

# 自然语言处理

WELCOME!

杨海钦

2025-2026-1学期

# 面试问题

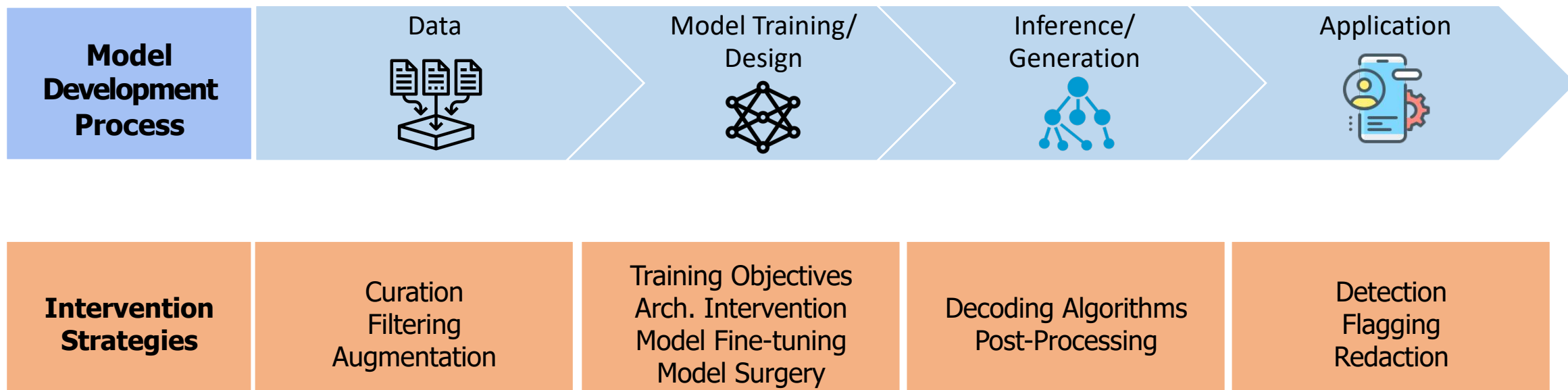
这是一个生物化学文本预处理项目，你计划如何处理？

- 特别的，蛋白质化学名称很长，如肌联蛋白的完整化学名称，由多个氨基酸残基拼接而成，常因超长成为词库外词 OOV
- 从解决词库外词、避免数据稀疏性、保障后续 NLP 任务（如检索、分析）可用性的角度，你会提出哪些高效的处理方案？

# 大纲

- 切词/Tokenization
- 早期解决方案：词语规范化
- 目前解决方案：子词模型/Subword Models

# 自然语言处理系统开发流程



Sachin Kumar, Vidhisha Balachandran, Lucille Njoo, Antonios Anastasopoulos, Yulia Tsvetkov. [“Language Generation Models Can Cause Harm: So What Can We Do About It? An Actionable Survey.” EACL 2023.](#)

# 为什么切词/Tokenization?

- 例子
  - 输入: “Friends, Romans and Countrymen”
  - 输出: 词元/Tokens
    - Friends
    - Romans
    - and
    - Countrymen
- 切词/Tokenization: 将类词单元从文本中分离出来
  - 文本处理的构建要素
- 切词的准确性会影响其他高级处理的结果, 例如: 语法分析/Parsing

# 切词的挑战

- 如何定义词元？
  - United States
  - AT&T
  - 3-year-old
  - Finland's capital
  - Hewlett-Packard
- 标点符号分句带有模糊性
  - Prof. Dr. J.M.

# 切词的挑战： 数字

- 例子：
  - 3/20/25      Mar. 20, 2025      20/3/20
  - 55 B.C.
  - B-52
  - commit 324a3df234cb23e
  - (800) 234-2333
- 观察
  - 通常有对应的表示
  - 创建日期、格式等
- 问题： 如何快速检索？

# 切词的挑战：语言

- 法语
  - *L'ensemble* → 一个词元还是两个词元
    - *L? L' L'e?*
  - 匹配: *l'ensemble*和*un ensemble*
- 德语名词复合词不分隔
  - *Lebensversicherungsgesellschaftsangestellter*
  - 人寿保险公司雇员
- 中文和日语没有用空格分词
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
- 日语: 平假名、片假名、汉字、罗马数字混用
  - フォーチュン500社は情報不足のため時間あた\$500K(約7,375万円)
  - 由于信息不足，《财富》500强企业每小时损失50万美元(约合7,375万日元)

