基于知识图谱的联通 QA 系统

--培训提纲

1. 准备工作

工程环境

安装 python 环境

这里推荐安装 anaconda 集成环境。

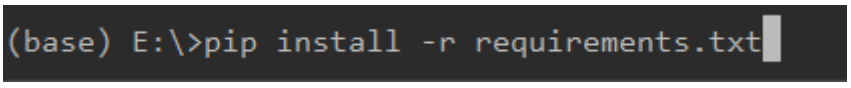
链接：https://www.anaconda.com/products/individual

根据需要进行安装，这里使用 python 3.7 版本。

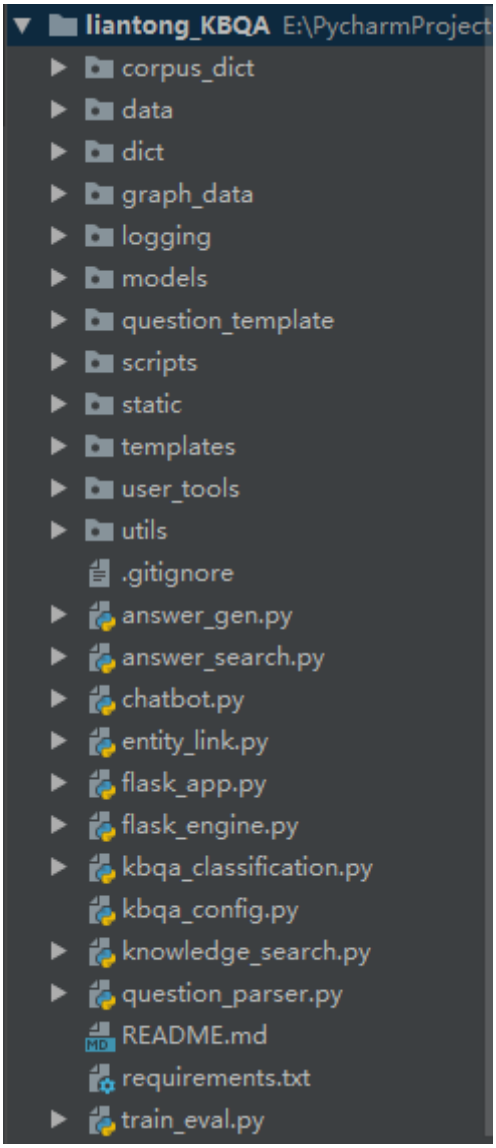
安装工程需要 python 库

进入工程根目录下，运行以下命令，进行安装：

pip install -r requirements.txt



工程目录结构



目录解析

目录/文件	说明
corpus_dict	深度模型构建语料的实体数据
data	包含训练数据、预训练模型以及训练好的模型。 ● model_data：训练好的模型 ● pretrain_models：预训练模型
dict	知识图谱中的实体数据
graph_data	导入 neo4j 数据库的实体数据和关系数据

logging	日志文件夹 ● models：模型训练产生的日志 ● web：启用 flask 产生的日志
models	模型文件夹
question_template	问题模版文件夹
scripts	shell 脚本
static	包含 js，图片等资源文件
templates	html 页面模版
user_tools	用户用到的 python 脚本文件
utils	工程用的 utils
answer_gen.py	问题答案生成模块
answer_search.py	操作 Neo4j 数据模块
chatbot.py	通过终端运行 KBQA 的脚本
entity_link.py	实体链接模块
flask_app.py	flask 服务启动脚本，将整个 QA 系统部署到 Web 端
flask_engine.py	为 flask_app 提供模型接口和数据接口
kbqa_classification.py	模型整体训练和测试脚本
kbqa_config.py	项目的配置文件
knowledge_search.py	知识搜索模块（含 AC 自动机和相似度计算）
question_parser.py	问题解析模块
README.md	项目的说明文档，markdown 格式
requirement.txt	项目所依赖的 python 库
train_eval.py	模型训练和评估工具

2. 数据构建

更详细的请参考文档:
《联通知识图谱创建用户手册 v1.0.docx》

知识图谱构建

目前仅可以通过人工的方式，去构建所需要的知识图谱。具体的过程如下:

1、 在 Excel 中相应的实体 sheet 中填写实体-属性相关信息。
2、 在 Excel 中“relation”sheet 中填写实体-关系三元组信息。
3、 通过提供的脚本（extract_graph_data.py），提取实体-属性信息，以及实体-关系三元组信息。

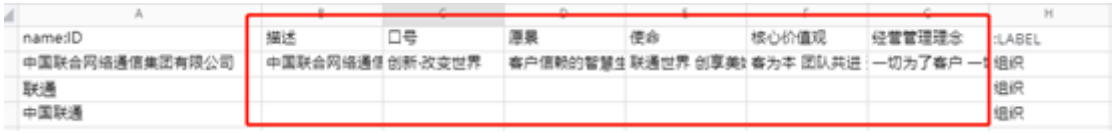
4、 通过提供的脚本（import_neo4j.sh），导入到 Neo4j 数据库中。

新建文档

新建《知识图谱数据表.xlsx》文档，并新建“relation”sheet，以及 19 种以实体命名的 sheet。

填写实体-属性信息

这里以“组织”实体为例：



name:ID	描述	口号	愿景	使命	核心价值观	经营管理理念	:LABEL
中国联合网络通信集团有限公司	中国联合网络通信	创新·改变世界	客户信赖的智慧生	联通世界 创享美	客为本 团队共进	一切为了客户 一	组织
联通							组织
中国联通							组织

