Linux 与词汇分析
（Brown vs. PubMed 案例研究）

Jingbo Xia

Huazhong Agricultural University

xiajingbo.math@gmail.com

2026/09/17

本节目标 / Learning objectives

完成本节课后，学生应能够：

1. 解释语料库（corpus）、词符（token）、词型（type）和词型-词符比（type-token ratio, TTR）；
2. 使用 Linux 管道（pipe）搭建一个简单的词汇分析（lexical analysis）流程；
3. 用 shell 命令计算 token count、type count 和 TTR；
4. 说明为什么原始 TTR 不适合直接比较长度差异很大的语料；
5. 设计一个可复现的方案，比较通用英语文本与生物医学文本的词汇特征。

本节课的核心不是背命令，而是用命令完成一个可复现的 BioNLP 文本分析任务。

来自 Alex 的启发案例 / Motivating case



目录 / Table of contents I

1. 语料库与 TTR / Corpus and TTR 5
   - 语料库 / Corpus 6
   - 词汇分析 / Lexical analysis 13
2. Linux 基础命令 / Commands in Linux 18
   - cat、tr、sort、uniq 与 wc 20
   - awk 与 sed 25
   - echo、read 与 IFS 29
   - while 循环 / While loop 33
3. 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37
   - 材料与任务 / Material and task 38
   - 代码与实践 / Codes 40
4. TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46
   - 从手写命令到智能体协作 / From manual Shell to agent collaboration 47
   - Prompt 是实验方案 / Prompt as experimental protocol 53
   - 挑战 1：生成完整 TTR 命令 / Challenge 1 59
   - 挑战 2：改进 tokenization / Challenge 2 67
   - 挑战 3：等长采样比较 / Challenge 3 74
   - 审查、验证与讨论 / Review and verification 81
   - 总结 85

Outline

1. 语料库与 TTR / Corpus and TTR 5
   - 语料库 / Corpus 6
   - 词汇分析 / Lexical analysis 13
2. Linux 基础命令 / Commands in Linux 18
   - cat、tr、sort、uniq 与 wc 20
   - awk 与 sed 25
   - echo、read 与 IFS 29
   - while 循环 / While loop 33
3. 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37
   - 材料与任务 / Material and task 38
   - 代码与实践 / Codes 40
4. TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46
   - 从手写命令到智能体协作 / From manual Shell to agent collaboration 47
   - Prompt 是实验方案 / Prompt as experimental protocol 53
   - 挑战 1：生成完整 TTR 命令 / Challenge 1 59
   - 挑战 2：改进 tokenization / Challenge 2 67
   - 挑战 3：等长采样比较 / Challenge 3 74
   - 审查、验证与讨论 / Review and verification 81
   - 总结 85

1 语料库与 TTR / Corpus and TTR 5
- 语料库 / Corpus 6
- 词汇分析 / Lexical analysis 13

2 Linux 基础命令 / Commands in Linux 18

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46

定义：
Corpus 通常指：

1. （Cambridge Dict [1]）保存在计算机中的书面或口语材料集合，可用于研究语言如何被使用。词典中的大量例句通常来自数十亿词规模的语料库。
2. （Merriam-Webster [2]）知识或证据的集合，特别是记录下来的语言材料集合，可作为描述性语言分析的基础。
3. （Wikipedia [3]）大规模、结构化的文本集合，通常以电子形式存储和处理。在语料库语言学中，语料库可用于统计分析、假设检验、用例查找和语言规则验证。

[1] https://www.merriam-webster.com/dictionary/corpus
[2] https://dictionary.cambridge.org/zhs/%E8%AF%8D%E5%85%B8/%E8%8B%B1%E8%AF%AD/corpus
[3] https://en.wikipedia.org/wiki/Text_corpus

1. 多语语料库（multilingual corpus）通常整理成可并排比较的格式，常用于翻译研究与机器翻译。
2. 标注（annotation）是语料库的重要增值方式。例如词性标注（part-of-speech tagging, POS）会标出动词、名词、形容词等；词形还原（lemma）会标出每个词的基本形式。
3. 语料库还可以加入更深层的结构分析，例如句法分析（parsing）。
4. 各类语料库的分析与处理，是计算语言学（computational linguistics）、语音识别（speech recognition）和机器翻译（machine translation）的重要研究对象。



GENIA corpus 是 GENIA 项目构建并标注的核心生物医学文献语料库。它的目标是支持分子生物学领域的信息抽取（information extraction）和文本挖掘（text mining）系统的开发与评测。

该语料库包含 1,999 篇 Medline 摘要，来源于 PubMed 中与三个 MeSH 术语 "human"、"blood cells" 和 "transcription factors" 相关的检索结果。语料中加入了多个层次的语言学与语义标注。

GENIA corpus 的主要标注类别及其子语料包括：

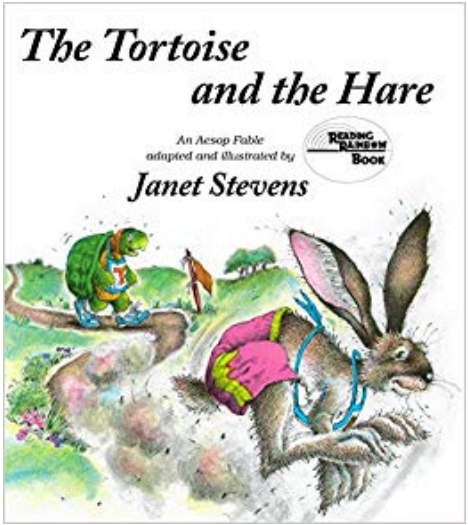
1. 词性标注（Part-of-Speech annotation）
2. 成分句法标注（Constituency / phrase structure syntactic annotation）
3. 术语标注（Term annotation）
4. 事件标注（Event annotation）
5. 关系标注（Relation annotation）

