1. 概述

在网络上查询了句法分析和依存句法分析的相关工具使用，以及查阅了句子分类实现过程的相关信息，了解到完成句法分析任务的主要工具有hanlp、斯坦福nlp和哈工大的ltp。后两个查询到具体的操作过程，在完整的句子分类实现过程中作者使用的是ltp（linux环境下）。下面分别是：参考修改后的句子类别概括（一）、完整实现过程（二、三、四）和斯坦福nlp/ltp的使用相关过程（五）。

一、句子类别及对应规则（3.0）

把字句规则

主语+"把"+宾语+动作

被字句规则

宾语+"被"+主语+动作

判断句规则

主语+\*+表语

\*指：（不）是，乃，皆（是），非，（不）为

注：动词谓语不表判断

因果复句常用关联词语：

因为、由于、是因为、是由于、所以、因此、因而、以致、致使、从而、故、才

因为（因）（由于）……所以（才）（就）（于是）（因此）（因而）、之所以……是因为（是由于）（就在于）

条件复句常用关联词：

如果……就（那么）、只要……就、凡是……都、只有……才、除非……才

陈述句规则

1.主语+谓语+宾语

2.主语+谓语+宾语+补语

3.复句（多个谓语宾语补语）

省略句规则

普通句式缺省人名主语或职称宾语

转折复句常用关联词

虽然……但是、尽管……还是、却、然而、但是、即使……也

递进复句常用关联词

不但……而且、不光……也、不仅……还、不但不……反而

感叹句

带有浓厚的感情的句子。它表示快乐、惊讶、悲哀、厌恶、恐惧等浓厚的感情。感叹句一般用降调，句末都用叹号“！”

祈使句

1.表示命令的祈使句：保持肃静！起立！立定！站起来！快去救火！等

2.表示请求的句式：请……（请等我一会）；

3.表示禁止的句式：不准……，不要……，别……；请勿……

疑问句（\*）

1.是非问句，主要特点是回答时用肯定或者否定答句，是非问句一般用语气助词“吗”：会（是）…吗，好吗，对吗，是吗，可以吗，能吗，信吗，会吗，怕吗，算吗

2.特指问句（疑问词问句）：“谁、哪儿、什么”等；还有一些疑问词语为“多+adj”格式，如“多大、多高、多长”等

3.选择问句：选择问句用选择形式提出两种(或多种)不同情况，要求对方选择其中一种情况作为回答。选择问句常用“是……还是……”连接，“是”也可以省略

4.正反问句：正反问句是把谓语的肯定形式和否定形式并列起来然后提问的疑问句，回答者可选择其中之一作为回答：比如：会不会，能不能

5.反问句：反问句是用疑问句的形式来表示肯定或否定的意义，带有很强的强调意味：难到，难道不，怎么会，怎能，怎么能，莫非，难不成，不是吗等

二、将句法分析与正则结合标注句子类型

可以使用LTP，StanfordCoreNLP等工具分析句子类型的语法结构；

LTP中文句法分析；

根据句子特定结构的关键词，构造正则表达式；

将正则表达式与句法分析结合进行句子类型标注；

可以使用XGBoost等模型进行训练和测试或者直接使用正则和句法结构进行判别；

三、中文句子类型分类工具sentypes实现

1、基于C++中regex库组件构建正则表达式接口类RegularEX；

2、完成句法分析LTP在Linux下的C++源码的编译和测试；

3、构建基于C++的句法分析类Parsing，利用该类能够进行分词、词性标注和句法分析；

4、基于正则类RegularEX、句法分析类Parsing和句子类型规则等完成所有句子类型的分类代码；

5、编写Makefile文件；

6、将库文件，模型文件和代码等文件打包，在该文件夹下，执行make命令，即可生成可执行文件sentypes，即为句子类型分类工具；

文件主要包括：（1）依赖的库文件：include、lib、thirdparty；（2）LTP模型：ltp\_model；（3）句法分析和正则表达式类的头文件：parsing.h、regular\_ex.h；（4）句子类型分类主程序：sentence\_type.cpp；（5）Makefile文件

7、sentypes工具包含两个命令参数，分别是输入文件和输出文件，其中，输入文件是句子文件，输出文件是标注好句子类型的结果文件；

四、Linux下运行sentypes工具

$ ./sentypes data/test\_sen data/results

五、相关工具使用

1.斯坦福使用步骤：

在使用StanfordCoreNLP对文本句子进行分析时，需要先对句子进行分词

nlp.word\_tokenize(sentence)

然后对分词后的句子进行句子成分分析

nlp.pos\_tag(sentence)

然后继续进行命名实体识别

nlp.ner(sentence)

再之后就是句法分析与依存句法分析

nlp.parse(sentence)

nlp.dependency\_parse(sentence)

在使用StanfordCoreNLP对中文文本句子进行分析时，步骤与英文句子分析是一样的，唯一不同的是这一句nlp=StanfordCoreNLP(r'E:\自然语言处理\stanford-corenlp-full-2018-10-05',lang='zh')

而在内部实现中，parser函数

public void testLexicalizedParser() throws IOException {

LexicalizedParser lp = LexicalizedParser.loadModel(this.getClass().getClassLoader().getResource("xinhuaFactoredSegmenting.ser.gz").getPath());

List<String> lines = Arrays.asList("\*\*\*\*\*\*");

lines.stream().forEach(sentence -> {

Tree tree = lp.parse(sentence);

ChineseGrammaticalStructure gs = new ChineseGrammaticalStructure(tree);

Collection<TypedDependency> tdl = gs.typedDependenciesCollapsed();

System.out.println("sentence:"+sentence);

tdl.stream().forEach(typedDependency -> {

System.out.println("Governor Word: [" + typedDependency.gov() + "]

Relation: [" + typedDependency.reln().getLongName() + "]

Dependent Word: [" + typedDependency.dep() + "]");

});

});

}

2.LTP相关：

哈尔滨工业大学中文依存句法分析工具（LTP）（该工具只能进行依存句法分析）：[GitHub - HIT-SCIR/pyltp: pyltp: the python extension for LTP](https://github.com/HIT-SCIR/pyltp)

LTP [LTP4 文档 — LTP4 4.1.4 文档](https://ltp.readthedocs.io/zh_CN/latest/)

斯坦福和ltp对比（截取自清华学生论文）：

在对中文进行依存句法分析时，standfordCorNLP使用起来比LTP简单许多，因为在不自己写函数的情况下，可以直接调用模型函数，而LTP里需要自己来写大部分内容，仅代码量来说就比StanfordCoreNlp多很多。不过如果自己写函数的话，其实两种方法的工作量是差不多的。再看处理效果，在我的实验结果中，对中文的处理效果是差不多的，可能StandfordNLP要更好一点，但程度不高。而LTP无法处理英文。