编译原理课程设计规格说明

本课程设计要求同学们利用集合演算中所提出的集合演算算法，利用 C/C++ 实现该算法(可参考原论文所提供的源代码)，并利用该算法实现自然语言的依赖解析过程(参考原论文读出依赖过程的伪代码)。

另外，也推荐同学们实现该算法的java版本，利用stanfordCoreNLP中所提供的(tagger和parser)接口扩展原算法的可用性。鉴于stanfordCoreNLP为java编写且项目代码量较大，该要求不强制，仅作为加分项。

在本文档中，首先将介绍传统自然语言解析过程与编译过程所涉及解析的相似性，以解释为何希望同学将在编译原理课上所学习的知识应用到自然语言解析过程中。而后，将简单介绍集合演算所涉及的基础概念以及底层的运算逻辑，使得同学们对该计算方法有直观的认识。随后，给出本课程设计的具体要求与给分细则，同学们需要据此一小组为单位完成课程设计的内容。最后，将简单介绍加分项中的stanfordCoreNLP，并对如何完成该加分项进行介绍。

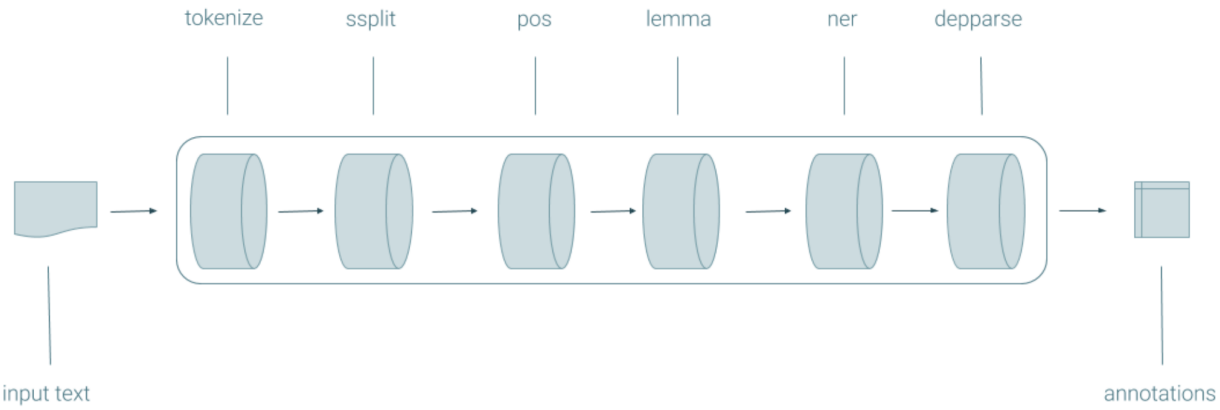
本文档中的叙述参考stanford的Speech and Language Processing课程。在完成本课程设计时可参考该课程第15课时的课件，加深对于依存关系图的理解。

自然语言解析过程与编译解析过程

本章将说明传统的自然语言解析过程与编译解析过程的相似性，以说明为何可以将编译过程中的知识迁移到自然语言解析过程中。

在传统的自然语言解析中，会将一个较为复杂的任务划分多个依赖关系的很多子任务，较为常用的划分如下：

1. 分词（Tokenization）：将连续的文本序列切分成更小的单位，如单词或子词。分词是NLP的基础步骤，用于为后续处理建立基本的文本单元。
2. 词性标注（Part-of-Speech Tagging）：为文本中的每个词语标注其词性，例如名词、动词、形容词等。词性标注有助于捕捉句子的语法和语义信息。
3. 句法分析（Syntactic Parsing）：分析句子的结构和语法关系，如主谓关系、修饰关系等。句法分析可以帮助理解句子的语法结构和句子成分之间的关系。
4. 语义角色标注（Semantic Role Labeling）：识别句子中的语义角色，如施事者、受事者、时间、地点等。语义角色标注有助于理解句子中的语义信息和事件结构。
5. 命名实体识别（Named Entity Recognition）：识别文本中的命名实体，如人名、地名、机构名等。命名实体识别有助于提取文本中重要的实体信息。
6. 实体关系抽取（Relation Extraction）：识别文本中实体之间的关系，如人物之间的关系、产品与制造商之间的关系等。实体关系抽取有助于构建实体之间的知识图谱。
7. 语义理解（Semantic Understanding）：对文本进行深层次的语义分析和理解，包括词义消歧、指代消解、情感分析等。语义理解模块旨在更好地理解文本的意义和语义。