说明：

1、首列必为 name:ID，其必须唯一。

2、除去首位列，中间红色方框的为实体的属性信息。若无，可不填。

3、末列必为:LABEL，其填写的是当前实体的实体类型。

填写实体-关系三元组信息

以下以组织实体-关系的三元组信息为例：

:START_ID	:END_ID	:TYPE
中国联合网络通信集团有限公司	北京市西城区金融大街21号	地点
中国联合网络通信集团有限公司	北京	成立地点
中国联合网络通信集团有限公司	2009年1月6日	成立时间
中国联合网络通信集团有限公司	新势力	品牌
中国联合网络通信集团有限公司	WO	品牌
中国联合网络通信集团有限公司	王晓初	董事长
中国联合网络通信集团有限公司	李国华	总经理
中国联合网络通信集团有限公司	李国华	董事
中国联合网络通信集团有限公司	王晓初	党组书记
中国联合网络通信集团有限公司	李国华	党组副书记
中国联合网络通信集团有限公司	邵广禄	副总经理
中国联合网络通信集团有限公司	奚彦州	副总经理
中国联合网络通信集团有限公司	朱可炳	党组成员
中国联合网络通信集团有限公司	范云军	党组成员
中国联合网络通信集团有限公司	中国联合网络通信集团有限公司	中文名
中国联合网络通信集团有限公司	China Unicom	英文名
中国联合网络通信集团有限公司	联通	别名
中国联合网络通信集团有限公司	原中国联通	前身
中国联合网络通信集团有限公司	原中国网通	前身
联通	中国联合网络通信集团有限公司	全称
中国联通	中国联合网络通信集团有限公司	全称
China Unicom	中国联合网络通信集团有限公司	全称

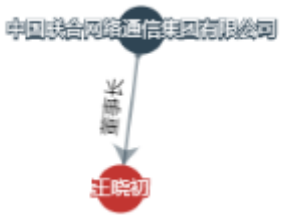
说明：

1、首列必为:START_ID，其为 Subject。

2、次列必为:END_ID，其为 Object。

3、末列必为:TYPE，其为 Subject 与 Object 间的关系，箭头从 Subject 指向 Object。

4、首列和次列的数据，即 Subject 和 Object 必须按照其相应的实体类型，在相应的实体类型 sheet 中存在。



抽取知识图谱信息

该步骤将抽取上述步骤在《知识图谱数据表.xlsx》文档中填写信息，存储为 csv 格式的文档。

- 配置

/user_tools/tools_config.py

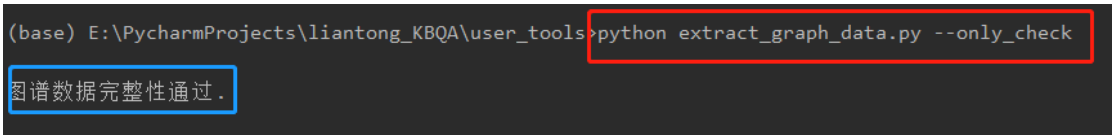
```
# ========graph data Extract setting=============
"""
根据提供的图谱数据xlsx文件，进行图谱数据生成
"""
# 图谱数据所在文件夹， 以及图谱数据输出文件夹
graph_data_dir = 'graph_data'

# 图谱数据xlsx文件
graph_data_file = '知识图谱数据表.xlsx'

# entity output
entity_output_dir = 'dict'
```

- **graph_data_dir**：为抽取知识图谱数据存放的文件夹，用于后期导入Neo4j.。
- **graph_data_file**：与抽取后的知识图谱数据同级目录，为输入文件。
- **entity_output_dir**：知识图谱的实体数据存在位置，会根据同级目录下的 **tagging.csv** 文件进行实体抽取，用于 QA 过程中快速进行实体计算等。

