基于 CRF 算法的 DailyMed 不良反应抽取、存储和可视化详解

马嘉力，周星宇，HZAU BioNLP Team

一、数据准备

1.1 数据来源

数据库：DailyMed

数据库介绍：DailyMed 是由美国国家医学图书馆（NLM）运营的数据库网站，用于向公众和医疗工作者们提供最新且准确的药品标签（也称为“药物包装说明书”）。DailyMed 数据库中提供的药物标签为 XML 格式，它提供了一种简便而通用的标记工具，包括了药物成分、药物 ID、药物名称、药物功能等相关信息，可以用于数据标注。

1.2 数据抓取

API 接口指导文档：

链接：https://dailymed.nlm.nih.gov/dailymed/webservices-help/v2/spls_api.cfm

参考(感谢史维瀚同学的技术支持)：
https://github.com/LTstrange/DailyMed_Summer_internship

API 接口：

链接：
https://dailymed.nlm.nih.gov/dailymed/services/v2/spls.xml?rxcui=312962&pagesize=5&page=2（下载 setIDs）

链接：

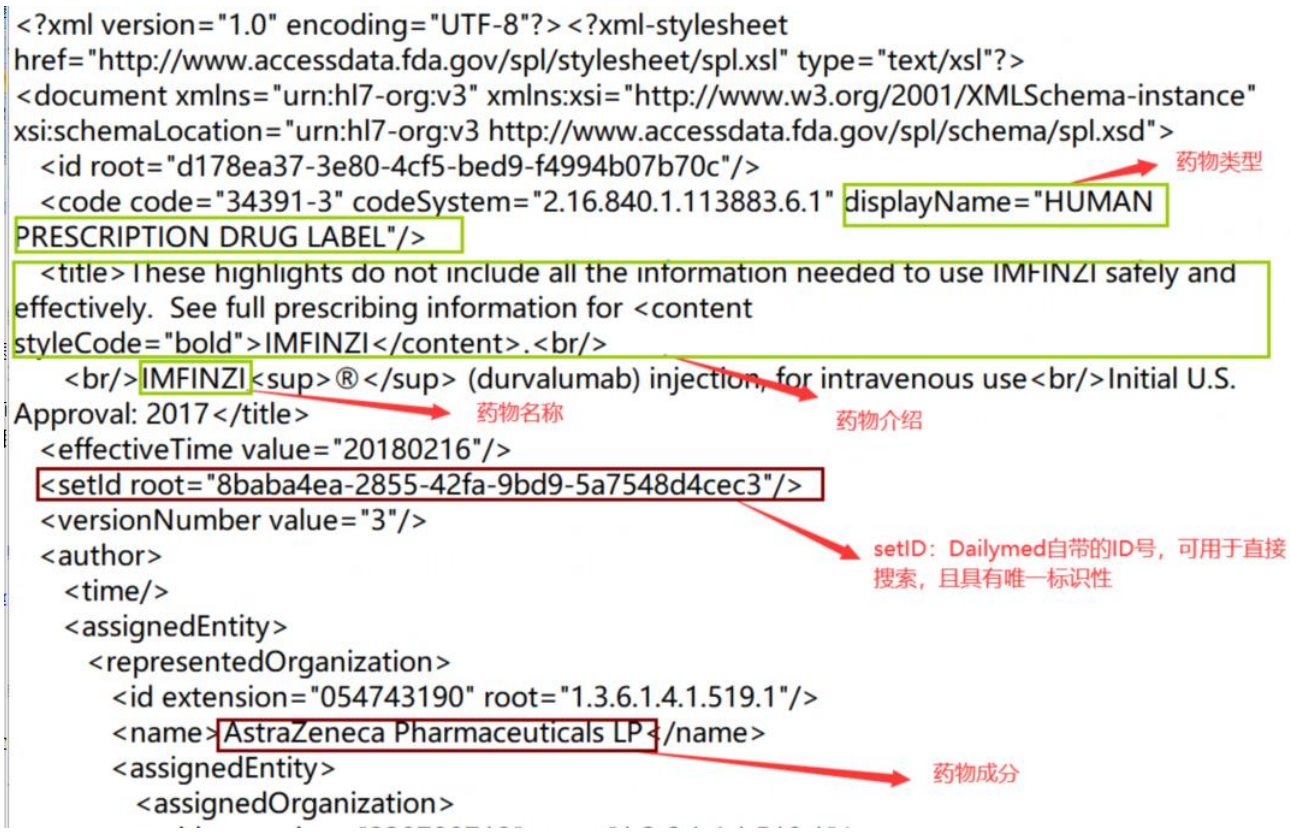
https://dailymed.nlm.nih.gov/dailymed/services/v2/spls/{setID}.xml(下载说明书.xml）

1.3 下载数据介绍

下载总量：125672 个药物说明书，总量：8G

文件格式：xml

数据内容:



二、环境准备

2.1 CRF 算法—序列标注

参考：https://github.com/bionlp-hzau/Tutorial_4_CRF [1,2]

2.1.1 配置虚拟环境

conda create -n python3.6 python=3.6

source activate python3.6

source deactivate python3.6

source activate python3.6

2.1.1 Wapiti 工具环境配置

mkdir yourProject (yourProject 是自定义的项目名)

cd yourProject

git clone https://github.com/Jekub/Wapiti.git

cd Wapiti

make

make install

./wapiti (有帮助文档输出表示安装成功)

2.1.2 CRF 工作环境配置

cd yourProject

git clone https://github.com:kyzhouhzau/2019SpringTextM.git

2.2 UMLS

2.2.1 UMLS 下载

链接：
https://www.nlm.nih.gov/research/umls/licensedcontent/umlsknowledgesources.html