8. 共指标注（Coreference annotation）

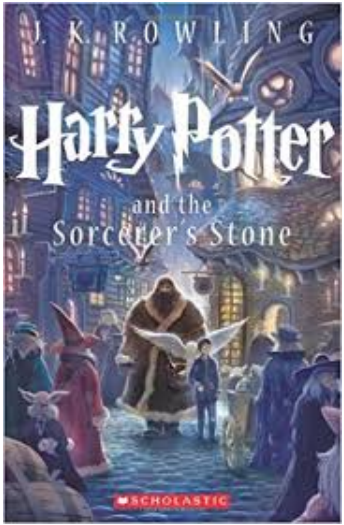


[4] AGAC 是由领域专家人工标注的语料库，目标是在致病语境中捕捉突变基因的功能变化。它覆盖与遗传变异、分子和细胞层面的后续表型变化相关的实体，重点追踪 LOF（loss-of-function，功能缺失）和 GOF（gain-of-function，功能获得）突变的生物学语义。

[4] https://sites.google.com/view/bionlp-ost19-agac-track

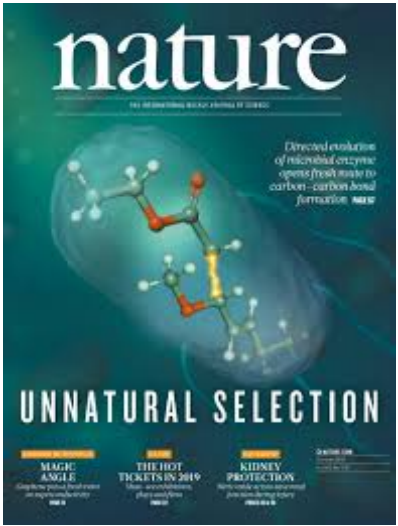

The Tortoise
and the Hare
An Aesop Fable
adapted and illustrated by
Janet Stevens
READING RAINBOW BOOK

VS.


J. K. ROWLING
Harry Potter
and the
Sorcerer's Stone
SCHOLASTIC


COPYCATS HIT
GREEN IS GOOD
CHINA DAILY
LEARNING ADVANTAGE

vs.


nature
Directed evolution of microbial enzyme opens fresh route to carbon–carbon bond formation
UNNATURAL SELECTION
MAGIC ANGLE
THE HOT TICKETS IN 2019
KIDNEY PROTECTION

1 语料库与 TTR / Corpus and TTR 5
- 语料库 / Corpus 6
- 词汇分析 / Lexical analysis 13

2 Linux 基础命令 / Commands in Linux 18

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46

在今天的任务中，词汇分析（lexical analysis）指的是：把原始文本转换为可以计数的词汇单位。

Raw text	→	Tokenization	→	Normalization	→	Counts
原始文本		分词/切词		小写化、去标点		tokens/types

示例问题：生物医学文本是否比通用英语文本包含更丰富的词汇？

1. 分词（tokenization）：什么算一个“词”？
2. 规范化（normalization）：Gene、gene 和 GENE 是否应视为同一个词？
3. 计数（counting）：总共有多少 tokens？有多少不重复的 types？

TTR (Type-Token Ratio，词型-词符比) 是文本中词型数量（types，不同词的总数）除以词符数量（tokens，所有词的总数）得到的比例。

$$\text{TTR} = \frac{\text{Amount of unique words}}{\text{Amount of total words}}.$$

较高的 TTR 通常表示词汇变化度更高；较低的 TTR 则说明文本中重复词更多、词汇多样性较低。

Text: gene mutation affects gene expression

1. Tokens（词符）：gene / mutation / affects / gene / expression
2. Token 数量：5
3. Types（词型）：gene, mutation, affects, expression
4. Type 数量：4

$$\mathrm{TTR} = \frac{4}{5} = 0.80.$$

这个小例子说明了任务的基本步骤：切分单词、进行规范化、统计所有词，再统计不重复词。

原始 TTR 有重要限制：它对文本长度敏感

当文本长度变化时，原始 TTR 往往不稳定。

1. 短文本常常会出现人为偏高的 TTR。
2. 长文本会重复常见功能词，因此 TTR 往往随长度增加而下降。
3. Brown 与 PubMed 不应只用全语料原始 TTR 直接比较。

更公平的比较方式：使用等长样本，例如 1000-token 窗口，再比较不同样本的平均 TTR。

因此，这个案例不仅是 shell 练习，也是一个方法学练习。

Outline

1. 语料库与 TTR / Corpus and TTR 5
   - 语料库 / Corpus 6
   - 词汇分析 / Lexical analysis 13
2. Linux 基础命令 / Commands in Linux 18
   - cat、tr、sort、uniq 与 wc 20
   - awk 与 sed 25
   - echo、read 与 IFS 29
   - while 循环 / While loop 33
3. 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37
   - 材料与任务 / Material and task 38
   - 代码与实践 / Codes 40
4. TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46
   - 从手写命令到智能体协作 / From manual Shell to agent collaboration 47
   - Prompt 是实验方案 / Prompt as experimental protocol 53
   - 挑战 1：生成完整 TTR 命令 / Challenge 1 59
   - 挑战 2：改进 tokenization / Challenge 2 67
   - 挑战 3：等长采样比较 / Challenge 3 74
   - 审查、验证与讨论 / Review and verification 81
   - 总结 85

1. 文件查看：`cat`, `head`, `wc`
2. 管道（pipeline）：`command1 | command2 | command3`
3. 重定向（redirection）：`>` 写入文件；`>>` 追加写入；`<` 从文件读取
4. 变量（variables）：`F_OUT=result.txt`；使用时写作 `$F_OUT`
5. 引号很重要：shell 命令对空格和特殊字符非常敏感