- 检测数据完整性

在终端执行以下命令：

python extract_graph_data.py --only_check



如果输出蓝色框信息，表明数据完整性通过。否则，会提示缺少的数据，需要补充完整直到数据完善为止。

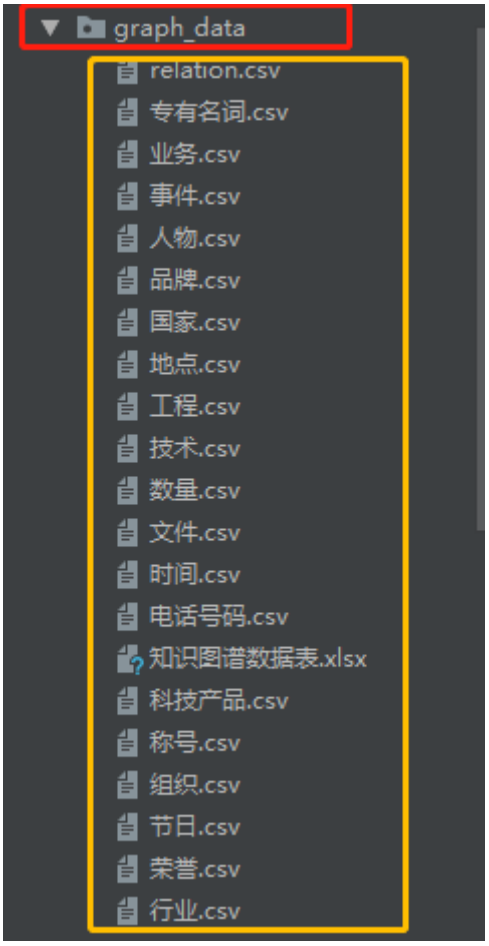
- 抽取数据

在终端，不需要带任何参数就可以完成整个抽取过程。

python extract_graph_data.py

```
(base) E:\PycharmProjects\liantong_KBQA\user_tools>python extract_graph_data.py

图谱数据完整性通过.

graph data relation save successful, itmes: 281
graph data 组织 save successful, itmes: 20
graph data 人物 save successful, itmes: 37
graph data 科技产品 save successful, itmes: 22
graph data 技术 save successful, itmes: 22
graph data 事件 save successful, itmes: 1
graph data 工程 save successful, itmes: 3
graph data 称号 save successful, itmes: 2
graph data 荣誉 save successful, itmes: 4
graph data 国家 save successful, itmes: 6
graph data 地点 save successful, itmes: 11
graph data 时间 save successful, itmes: 38
graph data 电话号码 save successful, itmes: 2
graph data 品牌 save successful, itmes: 6
```