# 切词的挑战：语言

- 阿拉伯语（或希伯来语）基本上是从右向左书写的，但也有一些项目，比如数字从左向右书写
- 单词是分开的，但单词中的字母形式形成复杂的连接
- 例子

حصلت الجزائر على استقلالها عام 1962 بعد 132 عاماً من الاحتلال الفرنسي.  
←开始      ↔      ↔

- 阿尔及利亚在被法国占领132年后，于1962年获得独立。
- 使用Unicode，表面表示很复杂，但存储的形式很简单

# 早期解决方案：词语规范化

Normalization to Terms (部署于信息检索系统)

# 文本预处理

1. 删除HTML标签 Removing HTML tags
2. 删除额外的空格和换行符 Remove extra whitespace and newlines
3. 将重音字符转换为ASCII码  
Convert accented characters to ASCII
4. 删除特殊字符和符号 (如: 数字)  
Remove special characters and symbols (optionally numbers)
5. 词干化或词形化 Stemming OR Lemmatization
6. 删除停止词 Removing stop words
7. 必要时进行词语切分  
Tokenization if needed
8. 拼写检查和语法检查 Spell check & grammar check

# 安装依赖关系

```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('averaged_per
ceptron_tagger')
```

```
# 导入自然语言处理工具库NLTK
# 下载分词模型
# 下载词形还原词典
# 下载停用词表
# 下载词性标注模型
```

[https://github.com/hqyang/nlp-codes/blob/main/02-tokenization/text\\_preprocessing.ipynb](https://github.com/hqyang/nlp-codes/blob/main/02-tokenization/text_preprocessing.ipynb)

<https://github.com/codespaces>

# 自然語言工具包 (Natural Language Toolkit, NLTK)

- 传统NLP任务的类套件
  - 语法分析 Parsing
  - 词性标注 Part-of-Speech (POS)
  - 分类器 Classifiers
- 文本处理工具
  - 用 “punkt” 切分句子
- 文本语料库
  - Brown
  - Penn Treebank语料库
  - ...

```
sent_tokens = nltk.sent_tokenize(text)
print(f'sent_tokens:\n {sent_tokens}')

word_text = nltk.word_tokenize(text)
print(f'word_text:\n {word_text}')

print(f'split:\n {text.split()}')
```

```
word_text: [..., 'field', '!', ...]
split: [..., 'field!', ...]
```

# 使用正则表达式

- 正则表达式(RE): 用于描述一组字符串的代数符号; 见[SLP]2.1

RE	Match	Example Patterns Matched
/[A-Z]/	an upper case letter	“we should call it ‘ <u>D</u> renched Blossoms’ ”
/[a-z]/	a lower case letter	“ <u>m</u> y beans were impatient to be hoed!”
/[0-9]/	a single digit	“Chapter <u>1</u> : Down the Rabbit Hole”

RE	Match	Example Patterns Matched
/woodchucks?/	woodchuck or woodchucks	“ <u>woodchuck</u> ”
/colou?r/	color or colour	“ <u>color</u> ”

# 去除HTML标签和噪音

## Removing HTML Tags & Noise

```
import re
from bs4 import BeautifulSoup
```

```
def strip_html_tags(text):
    soup = BeautifulSoup(text, "html.parser")
    [s.extract() for s in soup(['iframe',
    'script'])]
    stripped_text = soup.get_text()
    stripped_text = re.sub(r'[\r|\n|\r\n|'+
    '\n', stripped_text)
    return stripped_text
```

```
>>> clean_content = strip_html_tags(content)
>>> print(clean_content[953:1452])
```

- 输入的text (HTML 字符串) 转换为一个可操作的解析树 (soup对象)
- 删除特定标签及其内容
- 从处理后的解析树中提取所有文本内容
- 将所有匹配到的换行符组合替换为标准换行符: \n

# 删除特殊字符和符号（如：数字） Remove special characters and symbols (optionally numbers)

```
def remove_accented_chars(text):  
    text = unicodedata.normalize('NFKD',  
text).encode('ascii',  
'ignore').decode('utf-8', 'ignore')  
    return text
```