这些子任务通常按照顺序依次进行，形成一条数据流动链(pipeline)，以逐步深入理解和处理自然语言文本。实际上，这些子任务与编译过程中的各个子模块(词法分析，语法分析，语义分析)有着众多相似之处，通过理解自然语言中类似子任务执行逻辑，相信会拓展同学们在编译原理课上的思维，加深对原有知识的理解。

自然语言与形式语言的相似性

自然语言与形式语言虽然在二义性等方面有着显著的区别，但是二者在语法规则和语义解析上有着相似之处。首先，二者都具有语法规则，用于描述有效的句子或表达式结构。它们都依赖于一定的规则和约定来构成合法的语言结构。其次，二者都有语义，即语言单位的意义或解释。它们都使用特定的规则和约定来赋予语言单位以含义。

在`Google`提供的技术文档编写教程中，有如下的叙述：

> If you change the name of a variable midway through a method, your code won't compile. Similarly, if you rename a term in the middle of a document, your ideas won't compile (in your users' heads).

可以看到，其中将大脑解析自然语言的过程类比为编译器编译的过程，这也印证了编译过程和大脑解析自然语言过程的相似性。

词性标注过程与词法解析过程

> **自然语言中的词性(part of speech)与编译原理词法解析阶段中的token**
>
> 自然语言处理中，第一步往往是对于最小语义单元(i.e 在英语中为每个单词)*标注*词性信息，随后在利用解析结果进行后续的语法解析与语义解析操作；而在编译原理中，第一步即为利用正则语言解析的方法对源代码进行词法解析，生成所需要的tokens序列为后续的语法解析过程服务。

在英语中，词性(part of speech)有以下的属性：

Part of Speech	Definition
Noun	A person, place, concept, or thing
Pronoun	A noun that replaces another noun (or larger structure)
Adjective	A word or phrase that modifies a noun
Adverb	A word or phrase that modifies a verb, an adjective, or another adverb
Preposition	A word or phrase specifying the positional relationship of two nouns
Conjunction	A word that connects two nouns or phrases
Transition	A word or phrase that connects two sentences

在C语言中，则有如下的属性：

Attribution	Definition
Keywords	The keywords are pre-defined or reserved words in a programming language.
Identifiers	Identifiers are used as the general terminology for the naming of variables, functions, and arrays.
Constants	The constants refer to the variables with fixed values.
Strings	Strings are nothing but an array of characters ended with a null character (‘\0’).
Special Symbols	Special symbols are used in C having some special meaning and thus, cannot be used for some other purpose.(e.g. [] , {})
Operator	Operators are symbols that trigger an action when applied to C variables and other objects.(Binary or Unary)

可以看到，无论是自然语言处理中还是编译过程中，针对最小语法单元有着相似的处理。而其处理结果也有着相似之处。

利用`clang++ -fsyntax-only -Xclang -dump-tokens *.cpp`， 我们可以得到如下的输出：

```
...
l_paren '(' Loc=
</usr/local/opt/llvm@17/bin/../include/c++/v1/__verbose_abort:25:28>

const 'const' Loc=
</usr/local/opt/llvm@17/bin/../include/c++/v1/__verbose_abort:25:29>

char 'char' [LeadingSpace] Loc=
</usr/local/opt/llvm@17/bin/../include/c++/v1/__verbose_abort:25:35>
```

```
star '*' [LeadingSpace] Loc=
</usr/local/opt/llvm@17/bin/../include/c++/v1/__verbose_abort:25:40>

identifier '__format' Loc=
</usr/local/opt/llvm@17/bin/../include/c++/v1/__verbose_abort:25:41>

comma ',' Loc=</usr/local/opt/llvm@17/bin/../include/c++/v1/__verbose_abort:25:49>

ellipsis '...' [LeadingSpace] Loc=
</usr/local/opt/llvm@17/bin/../include/c++/v1/__verbose_abort:25:51>

r_paren ')' Loc=
</usr/local/opt/llvm@17/bin/../include/c++/v1/__verbose_abort:25:54>

semi ';' Loc=</usr/local/opt/llvm@17/bin/../include/c++/v1/__verbose_abort:25:55>

r_brace '}' [StartOfLine] Loc=
</usr/local/opt/llvm@17/bin/../include/c++/v1/__verbose_abort:58^C/usr/local/opt/l
lvm@17/bin/../include/c++/v1/__config:846:37
...
```