- 抽取结果
  - **graph_data 文件夹**



  - **dict 文件夹**

导入 Neo4j 数据库

执行提供的 shell 脚本，如下图所示：

bash import_neo4j.sh liantong_qa /data/QA/liantong_kbqa/graph_data

- 第一个参数（黄色方框）：数据库名称。要求：Neo4 数据库中未使用过，长度大于 3。
- 第二个参数（蓝色方框）：导入数据的路径。

（后期有新的实体或需要更改脚本，请自行修改。）

训练语料构建

更详细的请参考文档：
《联通 QA 系统—基于模版的语料构建说明 v1.0.docx》

新建文档

新建《问句模版表.xlsx》文档，根据需求，为不同的实体构建模版语料。

- 构建语料的实体为已定义的模型需要识别的 15 种实体，即 **tagging.csv** 中定义的。
- 目前仅构建单实体，双实体类型的语料。

单实体模版构建

支持问题类型：

- 事实类
- 描述类
- 枚举类-基于时间查询实体
- 枚举类-查询关系数量

➢ 模版格式

表 9 模版格式

实体类型	总意图	细意图	文本模版	问句分类
实体	意图	意图	文本	问句分类

注：当前总意图暂时未用。现在所用的是细意图。

➢ 模版实例

实体类型	总意图	细意图	句子	问题分类
组织	时间	成立时间	XXX 的成立时间？	事实类

双实体模版构建

支持问题类型：

- 关系类
- 枚举类-基于时间查询实体
- 枚举类-查询实体数量

➢ 模版格式

表 12 双实体模版格式

实体类型	总意图	细意图	文本模版	问句分类
实体-实体	意图	意图	文本	问句分类

注：

1、 当前总意图暂时未用。现在所用的是细意图。

2、 实体类型中间用“-”隔开。

➢ 模版实例

实体类型	总意图	细意图	句子	问题分类
组织-组织	查询关系	查询关系	XXX 和 YYY 的关系？	关系类

模版抽取

- 配置

/user_tools/tools_config.py

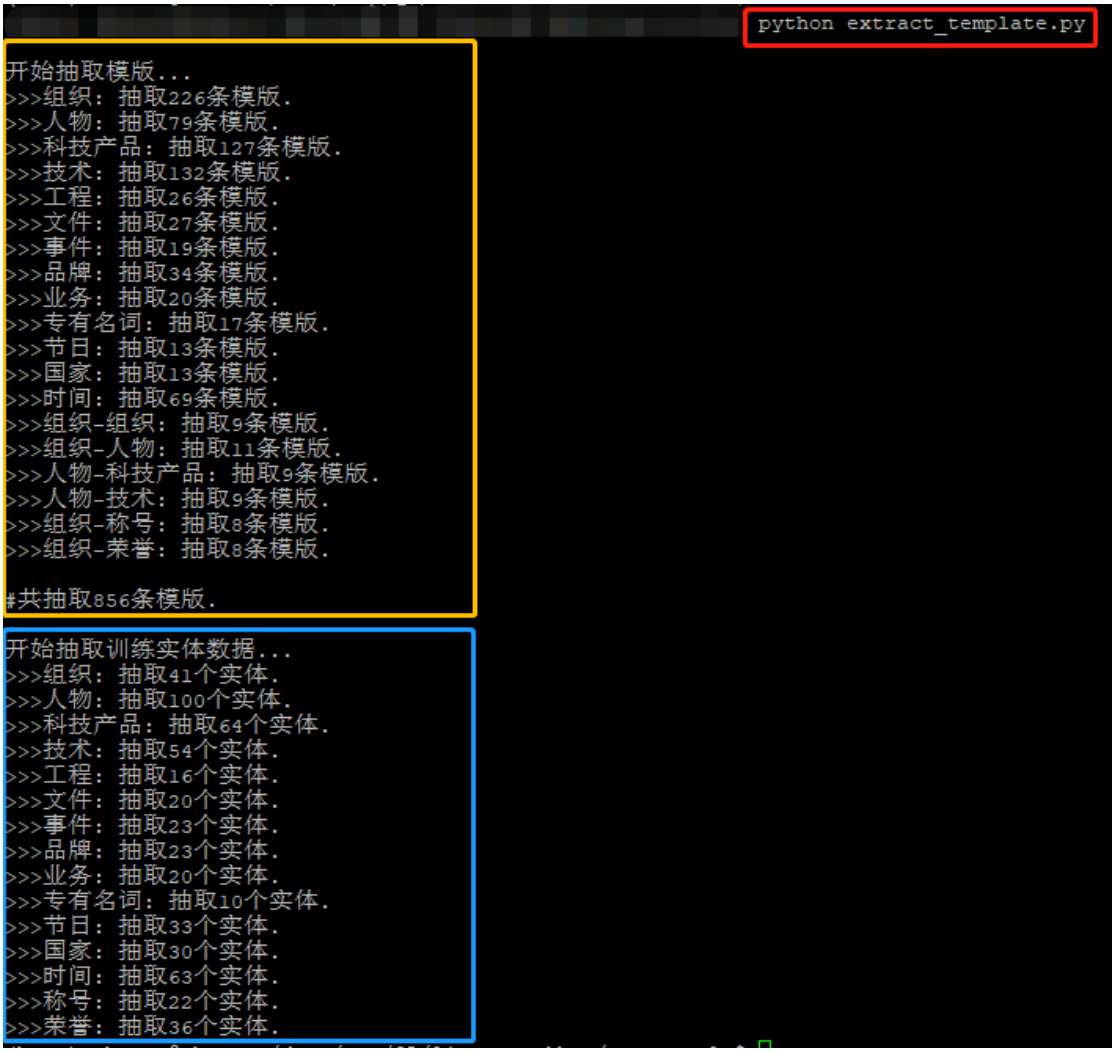
```
# ==========Template Extract setting===============
"""
根据提供的模版数据xlsx文件，进行训练数据的模版生成
"""
# 模版所在文件夹，以及模版输出文件夹
template_dir = 'question_template'

# 模版xlsx文件
template_file = '问句模板表.xlsx'

# 训练实体语料输出文件夹
entity_corpus_output = 'corpus_dict'

# 抽取的模版sheet表
template_list = ['专有名词', '业务', '事件', '人物', '品牌',
                 '工程', '技术', '文件', '科技产品', '组织', '节日',
                 '时间', '国家',
                 '组织-组织', '组织-人物', '人物-科技产品', '人物-技术',
                 '组织-称号', '组织-荣誉',
                 ]

# 模版起始所在列
template_start_col = 2
```

  - **template_dir**：输出抽取后的模版文件。
  - **template_file**：输入的模版文件名。
  - **entity_coupus_output**：模版需要的实体数据存放位置。
  - **template_list**：需要抽取的 sheet。（15 种实体的组合）
  - **template_start_col**：模版在 sheet 中开始的列数。（0 列是实体）

- 抽取模版

执行抽取模版的 python 脚本，如下图所示：

```
python extract_template.py
```



生成训练数据

- 配置
  /user_tools/tools_config.py

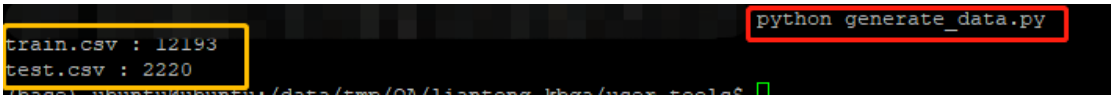
```
# ========generate data setting=============
"""
数据生成需要提供模版，以及相应的实体，
同时，模版如果是单实体，需要与实体文件同名；
如果是两个实体，中间需要'-'隔开。

"""
# 模版文件夹
template_input = template_dir

# 实体语料文件夹
entity_corpus_input = entity_corpus_output

# 生成的训练数据所在文件夹
train_data_dir = 'data'

# 数据集划分成的文件
split_file = ['train.csv', 'test.csv']

# 数据划分比例
split_ratio = [0.85, 0.15]
```

- **template_input**：抽取后的模版所在文件夹。
- **entity_corpus_input**：生成数据需要的实体数据。
- **train_data_dir**：生成的训练数据存放的位置。
- **split_file**：需要生成的数据集文件名。（这里将数据集划分成训练集和测试集）
- **split_ratio**：模数据集划分比例（与 **split_file** 对应）。

