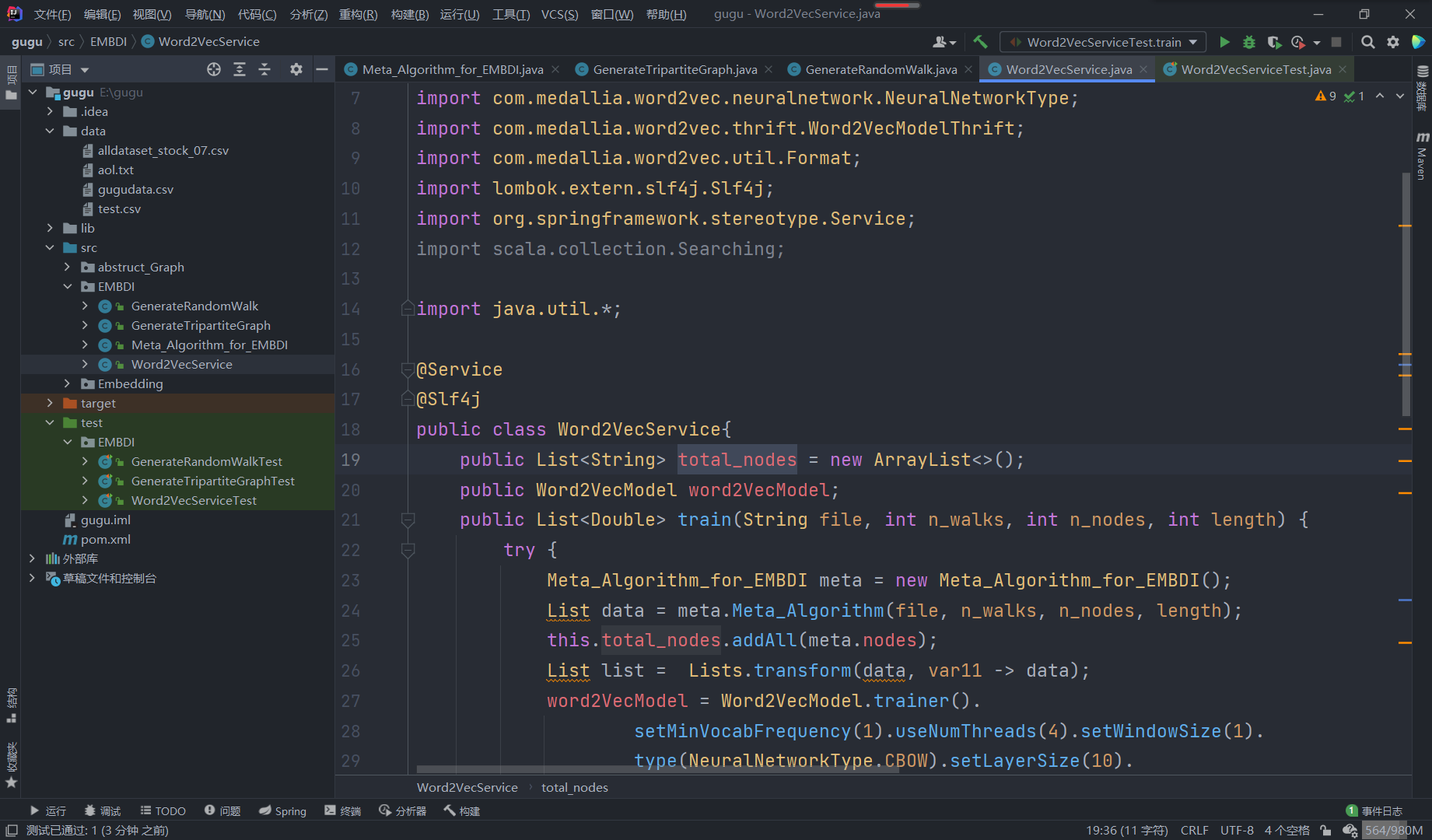
功能说明：

给定当前数据集，生成三分图，我们给出的关于三分图的信息包括：整体所有点的列表list：



在word2vec中可以直接获取到。

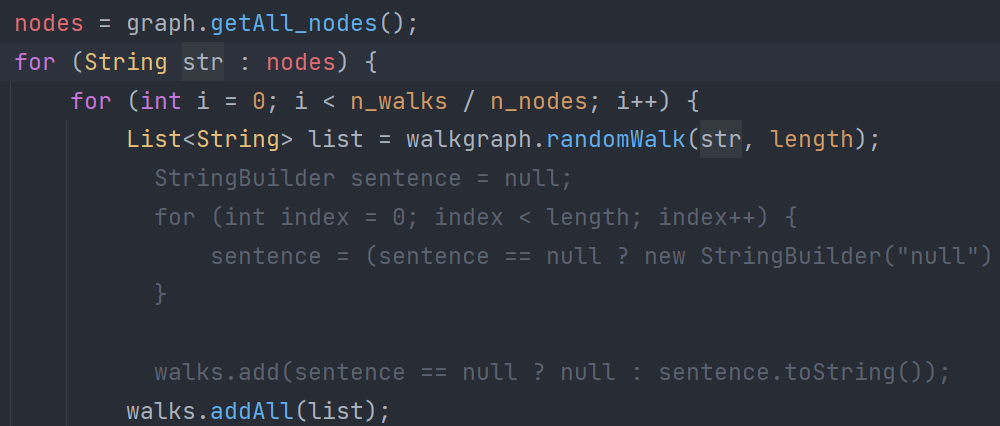
可以查看的信息包括图的点集合，边的集合（有向边，并且可以附加权重），元组RID的集合，所有属性的集合。

数据修复方法：对于连续值出现null，采用当前列的平均值进行修复（因为如果采用全部的列进行平均值计算，需要遍历整个表两次）。对于离散值的null暂未处理（待fix）。

整体代码架构：在GenerateTripartiteGraph进行三分图的生成，数据读取，空值处理，并且对于每一行加上行标记：row\_i；