注：（仅支持网页下载，不支持命令行下载）下载 UMLS 需要注册个人通行证，UMLS 通行证只颁发给个人，不颁发给公司或者组织。下载速度较慢，可能需要 6-10 小时。若在本地下载，电脑最好有 80G 以上空闲空间。以下为官方下载说明：、

①下载 UMLS zip 文件，将内容提取到单个目录中。

②下载后请阅读 README 文件，其中包括有关安装 UMLS 知识源的信息，并重点介绍了此版本的更改。

③每个 UMLS 版本都包含 MetamorphoSys，这是安装 Knowledge Sources 文件以及创建，搜索和浏览自定义的 Metathesaurus 子集所必需的。MetamorphoSys 至少需要 40 GB 的可用硬盘，并且需要 2 到 10 个小时才能在一系列经过测试的平台上运行。实际时间将取决于您的配置，硬件和操作系统平台。

2.2.2 UMLS 数据库构建[3]

1>解压

unzip umls-2020AA-full.zip

cd 2020AA-full

2>构建本地浏览工具

①Linux:

chmod 775 run_linux.sh

./run_linux.sh

②Mac:

chmod 755 run_mac.sh

./run_mac.sh

③Windows:

X32:run.bat

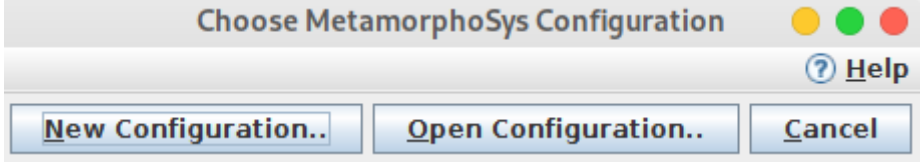
X64:run64.bat

#以 linux 为例：

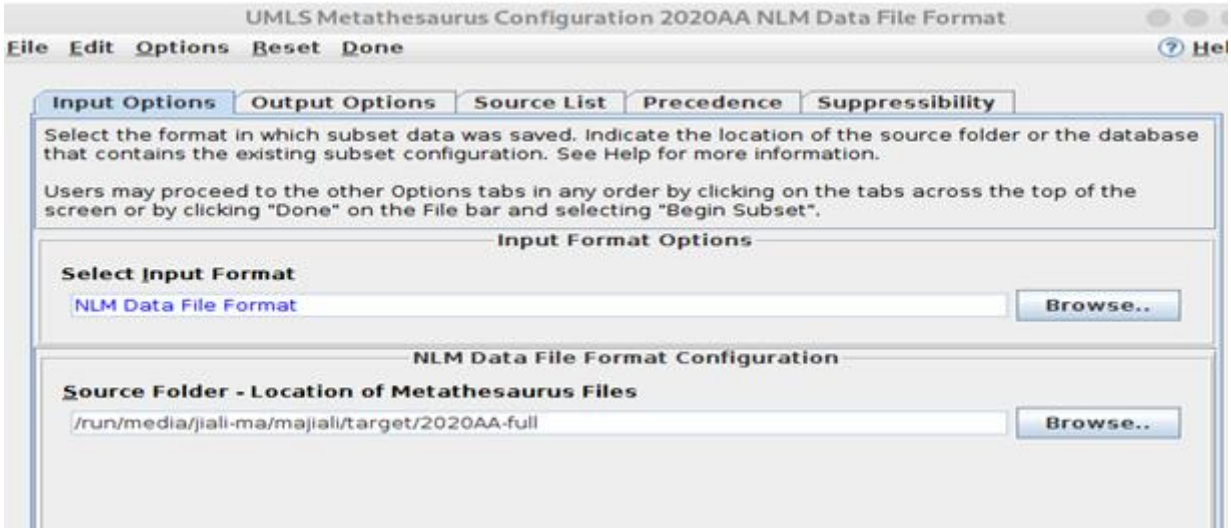
./run_linux.sh

(选择 install UMLS)

(选择安装目录，其中 choose database load scripts 选择 mysql5.6)



(这里部分安装设置可以直接按照官方说明进行设置)



（选择默认设置项，完成后选择 Done）

3>配置完成后会出现 2020AA/META 文件夹

剩余配置过程：

cd 2020AA/META

Chmod 775 populate_mysql_db.sh

修改 populate_mysql_db.sh 中的以下部分：

set MYSQL_HOME=<path to MYSQL_HOME>

set user=<username>

set password=<password>

set db_name=<db_name>

构建数据库(构建数据库过程大概需要 5-7 小时)：

./ populate_mysql_db.sh

UMLS 构建 Mysql 数据库的官方说明文档：

https://www.nlm.nih.gov/research/umls/implementation_resources/scripts/README_ORF_MySQL_Output_Stream.html

2.3 MongoDB 环境准备

介绍：MongoDB 是一个基于分布式文件存储的数据库，是当前 NoSQL 数据库产品中最热门的一种文档型数据库，功能最丰富，最像关系型数据库的产品，它支持的数据结构非常松散，BSON 格式类似于 JSON，可以存储比较复杂的数据类型，与传统关系型数据库最直观的一个区别就是在同一张表中，可以存储数据结构完全不同的数据，这是关系型数据库无法做到的，所以 MongoDB 的数据存储非常灵活。

2.3.1 Wins(cmd 操作)

以管理员权限打开 cmd

进入到 MongoDB 目录下

命令：mongod --config “~/etc/mongo.conf”

```
C:\Users\samsung\Desktop\SRF\program\mangodb\MongoDB>mongod --config "C:\Users\samsung\Desktop\SRF\program\mangodb\MongoDB\etc\mongo.conf"
```