- 数据生成

执行抽取模版的 python 脚本，如下图所示：

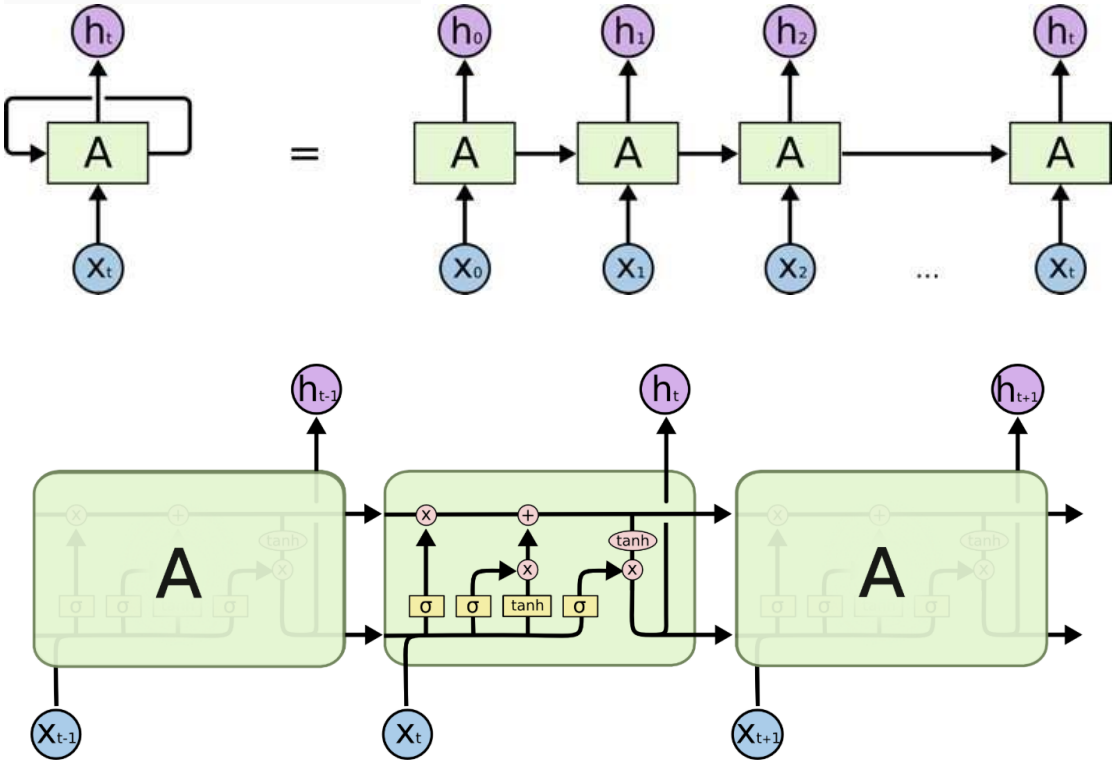
python generate_data.py



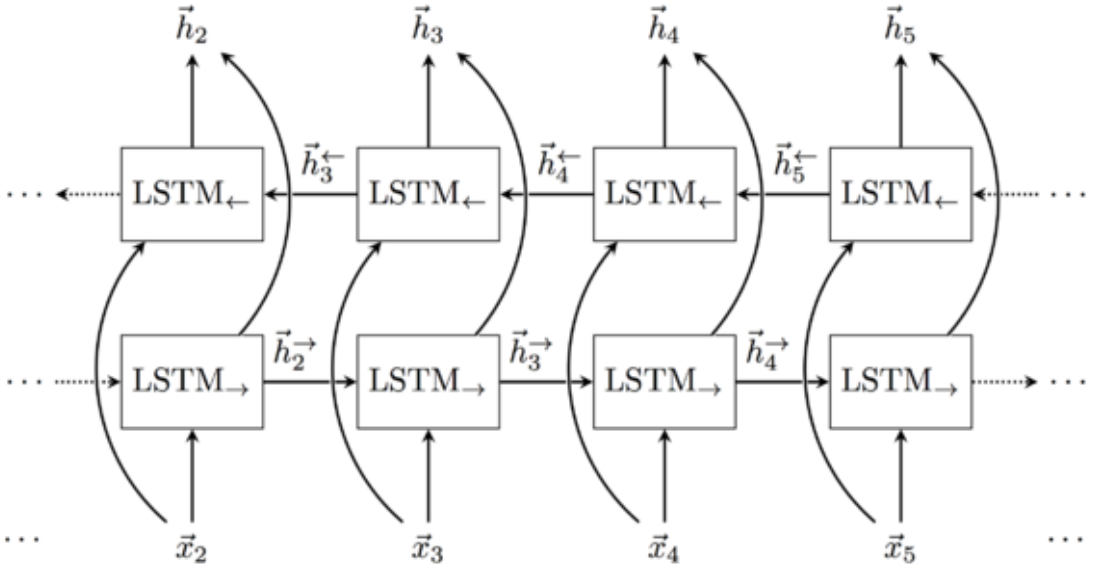
3. 模型部分

LSTM 介绍

LSTM 是一种循环神经网络模型，常用于处理序列数据，如一段文字或声音、购物或观影的顺序，甚至是图像中的一行或一列像素。因此，循环神经网络有着极为广泛的实际应用，如语言模型、文本分类、机器翻译、语音识别、图像分析、手写识别和推荐系统。



The repeating module in an LSTM contains four interacting layers.

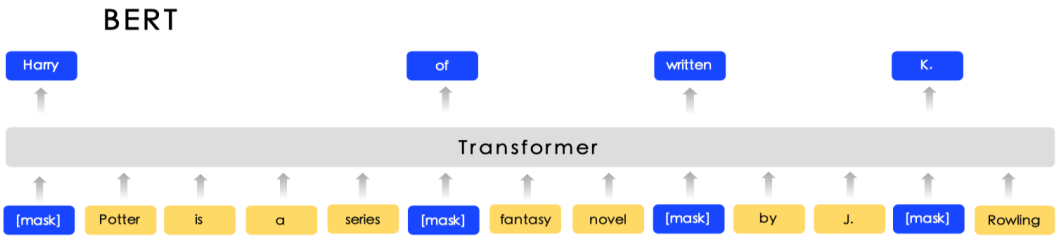


LSTM 模块内部计算可参考资料：
https://zh.d2l.ai/chapter_recurrent-neural-networks/lstm.html
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