在generaterandomwalk中，进行随机游走算法，从一个随机选定的开始游走，游走的长度为人为设定的参数length。

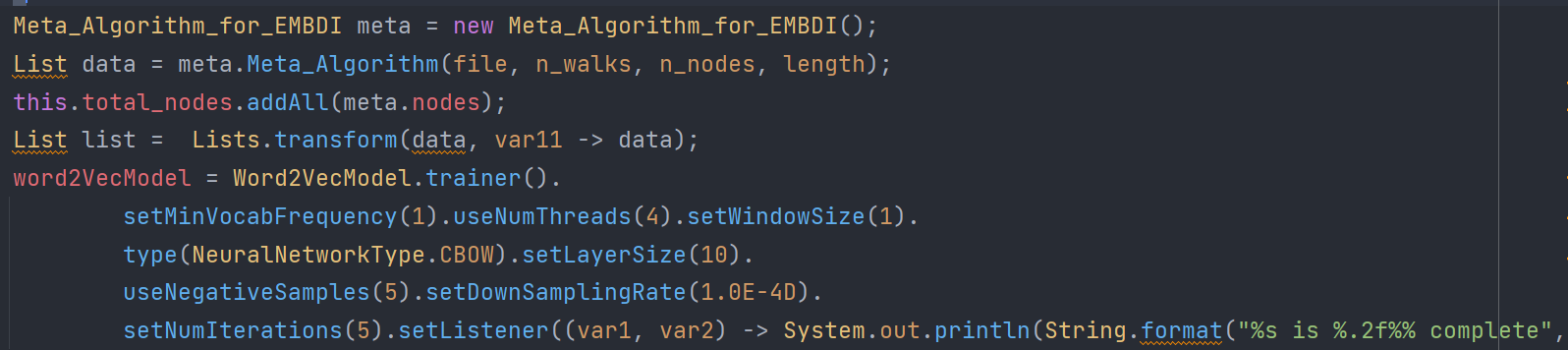
在meta中，对于上述的三分图生成和随机游走实现进行了包装。参数包括：n\_walks:总体游走的次数；n\_nodes：用于调参，具体如下所示：



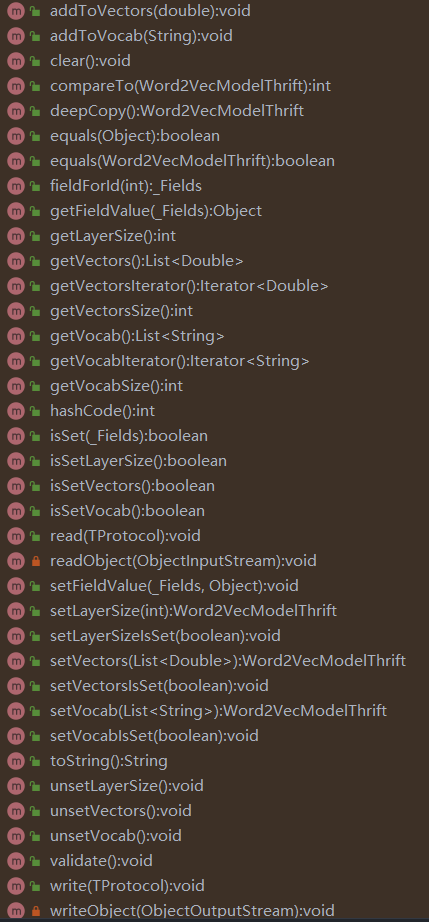
其中，用于word2vec进行训练的是walks中的元素，必须是字符串类型。n\_nodes不影响循环总体次数，起到调参作用。File为文件路径。