**常见问题：

error：由于目标计算机积极拒绝，无法连接

这是由于配置文件未配置正确，需要：

在 log 文件夹下建立一个 mongo.log 的日志文件，在配置文件中输入：

##数据文件(data 文件夹目录)

dbpath=F:\project \data

##日志文件(mongo.log 所在目录)

logpath=F: \project \log \mongo.log

2.3.2 服务器上构建 MongoDB 环境

1.下载 mongodb 压缩文件

wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-ubuntu1804-4.2.8.tgz

2. 解压

tar -zxvf mongodb-linux-x86_64-ubuntu1804-4.2.8.tgz

```
(base) xyzhou@node1:~/mongo$ tar -zxvf mongodb-linux-x86_64-ubuntu1804-4.2.8.tgz
mongodb-linux-x86_64-ubuntu1804-4.2.8/THIRD-PARTY-NOTICES.gotools
mongodb-linux-x86_64-ubuntu1804-4.2.8/README
mongodb-linux-x86_64-ubuntu1804-4.2.8/THIRD-PARTY-NOTICES
mongodb-linux-x86_64-ubuntu1804-4.2.8/MPL-2
mongodb-linux-x86_64-ubuntu1804-4.2.8/LICENSE-Community.txt
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/mongodump
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/mongorestore
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/mongoexport
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/mongoimport
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/mongostat
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/mongotop
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/bsondump
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/mongofiles
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/mongoreplay
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/mongod
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/mongos
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/mongo
mongodb-linux-x86_64-ubuntu1804-4.2.8/bin/install_compass
```

3. 重命名为 mongodb，也就是我们要安装 mongo 的文件夹名称

mv mongodb-linux-x86_64-ubuntu1804-4.2.8 mongodb

4. 创建数据文件夹

cd mongodb

mkdir ./data

5. 安装 mongodb 到个人文件夹

./bin/mongod --dbpath ./data

```
(base) xyzhou@node1:~/mongodb$ ./bin/mongod --dbpath ./data
2020-07-29T10:56:08.959+0800 I  CONTROL  [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0
specify --sslDisabledProtocols 'none'
2020-07-29T10:56:08.962+0800 W  ASIO     [main] No TransportLayer configured during NetworkInterface sta
tup
2020-07-29T10:56:08.962+0800 I  CONTROL  [initandlisten] MongoDB starting : pid=26213 port=27017 dbpath=
/data 64-bit host=node1
2020-07-29T10:56:08.962+0800 I  CONTROL  [initandlisten] db version v4.2.8
2020-07-29T10:56:08.962+0800 I  CONTROL  [initandlisten] git version: 43d25964249164d76d5e04dd6cf38f6111
21f5f
2020-07-29T10:56:08.962+0800 I  CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.1.1  11 Sep 2018
2020-07-29T10:56:08.962+0800 I  CONTROL  [initandlisten] allocator: tcmalloc
2020-07-29T10:56:08.962+0800 I  CONTROL  [initandlisten] modules: none
2020-07-29T10:56:08.962+0800 I  CONTROL  [initandlisten] build environment:
2020-07-29T10:56:08.962+0800 I  CONTROL  [initandlisten]     distmod: ubuntu1804
2020-07-29T10:56:08.962+0800 I  CONTROL  [initandlisten]     distarch: x86_64
2020-07-29T10:56:08.962+0800 I  CONTROL  [initandlisten]     target_arch: x86_64
2020-07-29T10:56:08.962+0800 I  CONTROL  [initandlisten] options: { storage: { dbPath: "./data" } }
2020-07-29T10:56:08.993+0800 I  STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=6384
,cache_overflow=(file_max=0M),session_max=33000,eviction=(threads_min=4,threads_max=4),config_base=false
statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idl
_time=100000,close_scan_interval=10,close_handle_minimum=250),statistics_log=(wait=0),verbose=[recovery_
rogress,checkpoint_progress],
```

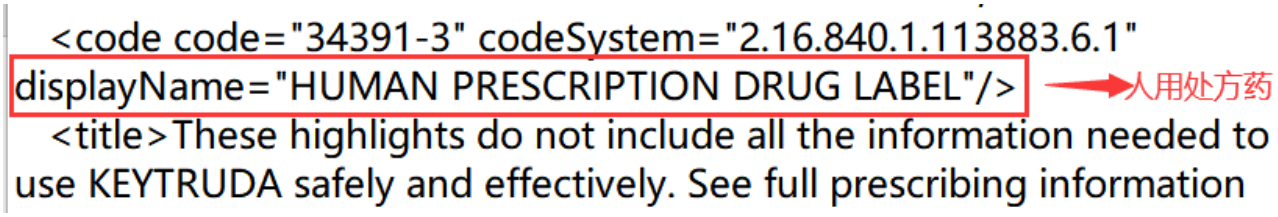
6.添加环境变量

echo “export PATH=~/mongodb/bin:$PATH” >> .bashrc

source .bashrc

三、数据(药物说明书)分类

分类依据：通过药物说明书文档(.xml)中的 displayName label 的类别进行分类，如：



分类结果：

animal-OTC	1948
animal-prescription	1043
human-OTC	79607
human-prescription	40844
other	2231

药物说明书分类统计表

注：1. 兽用非处方药 2. 兽用处方药 3. 人用非处方药 4. 人用处方药 5. 其他类

分类 shell 脚本：

```
for file in ./other/*                                    #对 other 文件夹
do
  grep &apos;Human OTC Drug Label&apos; $file      #匹配 Human OTC Drug Label&apos;
  if [ $? -ne 1 ];then
    mv $file ./human-otc/                                #将该 xml 文件分类到人类非处方中
  fi
done
```

