annotation传递过来，并根据这个进行不同的预测。