在Word2VecService中，对于上述的meta算法进行了包装，提供了两个成员变量进行操作。其中包括total\_nodes，用来提供图中所有的点，可以用来生成每个word的embedding。还有Word2VecModel类型的变量word2VecModel。

在具体的train过程中，我们采用了之前得到的随机游走后得到的数据处理后进行训练：

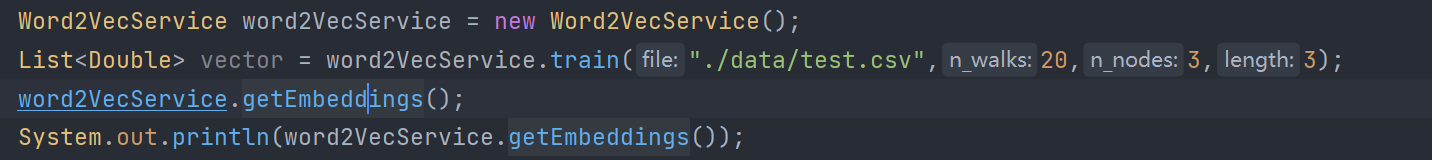


这里我们训练后得到的就是之前的成员变量word2vexModel,可以进行转换得到thrift类型，这种类型可以支持如下操作：包括获取当前vector的列表，当前单词表，laysize（这个就是最后生成的向量的维度）



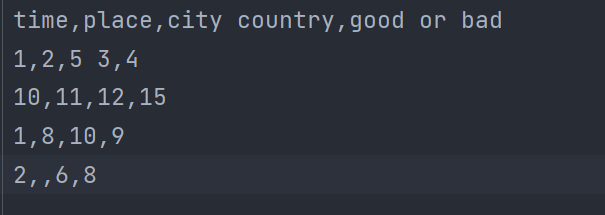
提供了getEmbedding函数，可以对于词计算它的词向量。目前代码中可以对于之前提到的整个图中的nodes进行词向量计算。计算时调用的是searcher接口的getRawVector方法，并且放入了map中，形成<word, embedding>的配对方式。