类似，在自然语言处理中，一般为标注单词的词性(较为常用的方式HMM算法或深度学习的算法)。虽然由于自然语言中一词多义的性质，其并不能被正则语言所表达，因此同学们课上所学的词法解析方式并不适用， 但是二者的输出实际上相当相似。在[stanfordCoreNLP]框架中，其在`tokenization`中的输出如下：

```
Input : <<
Scores of properties are under extreme fire threat as a huge blaze
continues to advance through Sydney's north-western suburbs. Fires
have also shut down the major road and rail links between Sydney and
Gosford.

The promotional stop in Sydney was everything to be expected for a
Hollywood blockbuster - phalanxes of photographers, a stretch limo to
a hotel across the Quay - but with one difference. A line-up of
masseurs was waiting to take the media in hand. Never has the term
"massaging the media" seemed so accurate.

Output : >>
Scores/NNS of/IN properties/NNS are/VBP under/IN extreme/JJ fire/NN threat/NN
as/IN a/DT huge/JJ blaze/NN continues/VBZ to/TO advance/VB through/IN Sydney/NNP
's/POS north/NN -/: western/JJ suburbs/NNS ./.
Fires/VBZ have/VBP also/RB shut/VBN down/IN the/DT major/JJ road/NN and/CC rail/NN
links/NNS between/IN Sydney/NNP and/CC Gosford/NNP ./.
The/DT promotional/JJ stop/NN in/IN Sydney/NNP was/VBD everything/NN to/TO be/VB
expected/VBN for/IN a/DT Hollywood/NNP blockbuster/NN -/: phalanxes/NNS of/IN
photographers/NNS ,/, a/DT stretch/NN limo/NN to/TO a/DT hotel/NN across/IN the/DT
```

```
Quay/NNP -/: but/CC with/IN one/CD difference/NN ./.
A/DT line/NN -/: up/IN of/IN masseurs/NNS was/VBD waiting/VBG to/TO take/VB the/DT
media/NNS in/IN hand/NN ./.
Never/RB has/VBZ the/DT term/NN "/'' massaging/VBG the/DT media/NNS "/``
seemed/VBD so/RB accurate/JJ ./.
```

同学们可以看到，此处的标注属性仿佛并不是以前学习的标签。实际上，此处的标签代表的是在自然语言词性标注的45-tag Penn Treebank tagset，其相关介绍如下：

45 tag penn treebank tagset

An important tagset for English is the 45-tag Penn Treebank tagset (Marcus et al., 1993), shown below, which has been used to label many corpora. In such labelings, parts of speech are generally represented by placing the tag after each word, delimited by a slash:

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	*$*
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	*#*
JJS	superlative adj	*wildest*	SYM	symbol	*+,%, &*	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