四、副作用标注

4.1 CRF 算法

4.1.1 介绍

CRF（Conditional Random Field, 条件随机场）是一种机器学习算法,它结合了最大熵模型和隐马尔可夫模型的特点，优化了隐马尔可夫模型独立性假设太强的缺点，是一种无向图模型，目前主要用于三个方面：（1）分词：标注字的位置信息，将字组成词；（2）词性标注：词性是词汇基本的语法属性，也称为词类。词性标注是在给定句子中判定每个词的语法范畴，确定其词性并加以标注的过程；（3）命名实体识别：命名实体识别（Named EntitiesRecognition，NER）也是自然语言处理的一个基础任务，是信息抽取、信息检索、机器翻译、问答系统等 NLP 技术重要组成部分。其目的是识别语料中人名、地名、组织机构名等命名实体[2]。

4.1.2 CRF 算法使用流程

4.1.2.1 数据描述：

（1）训练数据

下载 TAC 2017 ADR(https://bionlp.nlm.nih.gov/tac2017adversereactions)训练数据有 100 个 Drug Label，数据放在 train_xml 文件夹中用于模型的训练。

（2）测试数据

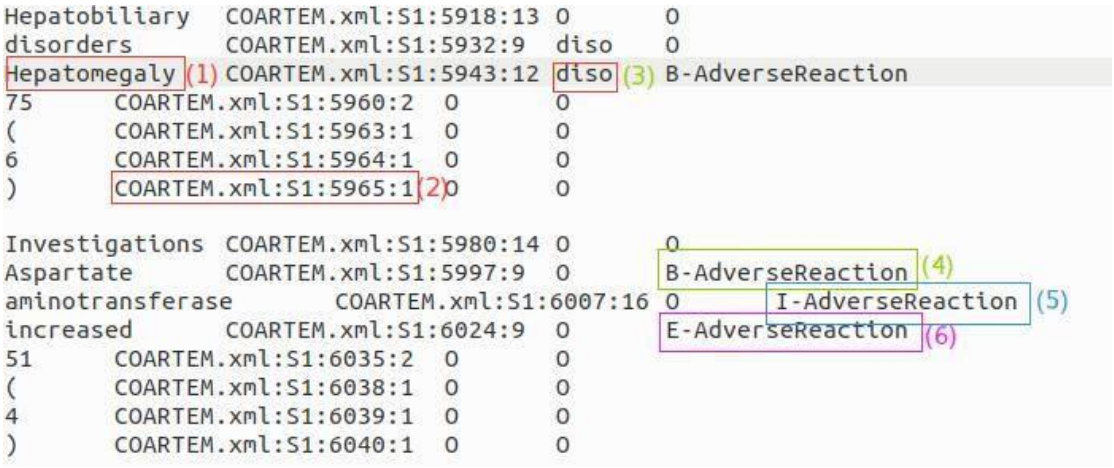
从 DailyMed （https://dailymed.nlm.nih.gov） 中下载 79607 个人类非处方药物说明书和 40844 个处方药物说明书，格式为 XML。

4.1.2.2 数据处理

(1)训练数据预处理

通过以下脚本进行训练数据的预处理

python tac2brat.py -d train_xml -o outtrain -F TokenDict:diso:diso-DISO.dic -t conll -s OBBEI



1 为 xml 文件中的内容分割而来，包括单词，数字和标点符号

2 为 1 在文档中的位置信息(文件名：所属段落：起始位置：单词长度)

3 为 1 在 DISO 字典中，表示 diso 中的副作用

4 表示一个副作用词组的开始，5 在副作用词组长度大于 3 时出现，表示副作用词组的中间位置，6 表示副作用词组的结束，O 表示不属于副作用

(2) 测试数据预处理

通过以下脚本处理测试数据。

cd 工作目录

python to_xml_needed.py ../DailyMed/human-otc ../DailyMed/human-otc-output

python tac2brat.py -d ../DailtMed/human-otc-output -o ../DailyMed/human-otc-outtest -F TokenDict:diso:diso-DISO.dic -t conll -s OBBEI

参数：-d 训练文件夹 -o 预处理结果输出文件夹 -F 字典特征 -t 输出格式 -s 标签格式

```
Catoctin        a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:12:8      O       O
Creek   a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:21:5       O       O
Distilling      a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:27:10      O       O
Company         a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:38:7       O       O
LLC     a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:46:3       O       O
Catoctin        a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:55:8      O       O
Creek   a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:64:5       O       O
Distilling      a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:70:10      O       O
Company         a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:81:7       O       O
LLC     a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:89:3       O       O
Catoctin        a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:123:8      O       O
Creek   a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:132:5      O       O
Hand    a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:138:4      O       O
Sanitizer       a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:143:9      O       O
Catoctin        a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:156:8      O       O
Creek   a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:165:5      O       O
Hand    a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:171:4      O       O
Sanitizer       a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:176:9      O       O
ALCOHOL         a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:195:7      O       O
ALCOHOL         a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:206:7      O       O
HYDROGEN        a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:225:8      O       O
PEROXIDE        a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:234:8      O       O
GLYCEROL        a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:252:8      O       O
FORMAL  a1a00068-4145-e171-e053-2a95a90aafd8.xml:S1:261:6      O       O
```