测试如下：

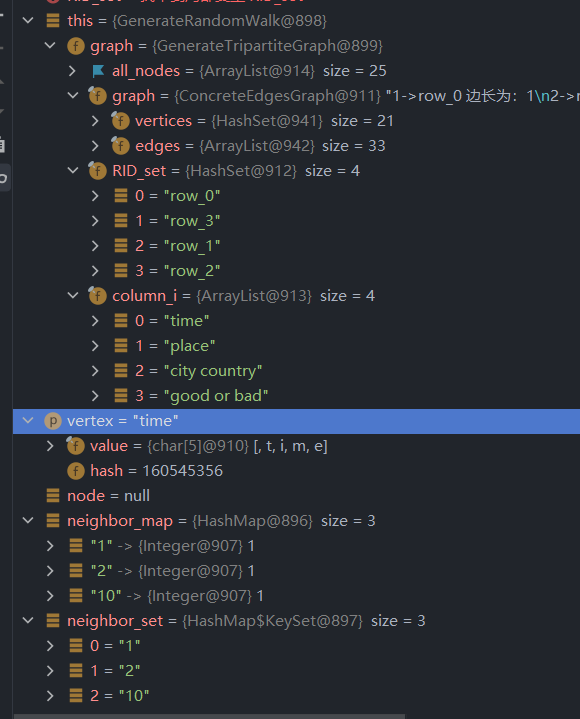


最后输出的结果对照：

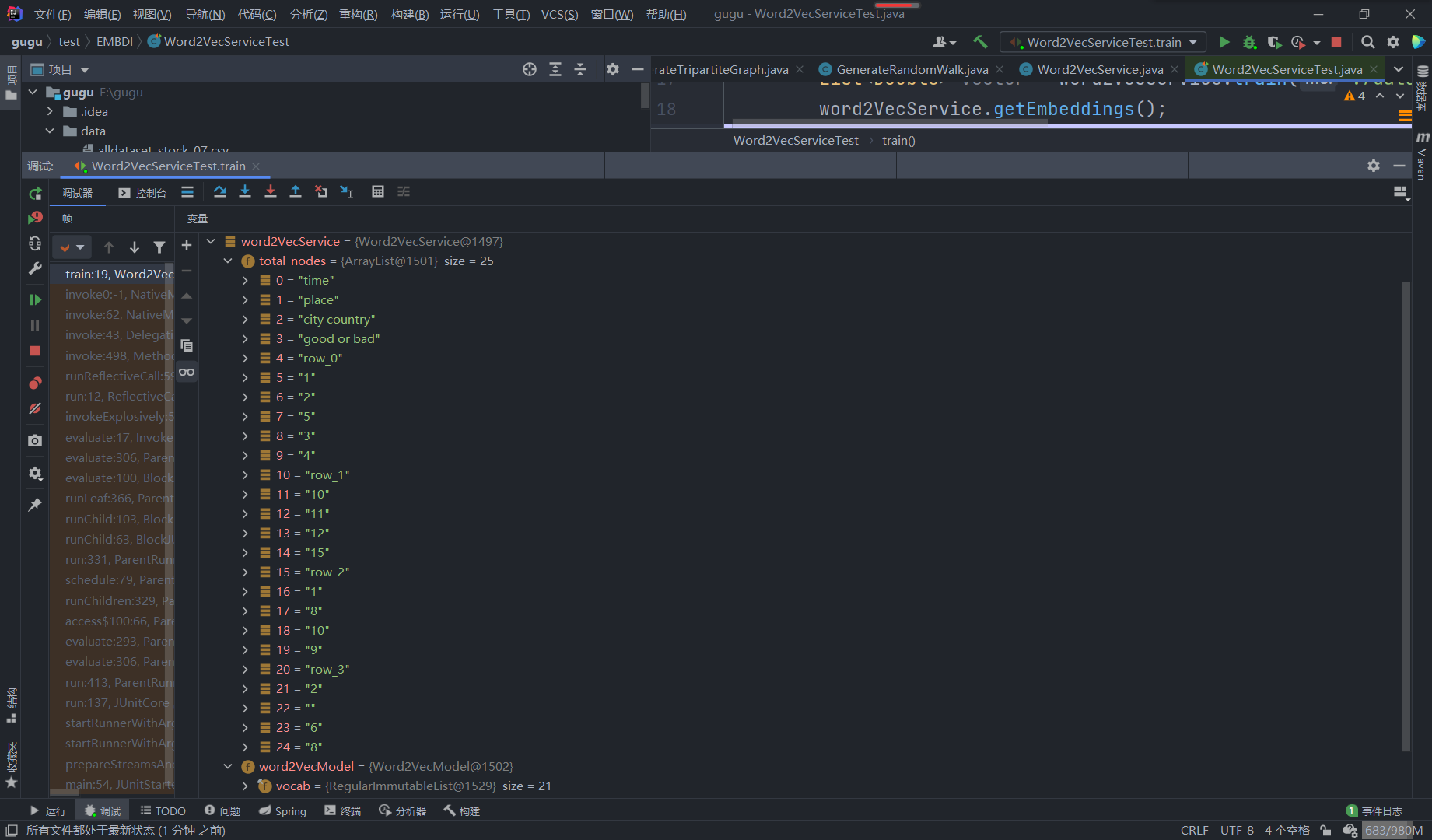
数据：



图的信息：从time开始的游走

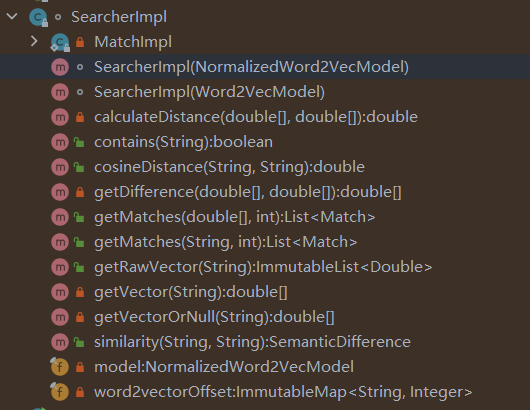


Model信息：

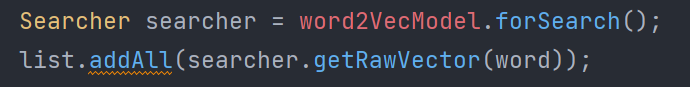


拓展函数to be done:

余弦相似度计算，距离计算：



这个接口的用法：



用word2vec初始化之后，调用函数就可以了