- unicodedata模块用于处理Unicode 字符相关的操作
  - normalize 函数的作用是将Unicode 字符串标准化，
  - 标准化形式：NFKD

```
>>> remove_accented_chars("café cliché  
naïve")  
'cafe cliche naive'
```

```
>>> remove_accented_chars("àèìòù ÁÉÍÓÚ ñ  
ç")  
'aeiou AEIOU n c'
```

# 中文处理

- “简体”
- “繁体”
- “臺灣正體”
- “香港繁體”
- 简繁分歧词
- 一简对多繁
- 一繁对多简

- 常用工具包: **opencc**
- `cc = OpenCC('t2s')`
- `traditional_text = "中華人民共和國，臺灣是中國的一部分。"`
- `simplified_text = cc.convert(traditional_text)`

# 主题词表(Thesauri)和语音索引法(Soundex)

- 问题1: 如何处理同义词synonyms和同音异义词homonyms
  - Car=Automobile,
  - Color = Colour
- 问题2: 拼写错误
- [Wordnet](#)
- 语音检索法Soundex
  - 保留第一个字母, 去掉其他位置的所有a, e, i, o, u, y, h, w
  - 将剩余的字母按照以下规则替换为数字(除非相邻的字母是相同的数字, 则只保留一个):
    - b, f, p, v → 1
    - c, g, j, k, q, s, x, z → 2
    - d, t → 3
    - l → 4
    - m, n → 5
    - r → 6
  - 如果结果长度不足4, 则在后面补0, 如果超过4, 则截断到4位

# 词根化 Lemmatization

- 词根化：用词根代替单词
  - am, are, is → be
  - computers → computer
  - He is reading NLP books. →
- 更复杂的方法：词法学解析器将单词解析成语素 (morphemes)
  - 词干 **Stem**：词的中心语素，提供主要的意思
  - 词缀：添加各种“附加”意义

# 词干提取 Stemming

- 更简单但更粗糙的方法：根据规则切断词缀
- 波特词干提取法 Porter Stemmer (1980)

阶段	处理目标（词缀类型）	示例
1	复数、过去式 / 分词 (-s, -ed, -ing)	cats→cat, worked→work, playing→play
2	派生后缀 (-ation, -ness, -ment)	collection→collect, happiness→happi
3	长后缀 (-icate, -ative, -alize)	activate→activ, relational→relat
4	常见后缀 (-al, -ance, -ence)	musical→music, difference→differ
5	简化词尾 (-e, -l)	argue→argu, probable→probabl

"Natural Language Processing"?

# 代码例子

```
# Porter Stemmer
from nltk.stem import PorterStemmer
ps = PorterStemmer()

print('Ex1:')
s = 'jumping jumps jumped'
print(f' Original: {s}')
print(f' Stemmed: {[ps.stem(word) for word in s.split()]}')

print('Ex2:')
s = "lying"
print(f' Original: {s}')
print(f' Stemmed: {ps.stem(s)}')
```

Ex1:

Original: jumping jumps jumped

Stemmed: ['jump', 'jump', 'jump']

Ex2:

Original: lying

Stemmed: lie

# In-class Practice I



1. B. Lie
2. ACDE

# 目前解决方案：子词模型

Subword Models

# 为什么需要子词模型？

- NLP的“词汇问题”：词级切词
  - 文本：I love unhappiness and applesauce
  - 词元：["I", "love", "unhappiness", "and", "applesauce"]
- 问题1：词库外词 (**Out-of-Vocabulary, OOV**)
  - 45个字母/肺尘病：Pneumonoultramicroscopicsilicovolcanoconiosis
  - 29个字母/(对荣华富贵等的)轻蔑：Floccinaucinihilipilification
  - 18.9万个字母：Methionylthreonylthreonylglutaminylarginyl...isoleucine
    - 一种蛋白质（肌联蛋白）的完整化学名称，由多个氨基酸残基名称拼接而成
- 问题2：数据稀疏性和低效率
  - 罕见的词
  - 多余的词汇
    - 屈折形式：run , running, ran
    - 复合词：bookstore , bookshelf

# 基本概念

- 将不太常见的单词拆分为多个子词词元

the companies are expanding  
↓  
the **compan\_ies** are **expand\_ing**

- 好处：
  - 在单词变体、复合词之间**共享参数**
  - 减小参数大小，**节省计算和内存**

# 字节对编码 Byte Pair Encoding (BPE)

(Sennrich+ 2015)

