说明：

1. 标点符号识别策略：

1.1）连边规则中的句子定义为：以句号、叹号、逗号、分号、省略号分隔。

1.2）标点符号括号、引号、顿号、冒号、破折号本身作间距1计算。

1.3）百分号、千分号、单位符号在分词软件中会直接与其修饰的数字分为一个词，如5.7%作为数词词性存在，故此三种标点无需考虑。

2. 连边规则：同一个句子中距离小于等于2的词语之间产生连边。

3. 文本类型

3.1）新闻文本：包括新闻标题和新闻正文，正文不考虑分段。

3.2）核心期刊论文文本：暂定只取论文题目和摘要。

（若考虑正文内容则会导致网络规模急剧扩大，拟根据后期效果决定是否取用正文内容。问：前期是否就应考虑此种情况？）

一、文本网络构建（假设能够调用分词软件接口）

1. 文本输入

2. 调用分词软件接口，分词结果存入字符串，平台显示分词结果

3. 字符串一次分割，得到字符串数组（数组元素格式为“词语\词性”）

4. 字符串数组元素二次分割，分割结果分别存入节点类的节点内容及节点词性属性

5. 记录词语在文本中的位置：通过检测标点句号、叹号、逗号、分号、省略号来断句，其中，位置编号规则为：位于标题的词语记为1，位于首末句的词语记为2，其他句词语记为3。存入节点类的节点位置属性

6. 预处理

6.1 去停用词：查询数据库中停用词表，若存在，则删去相应节点(节点建立时候去不创建即可，何必创建后再删除，浪费系统资源)

6.2 去无用词性词语：查询数据库中无用词性，若存在，则删去相应节点（此过程保留标点符号）（同上，创建时直接跳过）

7. 构建文本网络：依据连边规则，只对间距2以内的进行连边

7.1 若下一个节点为词语，则产生连边（即同一个句子中）

7.2 若下一个节点为标点，检测标点类型

7.2.1 若为句号、叹号、逗号、分号、省略号，则删除标点节点，当前节点与后续节点不再产生连边

7.2.2 若为括号、引号、顿号、冒号、破折号，则删除标点节点，当前节点与后续两个节点产生连边，加入该节点连边列表。（建议此时加入连边列表的是节点内容而非节点编号，因为在删除节点或者后续合并节点过程中，节点编号会发生变化）。【构建节点类时，节点编号还是列表中节点位置信息的映射吧？】

8. 词语相似度计算

两层循环，依词性一一计算，找出与当前节点相似度超过设定阈值的节点并进行合并（后面节点向前面节点合并），其中，连边列表的合并规则为：

8.1 若当前节点的连边列表中不存在某词语，则直接加入该词语；

8.2 若当前节点的连边列表中已存在某词语，则将这两个词语节点之间的连边权重增加1。

（此处可能要加一步短语发现，如名词短语、形容词短语、动宾短语。若增加这一步，就要考虑节点权重赋值问题。后期根据实验效果考虑是否要加）

9. 整理连边列表：将词语对应为相应的节点编号。

以上，文本网络构建完成。

关键词网络构建算法

输入数据：目标文本，停用词列表，无用词性列表

Step 1：分词，利用分词软件输出“词语+词性”组合列表，存入List<string> WordRes；

Step 2：分句，根据标点符号将目标文本分成句子，存入List<string> Sentence；

Step 3：构建节点，输入词语内容strWord，词性intType，根据Sentence确定词语位置intPositon，和所属句子索引intLineNum，构造节点存入List<kNode> NodeList中。

Step 4：扫描节点，如果命中停用词表和无用词性表的节点索引赋予-1。随后合并同义词节点，被兼并节点索引赋予-1，其kNode kOwner 引用指向兼并它的节点。最后，从前到后对所以不是-1的节点依次赋予索引，本索引为永久索引；

Step 5：根据连边规则，依据位置intPosition和所属句子索引intLineNum进行加边，对于被兼并节点，将边加载到其宗主节点上;

Step 6：新建关键词网络kNet，从 List<kNode> NodeList中取出索引>-1的节点，加入网络，完成后续处理，结束网络构建。