4.1.2.3 预测标签

（1） 调整参数优化模型：

sudo bash dev-wapiti.sh

（2） 预测测试数据：

sudo bash test_wapiti.sh

序列标注结果存储在 eval/bio/Tok321dis-train-test-outtrain.tab 中

4.2 UMLS 数据库使用说明

4.2.1 UMLS 介绍

统一医学语言系统（UMLS）：统一医学语言系统是一组文件和软件，集合了许多健康和生物医学词汇以及标准，以实现计算机系统之间的互操作性。

4.2.2 三种 UMLS 知识来源

Metathesaurus：来自许多词汇的术语和代码，包括 CPT，ICD-10-CM，LOINC，MeSH，RxNorm 和 SNOMED CT。层次结构，定义以及其他关系和属性。

语义网络：广泛的类别（语义类型）及其关系（语义关系）。

专家词典和词汇工具：生物医学和通用英语的大型语法词典，以及用于规范字符串，生成词汇变体和创建索引的工具。

4.2.3 引用 UMLS 中的数据

If you use UMLS in your work, please cite:

Bodenreider O. The Unified Medical Language System (UMLS): integrating biomedical terminology. Nucleic Acids Res. 2004 Jan 1;32(Database issue):D267-70. doi: 10.1093/nar/gkh061. PubMed PMID: 14681409; PubMed Central PMCID: PMC308795.

4.3 Wapiti 工具

4. 3. 1 Wapiti 简介

Wapiti 是使用不同的区分模型对序列进行快速分段和标记的工具包。它基于 maxent 模型，最大熵马尔可夫模型和线性链条件随机场(CRF)模型，提出了各种优化和正则化方法，可以提高标准模型的计算复杂性和预测性能。

Wapiti 本质上是一个程序，用于通过弹性惩罚训练和使用具有各种算法的判别性序列标记模型。Wapiti 可以训练具有一千多个标签的模型或是具有数十亿特征的模型，但训练时间会随这些特征总量的大小而增加。

本项目中，我们通过 Wapiti 工具，对数据(药物说明书)进行处理，标注出 xml 文件中对应的副作用标签(BIEO)，生成 tab 文件。

4.3.2 Wapiti 用法

wapiti mode [options] [input] [output]

Wapiti 具有两种模式，分别是 train 模式和 label 模式：

在 train 模式下，Wapiti 将加载先前给出的模型，读取 train 数据集和最终开发的模型，然后训练我们自己的模型。在所有这些步骤中，将输出进度信息。当模型达到停止标准之一或用户发送 TERM 信号时，停止训练并保存模型。训练结束后会对当前模型进行打分，通过调整参数进行模型优化。

在 label 模式下，必须提供已标记的数据。这里我们提供的标记数据是周开银师兄提供的 100 篇标记好的 xml 文件，并通过该文件训练好模型后，进行我们自己的药物说明书的副作用标注。

4.4 Wapiti 调参（https://wapiti.limsi.fr/manual.html）

调参是 Wapiti 使用过程中的一项重要内容，通过调整不同的单词组合方式和训练次数，可以得到更好的模型和更高的预测准确率。

Wapiti 的单词组合模式有两种，分别为 unigram(u)和 bigrams(b)，最常用的模式是 unigram 模式，调参过程通过修改.pat 文件实现，以下为示例：

U:tok:2:-1:%X[-1,0]/%X[0,0]

U:tok:2:+0:%X[0,0]/%X[1,0]

第一个字符必须是'u'、'b'或'*'(大写或小写)。

例如，如果数据是

a1 b1 c1

a2 b2 c2

a3 b3 c3

且当前位置为 a2，则将产生观测结果“u:a1/a2”和“u:a2/a3”。

通过调整参数组合和迭代次数，我们得到以下两组统计结果：

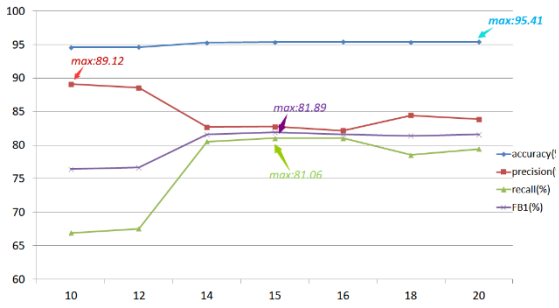


图 1　1-3 个单词组合

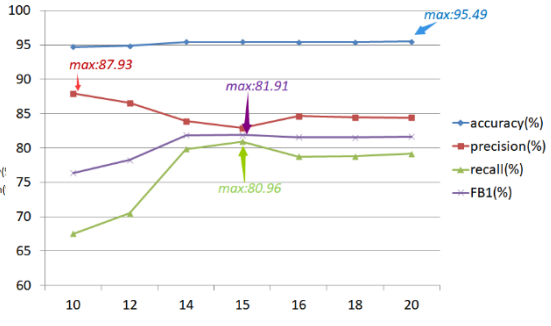


图 2　1-4 个单词组合

其中，X 轴为迭代次数，y 轴为模型训练的结果。通过表格可以看出，不论是 1-3 个单词的组合还是 1-4 个单词的组合，模型的 BF1 最大值都出现在第 15 次迭代上，且 1-4 的单词组合 BF1 值稍高于 1-3 的组合，且找到的副作用总数更多，所以这里我们采用 1-4 的组合迭代 15 次来训练我们的模型。

	Precision	Recall	FB1	Found
			81.91	3109
AdverseReaction	82.89%	80.96%		
Animal	76.47%	59.09%	66.67	17
DrugClass	40.00%	8.70%	14.29	5
Factor	74.34%	65.12%	69.42	113
Negation	43.75%	50.00%	46.67	16
Severity	68.83%	57.05%		
			62.39	247

图 3 模型优化结果统计

4.5 标注结果展示

	有副作用	无副作用	总计	有副作用比例
OTC	58548	21059	79607	73.55%
PRESCRIPTION	38870	1974	40844	95.17%