管道是核心思想：每个命令只做一件简单的事，把这些命令连接起来，就形成了可复现的分析流程。

1 语料库与 TTR / Corpus and TTR 5

2 Linux 基础命令 / Commands in Linux 18
- cat、tr、sort、uniq 与 wc 20
- awk 与 sed 25
- echo、read 与 IFS 29
- while 循环 / While loop 33

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46

http://www.tecmint.com/13-basic-cat-command-examples-in-linux/

Linux 命令 cat：

cat 是 concatenate 的缩写，是 Linux/Unix 系统中最常用的命令之一。它可以创建文件、查看文件内容、连接多个文件，并把输出重定向到终端或文件中。

http://www.linuxcommand.org/man_pages/tr1.html
https://www.cnblogs.com/badboy200800/p/10389973.html
Linux 命令 tr：
tr 用于转换或删除字符。

语法 / SYNOPSIS
tr [OPTION]... SET1 [SET2]

说明 / DESCRIPTION
从标准输入读取字符，对其进行转换、压缩重复或删除，并写入标准输出。

-c, –complement 对 SET1 先取补集
-d, –delete 删除 SET1 中的字符，不做转换
-s, –squeeze-repeats 将 SET1 中连续重复出现的字符压缩为一个
-t, –truncate-set1 先将 SET1 截断到 SET2 的长度

http://www.thegeekstuff.com/2013/04/sort-files
sort 命令用于对文本文件中的行进行排序。排序结果可以显示在屏幕上，也可以重定向到文件中。
根据需求，sort 提供了多种命令行选项。
sort 命令语法：
$ sort [-options]

```
$ printf "gene\nmutation\ngene\nexpression\n" > words.txt
$ cat words.txt | wc -l
4
$ cat words.txt | sort | uniq | wc -l
3
$ cat words.txt | sort | uniq -c
      1 expression
      2 gene
      1 mutation
```

1. `wc -l`：统计 token 行数。
2. `sort | uniq`：统计不重复的 word types。
3. `uniq -c`：生成一个简单的词频表。

1 语料库与 TTR / Corpus and TTR 5

2 Linux 基础命令 / Commands in Linux 18
- cat、tr、sort、uniq 与 wc 20
- awk 与 sed 25
- echo、read 与 IFS 29
- while 循环 / While loop 33

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46

https://www.runoob.com/linux/linux-comm-awk.html
AWK 是一种处理文本文件的语言，是一个强大的文本分析工具。
之所以叫 AWK，是因为它取自三位创始人 Alfred Aho、Peter Weinberger 和 Brian Kernighan 名字的首字符。
$ awk [选项参数] 'script' var=value file(s)

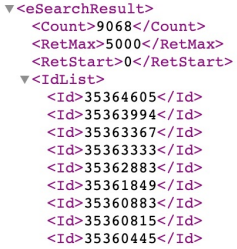
举例：
$ awk -F "\t" '{print $4 "\t " $7}' ***.tsv
或者
$ cat ***.tsv — awk -F "\t" '{print $4 "\t" $7}'
或者
$ cat ***.tsv — grep –color -e $'Gene\t' — awk -F "\t" '{print $4}'

sed 是流编辑器（stream editor），https://coolshell.cn/articles/9104.html。把 str1 替换为 str2 的基本写法如下：

```
$ sed 's/str1/str2/g' file
```

1. `s/str1/str2/g` 表示把所有 `str1` 替换为 `str2`；
2. 在文本挖掘任务中，`sed` 常用于删除标签、清理空白字符和替换固定模式；
3. 对 XML、HTML 或 API 返回文本做快速清理时，`sed` 常与 `grep`、`awk` 串联使用。

从 NCBI E-utilities 返回结果中提取 ID 列表。




```
$ cat ace2.txt | grep "<Id>" \
  | sed 's/<Id>//g' | sed 's/<\/Id>//g' \
  | sed 's/[[:blank:]]//g'
```

1 语料库与 TTR / Corpus and TTR 5

2 Linux 基础命令 / Commands in Linux 18
- cat、tr、sort、uniq 与 wc 20
- awk 与 sed 25
- echo、read 与 IFS 29
- while 循环 / While loop 33

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46

echo 是 Linux bash 和 C shell 中最常用的内置命令之一，常用于脚本或批处理文件，把一行文本或字符串输出到标准输出或文件中。[5]

输入一行文本，并把它显示到标准输出。
$ echo "It is fun."
输出如下文本：
It is fun.

声明一个变量，并输出它的值。例如，声明变量 x 并为其赋值。
$ x=3.1415926
$ echo "The value of variable x = $x."
The value of variable x = 3.1415926.

```
$ cd ~/Desktop
$ touch tmp.txt
$ F_OUT=tmp.txt
$ echo "This is fun." ¿$F_OUT
$ head tmp.txt
This is fun.
```

[5] https://www.tecmint.com/echo-command-in-linux/.

IFS= read -r line
命令含义: [6]
这行命令把 read 读取 buffer 时的域分隔符设置为空字符（IFS=），并在不解释反斜杠转义的前提下读取一行文本（read -r），赋值给变量 line。

```
$ IFS= read -r line ¡$F_OUT
$ echo $line
This is fun.
```

[6] https://blog.csdn.net/weixin_43958105/article/details/118726532.