自然语言中的依赖解析(dependency parsing)过程与编译过程语法解析过程

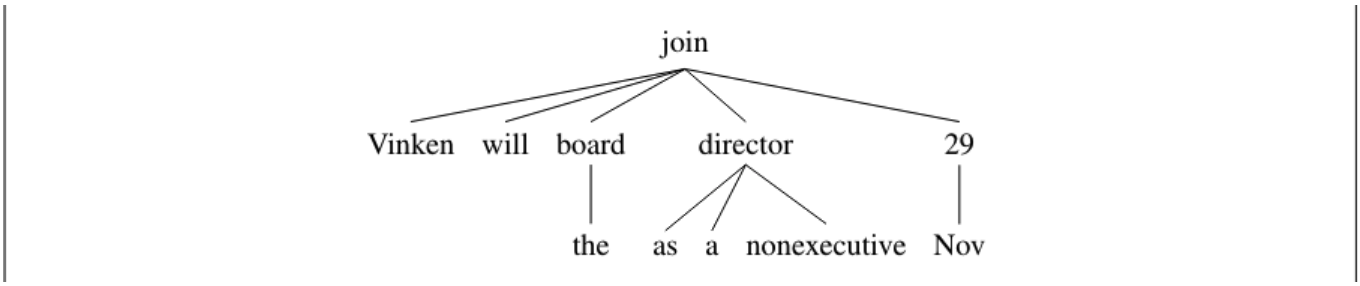
自然语言中的依赖解析(dependency parsing)是一种仅仅使用通过句子中的单词（或词性）以及单词之间相关的有向二元语法关系的语法解析过程，其是句法分析的其中一种表现形式。同学们需要参考stanford的Speech and Language Processing课程的第15课时的课件，了解相关领域的知识以及常用的解析算法。

实际上，无论是在自然语言处理还是编译过程语法解析中，我们一般都选用树状结构作为最终输出的表现形式。而在依赖解析过程中，其输入一般为完成词性标注过程的token，输出为以短语或单词为节点

的多叉树，这与编译原理中的语法解析以句法解析后的token作为输入，并以AST作为输出的形式是颇为相似的。

为了更加直观的显示二者的相似关系，此处直接比较二者的树状输出结果。在终端中输入`clang++ -Xclang -ast-dump -fsyntax-only .*.cpp`，我们可以看到：

而在依赖解析中的树状输出则如下所示(其一般以谓语作为根节点)：



集合演算基础运算逻辑

在当前的研究中，集合演算 是应用于依赖解析中的方法，从原论文的伪代码中我们可以得知，该模型需要输入文本的词性标注信息(词性以及句子成分)。而后，通过其定义的集合演算模型，解析出原句子中的依赖关系(该步骤可类似于编译原理中的IR(intermidiate representation)生成阶段)。

其中涉及的实体以及实体间的运算关系如下：

- **大脑(brain)**：整个大脑被定义为一个概率连接有向图$G_{n,p}$,其中划分为多个脑区。脑区为其子图，脑区内的节点以p的概率随机连接；脑区间的连接则成为**fiber**，如果两个脑区之间有**fiber**，则两个脑区间的神经元可以以概率p进行连接。在整个动力系统(在突触权重更新的情况下，可以建立该系统的微分方程或差分方程表示)运行时，假设其以timestep的离散时间步进行，在每个时间步会进行突触输入计算，选举cap以及权重等操作。
- **脑区(area)**:脑区拥有两种状态，抑制或者解除抑制。在抑制状态下保留其cap,在解除抑制状态下则会通过计算突触输入得到新的cap。
- **神经元集合(assemblies)**:一个**脑区**中神经元的子集，且具有固定的大小k(一般为未抑制脑区中活跃度最高的k个神经元)。
- **投影(projections)**:$project(x, B, y)$,一个位于A脑区中的神经元集合x通过**fiber**将自身的信息**project**(投影)到脑区B中的神经元集合y,可以称$x = parent(y)$。具体操作是利用已稳定的x神经元集，通过A和B间的连接改变B中神经元的活跃度,在一定的时间步后，得到更新后B中最活跃的k个神经元组成y神经元集合。

- **神经元兴奋度(突触输入SI)**:对于每一个神经元而言，其通过突触连接(有向边)计算兴奋度的公式如下:

$$SI(i,t+1)=\sum_{(j,i)\in E} fires(j,t)w_{ji}(t)$$

- cap:一个脑区中活跃度最高的k个神经元称为该脑区在对应时间内的cap,其中神经元的状态则为$fired$兴奋。设时间步t时活跃的神经元集为A_t，则定义**core neuron**为$A_t\cap A_{t+1}$，即在相邻时间步中都为cap的神经元。
- **支持集**:一个脑区的支持集定义为在所有时间步中曾经活跃过的神经元集的并集，其数学表示为:$A^*=\cup_t A_t$。
- **赫布可塑性(Hebbian plasticity)**:在该系统中，如果存在突触连接的神经元，源节点在时间步t兴奋($fired$),目标节点在时间步$t+1$兴奋，则二者之间的权重增加为原来的$1+\beta$。(β为人为设定，后面会看到其与支持集密切相关)