图 4 Wapiti 标注结果统计表

如图 4，我们可以发现，相比于非处方药，处方药药物说明文档中包含副作用相关说明的比例要大一些。

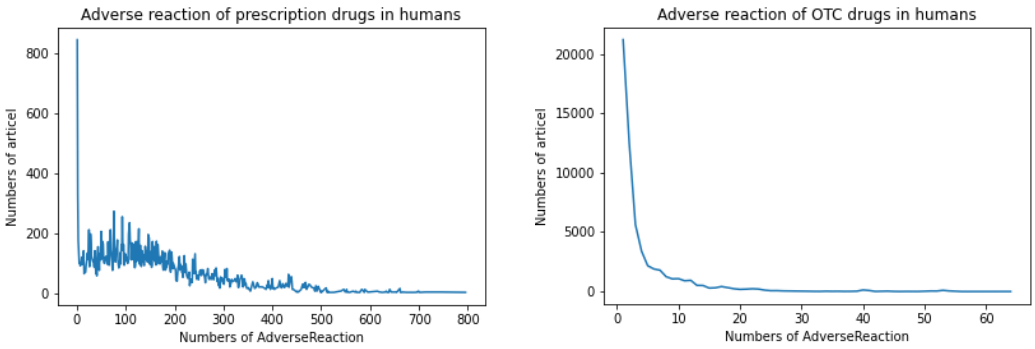


图 5 副作用分布

如图 5，我们通过 Wapiti 工具标注了所有人类相关的药物说明书，通过标注结果，我们发现，处方药物的副作用平均个数明显多于非处方药，并且处方药副作用多数分布于 0-300 之间，但是非处方药很少有大于 10 个副作用的，这可能是为什么在生活中我们可以到药店自己买到非处方药，但是处方药却需要医生的药方的原因之一。

五、标注结果存储以及可视化

5.1 在 MongoDB 中导入标注结果

5.1.1Wins 本地(cmd 下)

①导入文件 (.json)：

本地首先进入 Mongodb 的 bin 目录下：

mongoimport -d dbname -c collectionname 导入的 json 文件路径

```
C:\Users\samsung\Desktop\SRF\program\mangodb\MongoDB\bin>mongoimport -d test1 -c Prescription_human E:\target\jason_file\Prescription.json
2020-07-25T18:34:16.420+0800    connected to: localhost
2020-07-25T18:34:19.409+0800    [################........] test1.Prescription_human     78.7MB/108MB (72.7%)
2020-07-25T18:34:20.477+0800    [########################] test1.Prescription_human     108MB/108MB (100.0%)
2020-07-25T18:34:20.477+0800    imported 38871 documents
```

②导出文件 (.json)

本地首先进入 Mongodb 的 bin 目录下：

mongoexport -d dbname -c collectionname 导出的 json 文件路径

5.1.2 远程 Mongodb 的导入与导出

①导入文件 (.json)：

mongoimport -h IP -d dbname -c collectionname 路径

②导出文件 (.json)

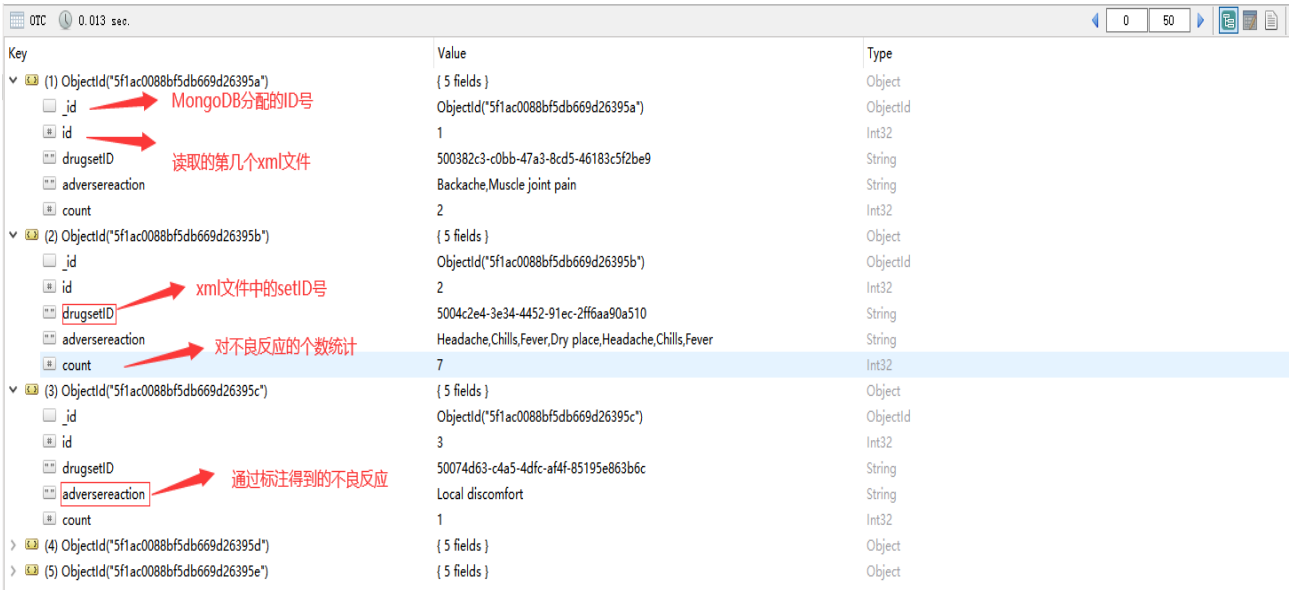
mongoexport -h 192.168.20.55:27017 -d dbname -c collectionname -o 路径