1 语料库与 TTR / Corpus and TTR 5

2 Linux 基础命令 / Commands in Linux 18
- cat、tr、sort、uniq 与 wc 20
- awk 与 sed 25
- echo、read 与 IFS 29
- while 循环 / While loop 33

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46

语法 / Syntax. [7]
while CONTROL-COMMAND
do CONSEQUENT-COMMANDS
done

```
$ i=0
$ while [ $i -lt 4 ]
$ do
$ i=$[i+1]
$ done
$ echo $i
4
```

注意

`while [ $i -lt 4 ]` 中的测试表达式对空格敏感。`[` 后面和 `]` 前面都要保留空格。

```
$ F_OUT="result.txt"
$ F_LIST="pmid-list.txt"
$ echo -e "\n I am curating the result. \n"
$ echo -e "\n" ¿$F_OUT
$ i=1
$ while IFS= read -r line
$ do
$ xx=‘echo https://www.ncbi.nlm.nih.gov/research/pubtator-api/publications/
export/pubtator?pmids=$line‘
$ curl $xx ¿¿$F_OUT
$ printf "$i -th result out of xxxxx is processing...\n"
$ i=$[i+1]
$ sleep 5.8s
$ done ¡"$F_LIST"
```

BioNLP 应用场景 / BioNLP use case

这是一个从 PMID 列表自动批量获取 PubTator 记录的终端流程。同样的思想也可以用于大规模文献整理（literature curation）。

[7] https://tldp.org/LDP/Bash-Beginners-Guide/html/sect_09_02.html.

Outline

1. 语料库与 TTR / Corpus and TTR 5
   - 语料库 / Corpus 6
   - 词汇分析 / Lexical analysis 13
2. Linux 基础命令 / Commands in Linux 18
   - cat、tr、sort、uniq 与 wc 20
   - awk 与 sed 25
   - echo、read 与 IFS 29
   - while 循环 / While loop 33
3. 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37
   - 材料与任务 / Material and task 38
   - 代码与实践 / Codes 40
4. TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46
   - 从手写命令到智能体协作 / From manual Shell to agent collaboration 47
   - Prompt 是实验方案 / Prompt as experimental protocol 53
   - 挑战 1：生成完整 TTR 命令 / Challenge 1 59
   - 挑战 2：改进 tokenization / Challenge 2 67
   - 挑战 3：等长采样比较 / Challenge 3 74
   - 审查、验证与讨论 / Review and verification 81
   - 总结 85

1 语料库与 TTR / Corpus and TTR 5

2 Linux 基础命令 / Commands in Linux 18

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37
- 材料与任务 / Material and task 38
- 代码与实践 / Codes 40

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46

案例任务 / Case study task

比较 Brown corpus 与 PubMed 文本的 TTR，分析通用英语文本与生物医学文本在词汇复杂度上的差异。

资源 / Resources:

1. Brown.
   - Brown University Standard Corpus of Present-Day American English（简称 Brown Corpus）由 Henry Kučera 和 W. Nelson Francis 于 20 世纪 60 年代在 Brown University 编制，是语料库语言学中的通用英语语料库（general corpus）。
   - 它包含 500 个英语文本样本，总计约一百万词，文本来源于 1961 年在美国出版的作品。
   - 文本由美国英语母语者写作。
2. PubMed
   - 生物医学论文文本。
   - 作者来源更复杂，写作背景也更不一致。

1 语料库与 TTR / Corpus and TTR 5

2 Linux 基础命令 / Commands in Linux 18

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37
- 材料与任务 / Material and task 38
- 代码与实践 / Codes 40

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46

```
$cat pubmed-a.txt |tr -cs "[:alnum:]" "\n" |tr [:upper:] [:lower:]
>pubmed.a.pure.txt
$wc pubmed.a.pure.txt
1032975 1032975 6634213
$sort pubmed.a.pure.txt |uniq |wc
31638   31638  282238
```

So the token/word ratio is 31638/1032975 = 3.06%.

```
$cat BROWN_A.txt |tr -cs "[:alnum:]" "\n" |tr [:upper:] [:lower:]
>BROWN_A.pure.txt
$wc BROWN_A.pure.txt
91064   91063  515866
$sort BROWN_A.txt |uniq |wc
11927   11926   95306
```

So the token/word ratio is 11927/91064 =13.09%.

But this is an unfair comparison.

```
$cat pubmed-a.txt |tr -cs "[:alnum:]" "\n" |tr [:upper:] [:lower:] |split -1000
$ wc xaa
1000     999    5935 xaa
$sort xaa |uniq |wc
458     457    3307

$cat pubmed-a.txt |tr -cs "[:alnum:]" "\n" |tr [:upper:] [:lower:] |split -2000
$wc xaa
2000    1999   11759 xaa
$ sort xaa |uniq |wc
782     781    5614
```

课堂挑战 1 / In-class challenge 1

能否写一个一体化 shell 脚本，同时完成 token count、type count 和 TTR 计算？

局部示例：下面这条命令只统计 tokens，还没有计算 TTR。

```
cat test.txt — tr -cs "[:alnum:]" "\n" —tr [:upper:] [:lower:] —wc — awk '{print $1}'
```

课堂挑战 2 / In-class challenge 2

能否使用 sed、更好的正则表达式（regular expressions），或领域规则（domain rules）来改进 tokenization？

课堂挑战 3 / In-class challenge 3

能否通过等长采样（equal-length sampling）对两个语料库进行统计比较？

Outline

1. 语料库与 TTR / Corpus and TTR 5
   - 语料库 / Corpus 6
   - 词汇分析 / Lexical analysis 13
2. Linux 基础命令 / Commands in Linux 18
   - cat、tr、sort、uniq 与 wc 20
   - awk 与 sed 25
   - echo、read 与 IFS 29
   - while 循环 / While loop 33
3. 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37
   - 材料与任务 / Material and task 38
   - 代码与实践 / Codes 40
4. TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46
   - 从手写命令到智能体协作 / From manual Shell to agent collaboration 47
   - Prompt 是实验方案 / Prompt as experimental protocol 53
   - 挑战 1：生成完整 TTR 命令 / Challenge 1 59
   - 挑战 2：改进 tokenization / Challenge 2 67
   - 挑战 3：等长采样比较 / Challenge 3 74
   - 审查、验证与讨论 / Review and verification 81
   - 总结 85