基于原论文给出的伪代码实现自然语言依赖解析

课程设计要求： 同学们需要根据集合演算中提供的伪代码实现其依赖解析的 `CPP` 版本。在论文所给出的代码仓库中，实际上给出了一个较为简单的 `CPP` 版本代码，但其尚为能完成解析的过程。同学们需要重构其中的代码，并使其可以完成简单的依赖解析。在提交作业的时候，需要同学们提交以下项目:

- 项目源代码文件夹，需要支持clang++或g++编译器，建议使用C++20版本。原论文的代码仓库使用的是bazel构建项目文件，此处建议同学们使用cmake这一较为传统的工具管理源文件和目标文件。
- 项目所支持的单词库文件，只需罗列出所有支持的可解析单词。
- 项目测试代码文件夹，需要使用google test验证其可以完成单词库中所有单词在满足英语语法规则下的句子均可解析成功，助教会验证相关结果。
- 项目性能测试文件夹，该文件夹需要测试同学们所编写的解析器的解析性能，最终的结果为解析时长/句子单词数。
- 项目实验报告文件夹，此处的报告推荐使用markdown或者latex编写，并转化为pdf版后提交。在实验报告中需要阐明:
  - 小组内组员的组员的分工;
  - 对于依赖解析以及集合演算模型的认识(类似读书报告的形式);
  - 项目设计的思路，包括设计的数据对象，类之间的交互关系，如何保证线程安全，如何提升代码性能等;

截止时间:
- 5.23: 对于依赖解析以及集合演算模型的认识(类似读书报告的形式)， 以 `第{}组-{组长学号}-{组长姓名}-编译原理课程设计-依赖解析-集合演算.zip` 的格式发送到 `chenjh535@mail2.sysu.edu.cn` 邮箱;
- 6.6: 整体项目，以 `第{}组-{组长学号}-{组长姓名}-编译原理课程设计.zip` 格式发送到 `chenjh535@mail2.sysu.edu.cn` 邮箱;

附加选项

完成了以上任务的同学可以考虑将集合演算耦合到stanfordCoreNLP的依赖解析模块中，利用stanfordCoreNLP提供的词性标注模型，提升集合演算的解析能力。

耦合stanfordCoreNLP完成依赖解析

同学们可以根据stanfordCoreNLP的官网

- 第一步是根据官方文档对该项目进行了解，清楚其中涉及的数据对象以及相互关系，建议使用草稿纸作出简要的UML图加深印象。这一部分需要足够的耐心，直至大致清楚其中的算法流程。
- 第二步是需要下载源码并完成其中关键实体的注释阅读，面对大型的开源项目，其一般具有可读性较强的注释。通过对其注释的阅读，我们可以更加清楚其中各个对象间的依赖关系(此处仍然推荐利用纸笔强化认识的过程)。需要在这一部分清楚数据流动的形式，并在之后的源码阅读中始终牢记当前所处理的数据对象。
- 第三步是阅读各个模块所提供的接口，通过接口提供的信息，我们可以知道不同模块直接是如何进行数据传输与转化的。
- 第四步是利用深度有限的深度优先搜索方式了解接口的实现方法，切忌无限递推的打开大量的文件，在不清楚一个类的实际作用时，首先是根据类名和函数名进行估计而非直接硬啃源码。

完成了该项目的深入了解后，同学们可以思考如何将集合演算耦合到stanfordCoreNLP的依存关系解析模块中，欢迎同学们与助教进行讨论，如果有同学对这部分内容感兴趣，我们可以一起完成这部分代码并提交到社区中，为开源作出贡献。

Java语言学习

此处推荐的java教程为on-java-8，其学习网址如下：
https://zyb0408.github.io/gitbooks/onjava8

此处推荐的maven教程为baeldung提供的教程，其中可自行下载电子书，其学习网址如下：
https://www.baeldung.com/executable-jar-with-maven