5.2 可视化

（Studio 3T / Robo 3T）可视化展示

① OTC_Human(人类非处方药)

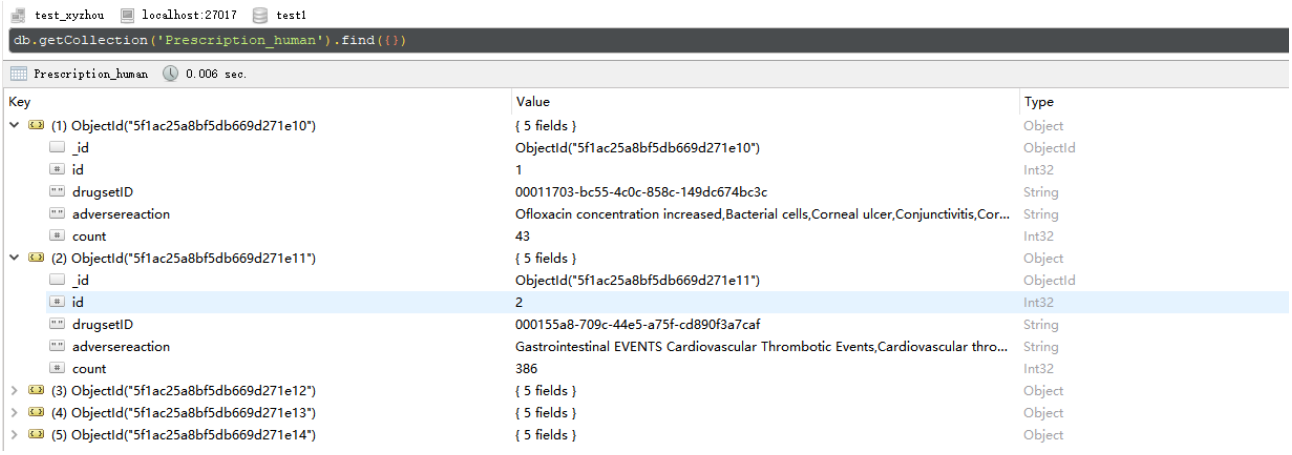
1.



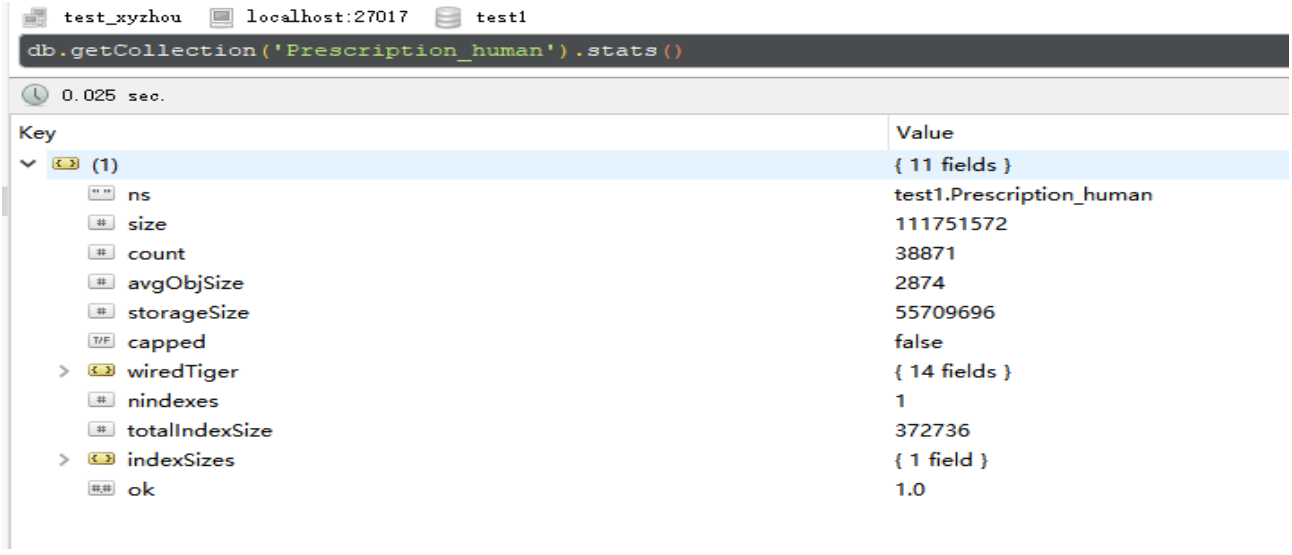
2.

②Prescription_human（人类处方药）

1.



2.



从可视化结果中，我们得知：对于 DailyMed 数据库中的 125672 个药物说明书，利用 CRF 算法进行表型(不良反应)标注，结果为：人类非处方药中具有不良反应的药物一共有 58549 个；人类处方药中具有不良反应的药物一共有 38871 个。

六、讨论

CRF 是判别模型，对问题的条件概率分布建模，而 HMM 是生成模型，对联合概率分布建模。相比于 HMM(Hidden Markov Model，隐马尔科夫模型)，所有能用 HMM 解决的问题，基本上也都能用 CRF 解决，并且 CRF 还能利用更多 HMM 没有的特征，这也体现了 CRF 算法优化了隐马尔可夫模型独立性假设太强的缺点。

CRF 可以用前一时刻和当前时刻的标签构成的特征函数，加上对应的权重来表示 HMM 中的转移概率，可以用当前时刻的标签和当前时刻对应的词构成的特征函数，加上权重来表示 HMM 中的发射概率。