1 语料库与 TTR / Corpus and TTR 5

2 Linux 基础命令 / Commands in Linux 18

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46
- 从手写命令到智能体协作 / From manual Shell to agent collaboration 47
- Prompt 是实验方案 / Prompt as experimental protocol 53
- 挑战 1：生成完整 TTR 命令 / Challenge 1 59
- 挑战 2：改进 tokenization / Challenge 2 67
- 挑战 3：等长采样比较 / Challenge 3 74
- 审查、验证与讨论 / Review and verification 81
- 总结 85



前半部分我们已经手写了若干 Linux 管道命令。现在的问题不再是“是否能记住命令”，而是：

1. 能否把 BioNLP 任务清楚地表达给智能体？
2. 能否让智能体生成可运行、可解释、可复现的 shell 命令？
3. 能否检查智能体生成的命令是否正确、安全、适合当前数据？

本部分的核心能力：**Prompt** → **命令** → **运行** → **验证** → **迭代**。



完成本部分后，学生应能够：

1. 说明 TRAE 在本课中的作用：把自然语言任务转化为可检查的计算流程；
2. 写出结构化 prompt，明确任务、输入、输出、约束和验证要求；
3. 针对三个课堂挑战，组织 prompt 让 agent 生成 shell 命令；
4. 阅读并审查 agent 输出，判断命令是否正确、稳健和安全；
5. 在小样例上验证命令，再推广到 Brown、PubMed、GENIA 和 AGAC。

教师强调

TRAE 是协作者，不是替代者。学生必须能解释、验证并修正它给出的命令。



TRAE 可被理解为一组 AI 原生的工作与编码智能体工具。产品形态会持续变化，课堂中只需把握它与本课有关的能力。

形态	本课中的理解方式
TraeWork	面向任务协作的 AI 工作台，可通过对话组织需求与生成方案。
TraeCode	面向代码开发的 AI IDE，可结合项目文件、终端与编辑器完成编码任务。
TraeCode CLI	命令行中的 Code Agent，适合与 Linux / Shell 工作流结合。
TraeCode Plugin	将 TRAE 能力接入 VS Code 或 JetBrains 等现有 IDE。

官方入口可参考：https://www.trae.cn/，https://www.trae.ai/，https://docs.trae.cn/。实际版本与功能以官方文档为准。

可以帮助

1. 把自然语言任务转成 shell 命令；
2. 解释 `cat`, `tr`, `sort`, `uniq`, `awk` 的作用；
3. 给出多种 tokenization 方案；
4. 帮助写循环、采样和结果汇总脚本；
5. 指出潜在边界情况。

不能替代

1. 不能替你定义科学问题；
2. 不能保证命令一定正确；
3. 不能替你承担数据安全责任；
4. 不能自动判断 BioNLP token 是否合理；
5. 不能替你解释最终结果。



步骤	学生要做什么
提出任务	把 BioNLP 问题说清楚：输入是什么，输出是什么，TTR 如何定义。
组织 prompt	写出任务、数据、约束、输出格式、验证要求。
获得命令	让 TRAE 生成 shell 命令或脚本。
人工审查	逐段读命令，检查 tokenization、文件操作、安全风险。
本地验证	用小样例手算，再在真实语料上运行。
迭代修正	把发现的问题反馈给 TRAE，要求它修改命令。

科研计算中的 agent 使用，不是“一问一答”，而是一个可记录、可复现、可追责的实验流程。



1 语料库与 TTR / Corpus and TTR 5
2 Linux 基础命令 / Commands in Linux 18
3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37
4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46
- 从手写命令到智能体协作 / From manual Shell to agent collaboration 47
- Prompt 是实验方案 / Prompt as experimental protocol 53
- 挑战 1：生成完整 TTR 命令 / Challenge 1 59
- 挑战 2：改进 tokenization / Challenge 2 67
- 挑战 3：等长采样比较 / Challenge 3 74
- 审查、验证与讨论 / Review and verification 81
- 总结 85



帮我算 TTR。

这个 prompt 的问题：

1. 没有说明 TTR 的定义：types/tokens 还是 tokens/types？
2. 没有说明输入文件名和文本格式；
3. 没有说明如何处理大小写、标点、数字和空行；
4. 没有说明需要一条命令还是完整脚本；
5. 没有要求解释命令，也没有要求说明边界情况；
6. 没有安全约束，例如不要覆盖原文件。

课堂提问

如果你是 agent，看到这个 prompt，你会补哪些假设？这些假设可能错在哪里？



1. **背景 / Context**：这是 BioNLP 文本分析任务，不是普通写作任务。
2. **任务 / Task**：计算 TTR，并明确 TTR = types/tokens。
3. **输入 / Input**：文件名、编码、语言、分隔方式、样例。
4. **输出与约束 / Output and constraints**：Shell 管道、脚本、表格、是否保留中间文件。
5. **验证 / Verification**：要求给出解释、测试样例、边界情况和安全提醒。

Prompt 不是“求助短信”，而是“实验方案”。写得越清楚，生成结果越可复现。



背景：我要分析英文生物医学文本的词汇复杂度。
任务：计算 TTR。定义：TTR = types/tokens，其中 types 是去重后的单词数，tokens 是总单词数。
输入：文件名为 `input.txt`，UTF-8，英文文本，单词可能包含数字、连字符和大小写差异。
要求：请生成可在 Linux shell 中运行的命令；需要完成分词、小写化、去空行、统计 tokens、types 和 TTR。
约束：不要修改原文件；不要使用危险命令；如果需要中间文件，请说明文件名。
验证：请解释每一步命令，并说明如何用小样例验证结果；指出边界情况。