- 反复合并语料中最频繁连续字符对(字节对), 生成子词单元

```
{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}
```

```
pairs = get_stats(vocab)
```

```
[(('e', 's'), 9), (('s', 't'), 9), (('t', '</w>'), 9), (('w', 'e'), 8), (('l', 'o'), 7), ...]
```

```
vocab = merge_vocab(pairs[0], vocab)
```

```
{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}
```

```
pairs = get_stats(vocab)
```

```
[(('es', 't'), 9), (('t', '</w>'), 9), (('l', 'o'), 7), (('o', 'w'), 7), (('n', 'e'), 6)]
```

```
vocab = merge_vocab(pairs[0], vocab)
```

```
{'l o w </w>': 5, 'l o w e r </w>': 2, 'n e w e s t </w>': 6, 'w i d e s t </w>': 3}
```

Code: <https://github.com/hqyang/nlp-codes/blob/main/02-tokenization/bpe.ipynb>

# 一元语言模型 Unigram Models

(Kudo 2018)

- 使用一元语言模型独立生成序列中的所有词（详见语言模型部分）
- 给定固定词汇量的情况下，**选择词汇表**
  - 使语料库的对数极大似然(log likelihood)最大
  - 使用EM算法执行优化（细节对大多数人来说并不重要）
- **找到输入的分割**，使得单元概率最大化

# SentencePiece

- 一个高度优化的库，可以训练和使用BPE和Unigram模型

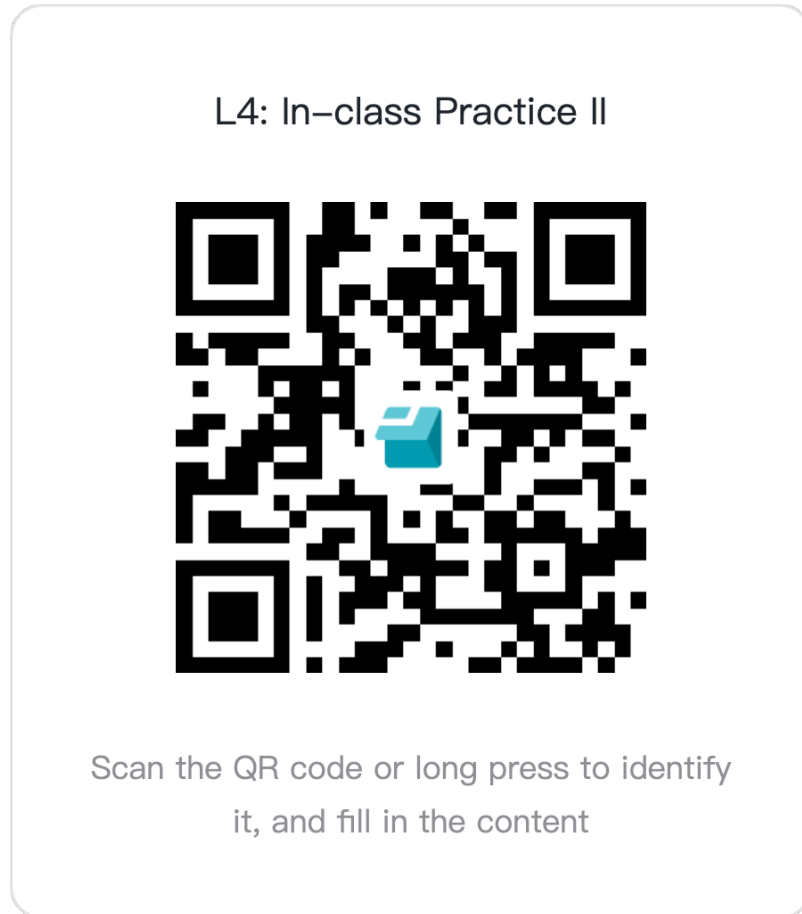
```
% spm_train --input=<input> \  
            --model_prefix=<model_name>  
            --vocab_size=8000 --character_coverage=1.0  
            --model_type=<type>  
  
% spm_encode --model=<model_file>  
            --output_format=piece < input > output
```