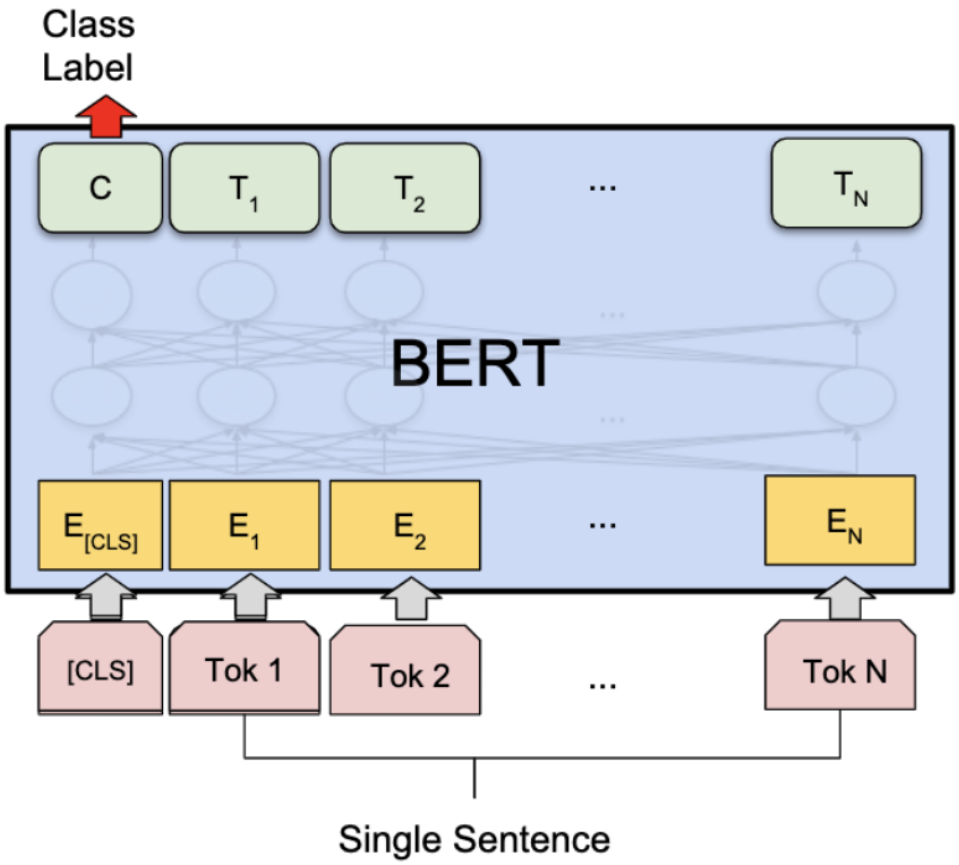
Bert 介绍

Auto-regressive LM：自回归语言模型
在 BERT 问世之前，我们常说的语言模型都属于自回归语言模型，类似于 RNN 这样的结构，我们用上文信息来预测下一个单词，或者根据下文信息来预测上一个单词，这种类型的结构就称为自回归语言模型。**只能利用上文或者下文的信息(只能考虑单向)，不能同时利用上文和下文的信息(不能双向同时考虑)。**
Auto-encoder LM：自编码语言模型
自回归语言模型只能根据上文预测下一个单词，或者反过来，只能根据下文预测前面一个单词。相比而言，Bert 通过对输入的句子进行随机 Mask，然后预测。**双向语言模型，同时看到被预测单词的上文和下文。**