教师提示

后面的三个挑战都用这个模板，只改变任务重点。



Shell 命令有真实的文件系统影响。让 agent 生成命令时，prompt 中应主动加入安全边界：

1. 不要删除文件，不要使用 `rm` 或 `rm -rf`；
2. 不要覆盖原始数据文件；
3. 如果使用 `>` 重定向，必须写入新文件；
4. 不要自动下载或上传数据，除非我明确要求；
5. 先给命令和解释，等我确认后再运行；
6. 如果命令依赖软件包，请说明依赖，不要自动安装。

学生要养成习惯：凡是 agent 给出的 shell 命令，运行前先读一遍。



1. 每个挑战先展示一个 prompt；
2. 再展示一个“TRAE 返回示例”；
3. 学生先不运行，先逐段读命令；
4. 小组讨论：这条命令哪里正确？哪里可能不适合 BioNLP？
5. 用小样例验证，再决定是否用于 Brown/PubMed；
6. 最后把问题反馈给 TRAE，要求它迭代修改。

课堂分工建议

每组 2–3 人：一人读 prompt，一人读 shell 命令，一人负责设计验证样例。



1 语料库与 TTR / Corpus and TTR 5

2 Linux 基础命令 / Commands in Linux 18

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46
- 从手写命令到智能体协作 / From manual Shell to agent collaboration 47
- Prompt 是实验方案 / Prompt as experimental protocol 53
- 挑战 1：生成完整 TTR 命令 / Challenge 1 59
- 挑战 2：改进 tokenization / Challenge 2 67
- 挑战 3：等长采样比较 / Challenge 3 74
- 审查、验证与讨论 / Review and verification 81
- 总结 85



课堂挑战 1

能否写一个一体化 shell 脚本，同时完成 token count、type count 和 TTR 计算？

前半部分已有局部命令：

```
cat test.txt | tr -cs "[:alnum:]" "\n"
| tr [:upper:]  [:lower:]  | wc
| awk '{print $1}'
```

它的问题：

1. 只计算了 tokens，没有计算 types；
2. 没有输出 TTR；
3. 没有解释中间步骤；
4. 没有说明如何验证。



挑战 1：生成完整 TTR 命令

Prompt 版本对比

模糊版

帮我算 test.txt 的 TTR。

可能导致：定义不清、输出格式不清、忽略小写化、忽略验证、难以复现。

结构化版

我要计算 `test.txt` 的 TTR，定义为 types/tokens。请用 Linux shell 管道完成分词、小写化、去空行、统计 tokens/types/TTR，并解释每一步。

优势：定义明确、文件明确、处理步骤明确、输出明确、可验证。



我要分析一个英文文本文件的词汇复杂度 TTR（Type-Token Ratio）。定义：TTR = 去重后的单词数（types）除以总单词数（tokens）。输入文件为 `test.txt`，UTF-8 编码，英文文本，单词可能被空格或标点分隔。请用一条 Linux shell 管道命令完成：分词、转小写、去掉空行、统计 tokens、统计 types，并输出 types、tokens 和 TTR，TTR 保留 3 位小数。不要修改原始文件。请逐段解释每个命令，并说明我如何用一个小样例验证结果。

教师提示

这里把“任务定义”和“输出格式”写得很死，是为了避免 agent 自行发挥。



挑战 1：TRAE 返回示例

Agent 返回的一条可运行命令

```
cat test.txt \
  | tr -cs '[:alpha:]' '\n' \
  | tr '[:upper:]' '[:lower:]' \
  | grep -v '^$' \
  | sort | uniq -c \
  | awk '{types++; tokens+=$1} END {
      printf "types=%d tokens=%d TTR=%.3f\n",
      types, tokens, types/tokens
    }'
```

注意

这条命令是“教学示例”：基本能完成任务，但不是完美答案。我们要训练的是审查能力。



挑战 1：读懂命令

逐段解释 agent 的返回

片段	作用
`cat test.txt`	读取输入文件。
`tr -cs '[:alpha:]' '\n'`	把非字母字符当作分隔符，切成一行一个词。
`tr '[:upper:]' '[:lower:]'`	将大写转成小写，合并 Gene/gene/GENE。
`grep -v '^$'`	删除空行。
`sort | uniq -c`	排序后统计每个词出现次数。
`awk '{...}'`	累加 types 与 tokens，并计算 TTR。



挑战 1：用小样例验证

先手算，再运行

```
Text: gene mutation affects gene expression
Tokens: gene / mutation / affects / gene / expression
tokens = 5
types  = 4
TTR    = 4/5 = 0.800
```

```
printf "gene mutation affects gene expression\n" > test.txt
```

运行 agent 给出的命令，检查是否输出：

```
types=4 tokens=5 TTR=0.800
```



挑战 1：讨论与迭代

这条命令有哪些 BioNLP 风险？

1. `[:alpha:]` 只保留字母，会丢掉数字；
2. `COVID-19` 可能变成 `covid`，数字 19 被丢掉；
3. `IL-6` 可能被切成 `il`；
4. `NF-kB` 可能被切成 `nf` 和 `kb`；
5. 撇号、希腊字母、上标下标、基因突变写法都可能出问题。

给 TRAE 的迭代 prompt

这条命令会丢掉 `COVID-19`, `IL-6`, `p53` 中的数字。请改写命令，保留字母和数字，并解释这种修改的影响。



1 语料库与 TTR / Corpus and TTR 5

2 Linux 基础命令 / Commands in Linux 18

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46
- 从手写命令到智能体协作 / From manual Shell to agent collaboration 47
- Prompt 是实验方案 / Prompt as experimental protocol 53
- 挑战 1：生成完整 TTR 命令 / Challenge 1 59
- 挑战 2：改进 tokenization / Challenge 2 67
- 挑战 3：等长采样比较 / Challenge 3 74
- 审查、验证与讨论 / Review and verification 81
- 总结 85