- Code: <https://github.com/google/sentencepiece>

# 例子：使用预训练模型的训练结果

```
• from transformers import BertTokenizer
• tokenizer = BertTokenizer.from_pretrained("./tiansz/bert-base-chinese") #
  路径与下载路径一致
•
  # 待分词的文本
• english_text = "lowest"
• chinese_text = "当我还只有六岁的时候在一本描写原始森林的名叫真实的故事的书中看到了一幅精彩的插画"
•
  # 分词（返回子词列表）
• en_tokens = tokenizer.tokenize(english_text)
• zh_tokens = tokenizer.tokenize(chinese_text)
•
  print(f"英文 '{english_text}' 分词结果：{en_tokens}")
• print(f"中文句子分词结果：{zh_tokens}")
```

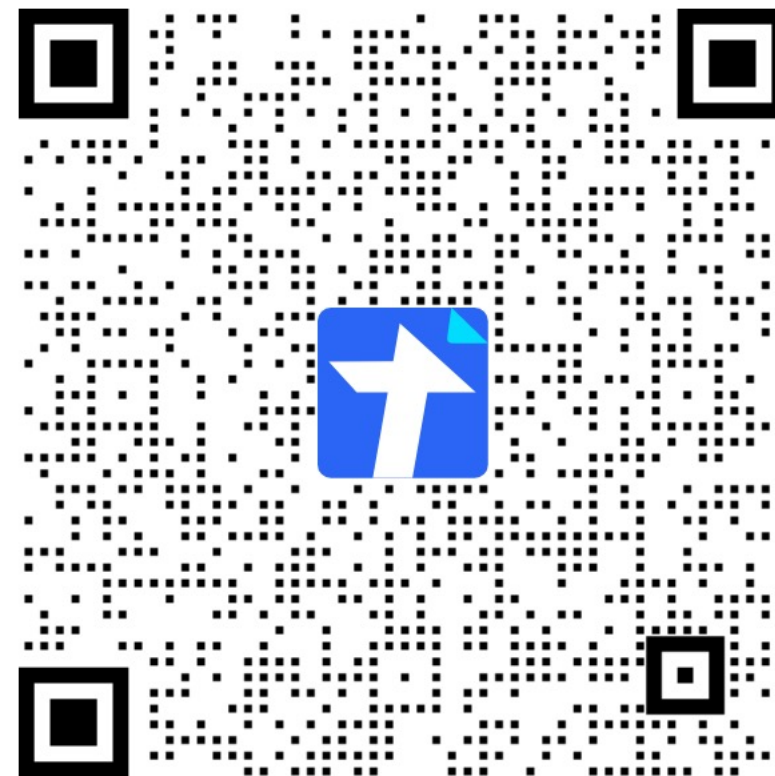
# In-class Practice II



## 1. ABD

# 一句话总结

- 切词的定義和必要性
- 切词的挑戰
- 切词的早期方案与工具
  - 文本预处理8步骤
    - 主题词表、语音索引法Soundex
    - 词根法Lemmatization
    - 词干提取Stemming
  - 工具：NLTK，正则表达式
- 切词的目前方案
  - 子词模型



1. 作业1(ddl: Oct. 19, 23:59:00)
2. 项目分组 (ddl: Oct. 19, 23:59:00)