Mask Language Model：

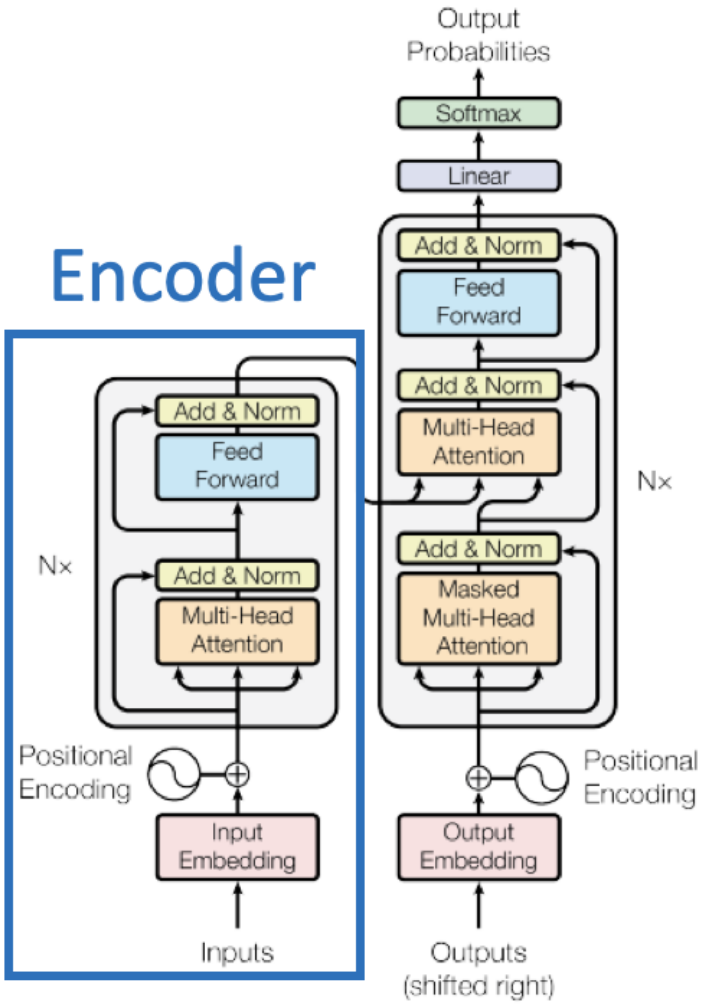


[CLS]标识符：



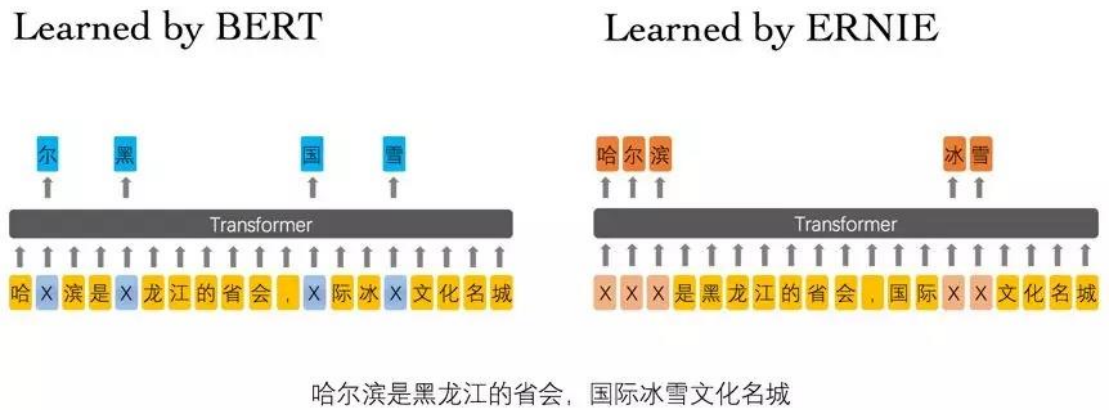
BERT 内部结构：

base 版的参数：Layer:12, Hidden:768, Heads:12

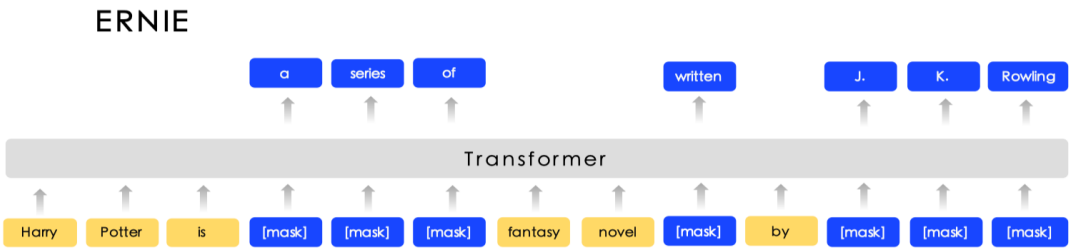


ERNIE 介绍

ERNIE 在 Bert 的基础上增加了短语级的 mask 和实体级的 mask，这样做的好处是让模型能够学习到更多的先验知识，ERNIE 模型通过对词、实体等语义单元的掩码，使得模型学习完整概念的语义表示。ERNIE 对先验语义知识单元进行建模，增强了模型语义表示能力。