课堂挑战 2

能否使用 sed、更好的正则表达式（regular expressions），或领域规则（domain rules）来改进 tokenization？

在 BioNLP 中，什么算一个 token 并不简单：

1. `IL-6` 是一个 token，还是 `IL` 和 `6`？
2. `COVID-19` 是否应保留连字符？
3. `NF-kB` 中大小写是否携带生物学信息？
4. `p53`, `BRCA1`, `TNF-alpha` 如何处理？

tokenization 决策会直接改变 tokens、types 和 TTR，因此它是方法学问题。



我正在处理英文生物医学文本，目标是计算 TTR。请给出三种 tokenization 的 shell 实现，并比较它们的优缺点：
1. 只保留字母；
2. 保留字母和数字；
3. 尽量保留生物医学术语中的连字符，例如 `COVID-19`, `IL-6`, `TNF-alpha`。
请分别给出对应的 shell 管道片段，并说明它们对 tokens、types 和 TTR 的影响。不要修改原始文件。

教师提示

这里不是要求学生立即接受 agent 的方案，而是让 agent 提供“可比较的候选方法”。



挑战 2：TRAE 返回示例

三种 tokenization 策略

策略	规则	适合讨论的问题
字母型	只保留 `A-Z/a-z`	简单，但丢掉 p53, IL-6 中的数字。
字母数字型	保留 `A-Z/a-z/0-9`	更适合基因名和疾病名，但会把连字符切开。
领域友好型	保留词内连字符	更贴近 BioNLP 术语，但命令更复杂，规则需说明。

没有绝对正确的 tokenization。关键是：规则要明确、结果要可复现、局限要写出来。



挑战 2：TRAE 返回示例

三种 shell 片段

```
# 1. 只保留字母
tr -cs '[:alpha:]' '\n' < input.txt

# 2. 保留字母和数字
tr -cs '[:alnum:]' '\n' < input.txt

# 3. 尝试保留词内连字符
sed "s/[^[:alnum:]-]/\n/g" input.txt \
  | sed '/^$/d' \
  | sed 's/^-//; s/-$//'
```

课堂审查点

第三种规则能保留 COVID-19 和 IL-6，但也可能保留不该保留的连字符。它是“更好”，还是“更复杂但不一定更稳”？

挑战 2：构造边界样例

用边界样例检验 tokenization

```
printf "IL-6 activates NF-kB. COVID-19 affects p53 and BRCA1.\n" \
  > bio_test.txt
```

学生需要比较三种规则的输出：

1. IL-6 是否被拆开？
2. NF-kB 是否保持为一个词？
3. p53 与 BRCA1 是否保留数字？
4. 输出是否出现空 token？
5. 不同规则下 TTR 会如何变化？

讨论题

如果你的研究对象是基因、疾病和突变，哪种 tokenization 更合理？理由是什么？



不同 tokenization 会改变：

1. token 总数；
2. type 总数；
3. 高频词表；
4. TTR；
5. Brown 与 PubMed 的比较结论。

写报告时必须说明 tokenization 规则。否则，别人无法复现实验，也无法判断结论是否可信。

给 TRAE 的迭代 prompt

请基于“保留字母、数字和词内连字符”的规则，生成完整 TTR 计算命令，并说明这种规则相对于 `[:alpha:]` 的优缺点。



1 语料库与 TTR / Corpus and TTR 5

2 Linux 基础命令 / Commands in Linux 18

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46
- 从手写命令到智能体协作 / From manual Shell to agent collaboration 47
- Prompt 是实验方案 / Prompt as experimental protocol 53
- 挑战 1：生成完整 TTR 命令 / Challenge 1 59
- 挑战 2：改进 tokenization / Challenge 2 67
- 挑战 3：等长采样比较 / Challenge 3 74
- 审查、验证与讨论 / Review and verification 81
- 总结 85



课堂挑战 3

能否通过等长采样（equal-length sampling）对两个语料库进行统计比较？

为什么需要它？

1. 原始 TTR 对文本长度敏感；
2. 短文本容易得到偏高 TTR；
3. 长文本会重复常见词，TTR 往往降低；
4. Brown 与 PubMed 的总词数未必相同，直接比较全语料 TTR 不公平。

更公平的比较：把每个语料切成相同 token 数的窗口，例如每 1000 tokens 一个 window，再比较窗口 TTR 的均值和分布。



我有两个英文文本文件：`brown.txt` 和 `pubmed.txt`。我要公平比较它们的 TTR。请写一个 Linux shell 脚本：

1. 对每个文件进行 tokenization、小写化、去空行；
2. 将 token 列表按每 1000 tokens 切成窗口；
3. 对每个窗口计算 tokens、types 和 TTR；
4. 输出一个 TSV 表格，列为 corpus, window, types, tokens, TTR；
5. 不要覆盖原始文件；
6. 解释如何检查最后不足 1000 tokens 的窗口。



挑战 3：TRAE 返回示例

等长窗口脚本

```
for corpus in brown pubmed; do
  infile="${corpus}.txt"
  outdir="${corpus}_windows"
  mkdir -p "$outdir"

  cat "$infile" \
    | tr -cs '[:alnum:]' '\n' \
    | tr '[:upper:]' '[:lower:]' \
    | sed '/^$/d' \
    | split -l 1000 - "$outdir/window_"

  for f in "$outdir"/window_*; do
    tokens=$(wc -l < "$f")
    types=$(sort "$f" | uniq | wc -l)
    awk -v c="$corpus" -v w="$(basename "$f")" \
        -v t="$types" -v n="$tokens" \
        'BEGIN {printf "%s\t%s\t%d\t%d\t%.4f\n", c,w,t,n,t/n}'
  done
done
```