# 附录

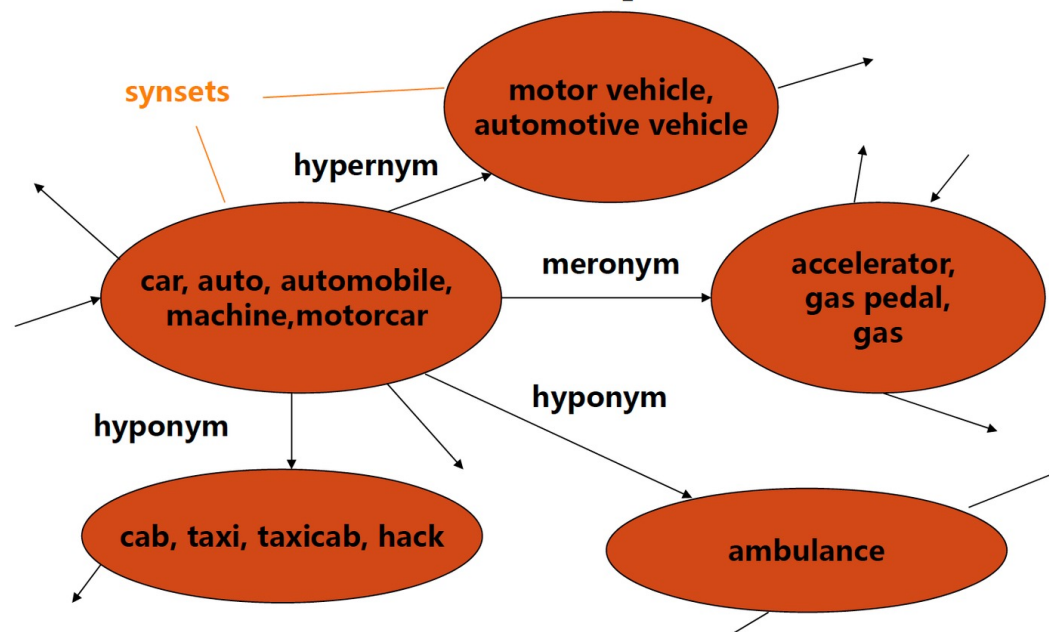
# WordNet

- 通过语义关系相互连接的机器可读的在线语义词典
- 普林斯顿大学开发
- 大型英语词汇数据库
- 语言形成一个可自由扩展的网络，以单词为节点，以概念为链接，平均最短路径相连
- 来源：<http://wordnet.princeton.edu/>

# WordNet

- 核心概念
  - 同义词集 Synonym set
  - 词汇矩阵 Lexical matrix
- 主要词汇关系
  - 同义关系 synonymy (构成Synsets)
  - 反义关系 antonymy
  - 上位关系 hypernym
  - 下位关系 hyponymy
  - 整体关系 meronymy
  - 部分关系 holonymy
  - 蕴含关系 entailment
  - 因果关系 causality
  - 近似关系 similarity

## A WordNet Snapshot



# 其他在线词典

- BabelNet
  - 多语言语义网络和词典资源
  - 罗马萨皮恩扎大学 (Sapienza University of Rome) 开发
- FrameNet (框架网络)
  - 基于框架语义学 (Frame Semantics) 的英语词汇资源库
  - 美国加州大学伯克利分校 (UC Berkeley) 开发

# Answer to In-class Practice I

## 3. “词根化 (Lemmatization)” 与 “词干提取 (Stemming)”

- 核心区别

- 词根化 (Lemmatization)：基于词法学解析，将单词还原为其“词根”（核心语义单元），处理更复杂、精准，需考虑单词的语法属性；
- 词干提取 (Stemming)：基于规则直接切断词缀（后缀为主），处理更简单、粗糙，不考虑语法属性，仅做字符串层面的截断。

- 示例

- 词根化：“am, are, is” → “be”
- 词干提取：“cats” → “cat”、“playing” → “play”

# Answer to In-class Practice I

## 4. Soundex

Johnson

1. 保留第一个字母: J
2. 去掉其他位置的所有 a, e, i, o, u, y, h, w”, 剩
  - n、s、n
3. 字母替换为数字
  - n → 5, s → 2, n → 5
4. 调整编码长度（长度4位）
  - J525

Jansons

1. 保留第一个字母: J
2. 去掉其他位置的所有 a, e, i, o, u, y, h, w”, 剩
  - n、s、n、s
3. 字母替换为数字
  - n → 5, s → 2, n → 5, s → 2
4. 调整编码长度（长度4位）
  - J525

# Answer to In-class Practice II

## 2. 字节对编码(BPE)

### 1. 统计字符对频率

- 单词c a t </w> (5 次) : 字符对为(c, a) (5 次)、(a, t) (5 次)、(t, </w>) (5 次) ;
- 单词c a t s </w> (7 次) : 字符对为(c, a) (7 次)、(a, t) (7 次)、(t, s) (7 次)、(s, </w>) (7 次) ;
- 单词d o g </w> (4 次) : 字符对为(d, o) (4 次)、(o, g) (4 次)、(g, </w>) (4 次) ;
- 单词d o g s </w> (6 次) : 字符对为(d, o) (6 次)、(o, g) (6 次)、(g, s) (6 次)、(s, </w>) (6 次) ;

# Answer to In-class Practice II

## 2. 字节对编码(BPE)

### 2. 高频字符对汇总

- (c, a) (12 次)、(a, t) (12 次)、(s, </w>) (13 次)、(d, o) (10 次)、(o, g) (10 次) 等

### 3. 确定合并对象：(s, </w>) (13 次)

### 4. 更新词汇表：

- {'c a t </w>' : 5,  
'c a t s</w>' : 7,  
'd o g </w>' : 4,  
'd o g s</w>' : 6}