另外，CRF 相比 HMM 能够利用更加丰富的标签分布信息，因为：①HMM 只能使用局部特征，转移概率只依赖前一时刻和当前时刻，发射概率只依赖当前时刻，CRF 能使用更加全局的特征，例如词性标注问题中，如果句子末尾出现问号“？”，则句首第一个词是动词的概率更高。②HMM 中的概率具有一定的限制条件，如 0 到 1 之间、概率和为 1 等，而 CRF 中特征函数对应的权重大小没有限制，可以为任意值。

本项目中，我们将 Dailymed 中下载的药物说明文档(.xml)根据文档中的信息，分为了 5 类（人类用药两类：处方与非处方，动物用药两类：处方与非处方，以及其他类）。基于 CRF 算法，我们下载了 100 个 Drug Label 进行了数据训练。通过调整参数，优化模型，提高标注结果的准确性与真实性。

最终，我们将分类好的药物说明文档(.xml)进行了 BIEO 标签的标注，标注出了各药物的表型(即：不良反应)。通过 MongoDB 进行了结果的存储，并借用了可视化工具(Studio 3T / Robo 3T)进行展示。

后续我们会继续对该项目进行深入研究，去探究标注结果中隐藏着的规律，并联系疾病实体(如：AD,阿尔兹海默症)进行药物说明文档进一步的分析。同时加深对数据挖掘相关知识的学习，提高我们的算法学习能力。

七、致谢

感谢夏静波老师的项目提供以及全程的悉心指导。感谢王宇星学姐、周开银学长、姚昕智学长对项目的建议。感谢史维瀚、马欣晨同学提供的代码支持。感谢夏老师项目组所有同学与老师一直以来的项目交流与陪伴！

由于疫情的原因，很遗憾没有能够跟同学与老师们一同于狮山校园度过自己的大三时光，但也很庆幸，能够在大学期间加入到夏老师项目组，认识到一群可爱又喜爱科研的同学们，能够与你们一同学习，让我们受益匪浅。在项目完成过程中，我们学习到了很多数据处理的方法、MongoDB 数据库的使用以及 CRF 等算法，让我们对课题有了更进一步的了解，锻炼了我们的科研能力。最后，再次感谢夏老师在这半年期间为我们提供的帮助与鼓励。望早日相逢狮山！

八、参考文献

[1] Kaiyin Zhou, Sheng Zhang, Xiangyu Meng, Qi Luo, Yuxing Wang, Ke Ding, Yukun Feng, Mo Chen, Kevin B Cohen, Jingbo Xia*. CRF-LSTM Text Mining Method Unveiling the Pharmacological Mechanism of Off-target Side Effect of Anti-Multiple Myeloma Drugs. BioNLP 2018, workshop in ACL 2018. Melbourne, Australia, pp.166-171.

[2] Zheng Y, Meng X, Zweigenbaum P, et al. Hybrid phenotype mining method for investigating off-target protein and underlying side effects of anti-tumor immunotherapy[J]. BMC Medical Informatics and Decision Making, 2020, 20(3): 1-11.

[3] .Bodenreider O. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic acids Res*, 2004, 32: D267-D270.

附录：

MongoDB.py

```
#!/usr/bin/env python
# coding: utf-8

import pymongo
class DATABASES():
    def __init__(self):
        self.mongo_client = pymongo.MongoClient('localhost',27017)
        self.db = self.mongo_client['test']
        self.table = self.db['DRUG']
    def insert(self,data):
        #if self.table.find_one({'id' : data['id']}):
            #return
        #else:
        self.table.insert_one(data)
        #return
    def data(self,file_input):
        adversereaction = ''
        article = ''
        lastarticle = 'start'
        count = 0
        ids = 0
        flag = False
        flag2 = False
        with open(file_input,'r') as f:
            for line in f:
                if line[-2:]=="O\n" or line=="\n":
                    flag = False
                    continue
                li = line.strip().split()
                if li[-1] == 'B-AdverseReaction':
```

```
        article = li[1].strip().split('.')[0]
        if article != lastarticle and ids != 0 :#and count != 0:
            adversereaction = adversereaction[:-1]
            data = {
                'id' : ids,
                'drugsetID' : lastarticle,
                'adversereaction' : adversereaction,
                'count' : count
            }
            #print(data)
            ids += 1
            count = 1
            self.insert(data)
            adversereaction = li[0].capitalize()+','
            data = {}
            lastarticle = article
            flag = True
        elif article != lastarticle and ids ==0 :
            lastarticle = article
            count += 1
            adversereaction = li[0].capitalize()+','
            ids =1
            flag = True
        elif article == lastarticle :
            count += 1
            adversereaction = adversereaction + li[0].capitalize()+','
            flag = True
    elif li[-1] == 'I-AdverseReaction' and flag:
        if li[0] == '-':
            adversereaction = adversereaction[:-1] + li[0]+','
            flag2 = True
        else :
            if flag2:
```

```
                    adversereaction = adversereaction[:-1] + li[0]+','
                else:
                    adversereaction = adversereaction[:-1] + ' ' + li[0]+','
            elif li[-1] == 'E-AdverseReaction' and flag:
                adversereaction = adversereaction[:-1] + ' ' + li[0] +','
                #print(adversereaction)
                flag = False
                flag2 = False


        data = {
                    'id' : ids,
                    'drugsetID' : article,
                    'adversereaction' : adversereaction,
                    'count' : count
                }
        self.insert(data)
        f.close()
    return

if __name__=='__main__':
    inf = './target/result/Tok321dis-train-test-outtrain.tab'
    Drug = DATABASES()
    Drug.data(inf)
```