挑战 3：审查 agent 脚本

这段脚本需要检查什么？

1. `mkdir -p` 是否只创建新目录？是否安全？
2. `split -l 1000` 是否按 1000 tokens 切分？
3. 最后一个 window 如果不足 1000 tokens，是否应该保留？
4. `wc -l < "$f"` 与 `cat $f | wc -l` 有何差异？
5. 输出是否有表头？没有表头是否影响后续分析？
6. Brown 与 PubMed 的 window 数量不同，比较时如何处理？

课堂讨论

如果 PubMed 有 500 个窗口，Brown 只有 80 个窗口，应该比较全部窗口、随机抽取 80 个，还是报告不同样本量下的均值与方差？



挑战 3：改进输出格式

给结果表加表头

可以要求 TRAE 迭代脚本，增加 TSV 表头：

```
echo -e "corpus\twindow\ttypes\ttokens\tTTR" > ttr_windows.tsv

# 在循环中把输出追加到同一个表格
awk -v c="$corpus" -v w="$(basename "$f")" \
    -v t="$types" -v n="$tokens" \
    'BEGIN {printf "%s\t%s\t%d\t%d\t%.4f\n", c,w,t,n,t/n}' \
    >> ttr_windows.tsv
```

安全审查

> 会覆盖 `ttr_windows.tsv`。这可以接受吗？如果不能，应如何改成带时间戳的新文件名？

挑战 3：结果解释

等长采样之后如何解释？

学生最终不应只提交一个 TTR 数字，而要报告：

1. 每个语料的窗口数量；
2. 每个窗口的 tokens 是否约为 1000；
3. TTR 均值；
4. TTR 标准差或范围；
5. Brown 与 PubMed 差异是否稳定；
6. tokenization 规则如何影响结论。

方法学提醒：如果只看一个全语料 TTR，可能把“文本长度差异”误解为“词汇复杂度差异”。



1 语料库与 TTR / Corpus and TTR 5

2 Linux 基础命令 / Commands in Linux 18

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46
- 从手写命令到智能体协作 / From manual Shell to agent collaboration 47
- Prompt 是实验方案 / Prompt as experimental protocol 53
- 挑战 1：生成完整 TTR 命令 / Challenge 1 59
- 挑战 2：改进 tokenization / Challenge 2 67
- 挑战 3：等长采样比较 / Challenge 3 74
- 审查、验证与讨论 / Review and verification 81
- 总结 85



1. **可读性**：每个命令和参数能解释吗？
2. **正确性**：是否符合 TTR = types/tokens 的定义？
3. **稳健性**：空行、标点、大小写、数字、连字符如何处理？
4. **安全性**：是否删除、覆盖、上传、下载或执行未知脚本？
5. **结果合理性**：输出是否符合小样例手算结果？用于真实语料是否合理？



运行 agent 生成的命令前，先看有没有：

1. `rm`, `rm -rf`：删除文件；
2. `mv`：移动或覆盖重要文件；
3. `>`：覆盖已有结果文件；
4. `curl | sh` 或 `wget | sh`：下载并直接执行脚本；
5. 未解释的网络请求；
6. 对系统目录或上级目录的写操作。

注意：任何不能解释的 shell 命令，都不要运行。



建议学生在实验记录中保留四类内容：

1. 原始 prompt；
2. agent 返回的原始命令；
3. 自己做过的修改；
4. 验证样例与运行结果。

在智能体时代，prompt 也是方法的一部分。没有 prompt，实验很难复现。



1 语料库与 TTR / Corpus and TTR 5

2 Linux 基础命令 / Commands in Linux 18

3 案例研究：Brown 语料库与 PubMed 文本比较 / Case study 37

4 TRAE 智能体辅助 Shell 实验 / Agent-assisted Shell 46
- 从手写命令到智能体协作 / From manual Shell to agent collaboration 47
- Prompt 是实验方案 / Prompt as experimental protocol 53
- 挑战 1：生成完整 TTR 命令 / Challenge 1 59
- 挑战 2：改进 tokenization / Challenge 2 67
- 挑战 3：等长采样比较 / Challenge 3 74
- 审查、验证与讨论 / Review and verification 81
- 总结 85



1. **Question**：我要完成什么？
2. **Data**：数据来源、format、规模。
3. **Prompt log**：关键 prompt 与 agent 返回。
4. **Shell workflow**：最终采用的命令或脚本。
5. **Verification**：小样例手算与命令输出是否一致。
6. **Result**：结果可靠性，一致性，SOTA比较。
7. **Discussion**：差异原因、因素影响、方法局限。



维度	思考点
Prompt 质量	是否明确任务、定义、输入、输出、约束和验证要求。
命令理解	是否能解释每段 shell 命令，而不是只贴结果。
验证能力	是否用小样例验证 tokens、types 和 TTR。
方法公平性	是否考虑长度敏感性、等长采样和 tokenization 规则。
安全意识	是否避免危险命令、原文件覆盖和未知网络操作。
结果解释	是否能把结果回到 BioNLP 文本特点，而不是只报数字。



1. 手写 shell 命令帮助我们理解计算过程；
2. TRAE 可以帮助生成候选命令，提高探索效率；
3. prompt 要像实验方案一样写清楚；
4. agent 输出必须经过人工审查；
5. 小样例验证是最低要求；
6. 最终科学解释仍由研究者负责。

智能体时代的 BioNLP 学习目标：不是少学命令，而是更会提出问题、更会验证工具、更会解释结果。



讨论