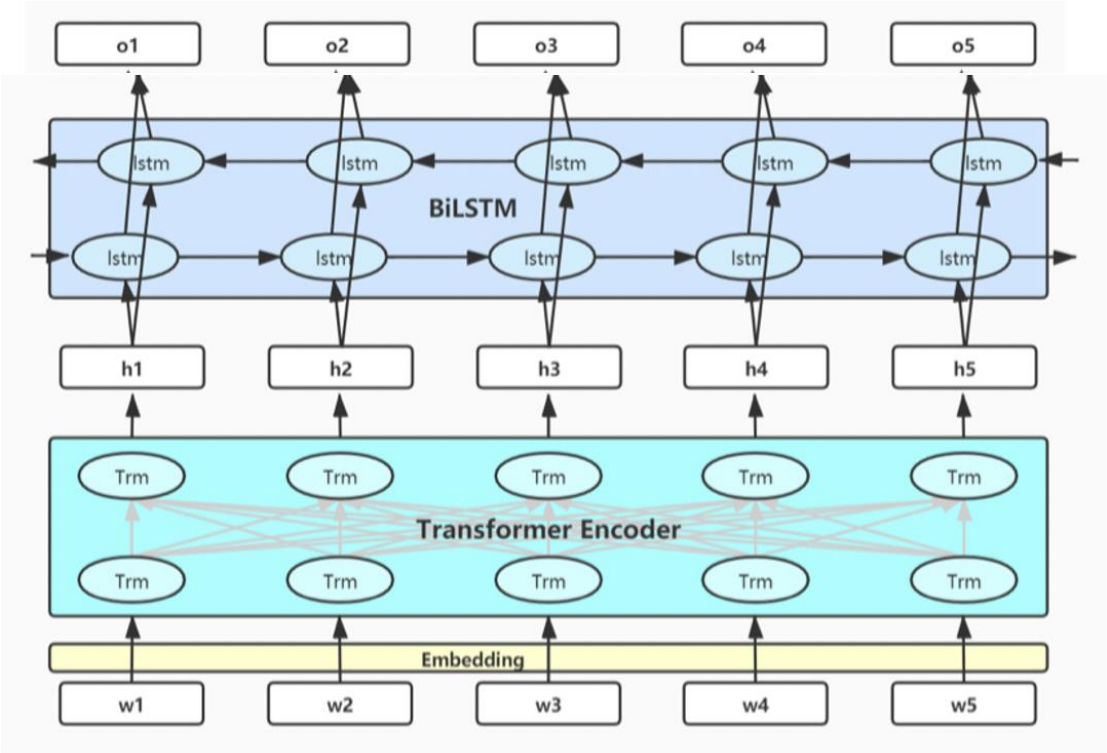


哈尔滨是黑龙江的省会，国际冰雪文化名城



联通 QA 算法模型

模型结构



模型代码

/models/model.py

BERT/ERNIE

```
outputs = self.bert(
    input_ids,
    attention_mask=attention_mask,
)
classifier_output = outputs[1]  # [batch, hidden]
sequence_output = outputs[0]   # [batch, seq_len, embedding]

question_out = self.question_dropout(classifier_output)  # [batch, hidden]
intention_out = self.intention_dropout(classifier_output)  # [batch, hidden]
slot_out = self.slot_dropout(sequence_output)  # [batch, seq_len, embedding]
```

BiLSTM

```
lstm_out, _ = self.lstm(slot_out)   # [batch, seq_len, hidden]
slot_logits = self.slot_classifier(lstm_out)   # [batch, seq_len, s_labels]

intention_out = torch.cat((intention_out, lstm_out[:, -1, :]), -1)  # [batch, hidden*2]
intention_hidden = self.intention_hidden(intention_out)             # [batch, hidden]
question_out = torch.cat((question_out, intention_hidden), -1)      # [batch, hidden*2]
question_hidden = self.question_hidden(question_out)                # [batch, hidden]

question_logits = self.question_classifier(question_hidden)
intention_logits = self.intention_classifier(intention_hidden)  # [batch, i_labels]
outputs = (question_logits, intention_logits, slot_logits)
```

Loss

```
question_loss = F.cross_entropy(question_logits.view(-1, self.question_cls_num), question_labels.view(-1))
intention_loss = F.cross_entropy(intention_logits.view(-1, self.intention_cls_num), intention_labels.view(-1))

active_labels = slot_labels.view(-1)[active_slot]
slot_loss = F.cross_entropy(active_logits, active_labels, ignore_index=-100)

total_loss = 0.8 * question_loss + 0.8 * intention_loss + slot_loss
```

模型训练

- 确保 data 文件下，含有 train.csv 和 test.csv 数据
- 可执行以下命令进行模型训练

python kbqa_classification.py --train --predict

实体链接

- 实体识别
- Trie 搜索树
- 同义词典
- 相似度计算（Jaro–Winkler distance）

https://blog.csdn.net/asty9000/article/details/81348857

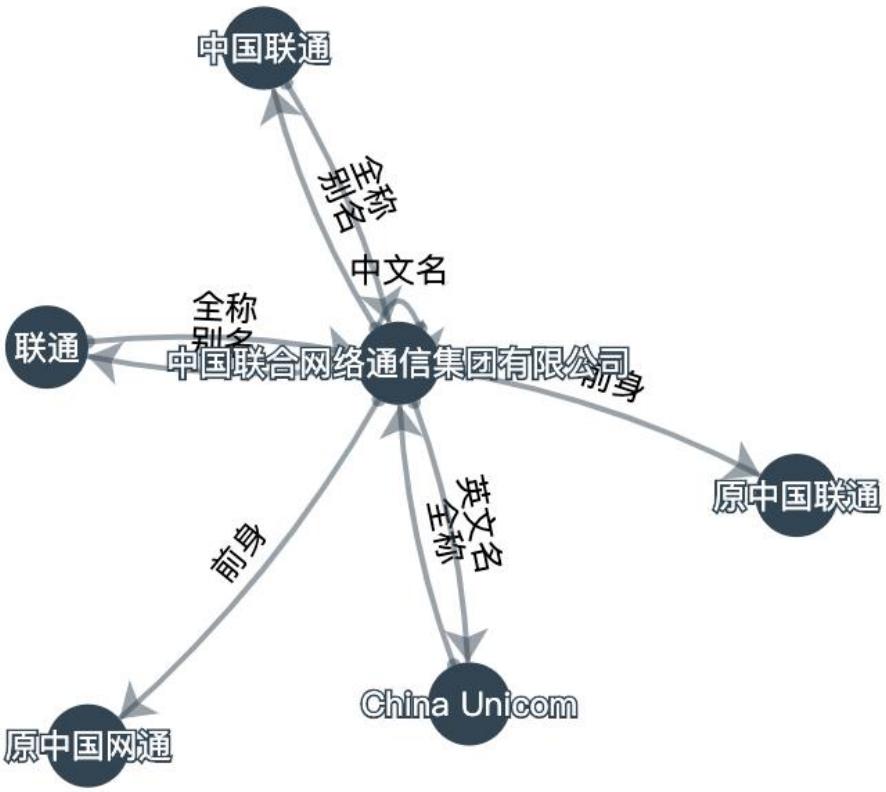
举例：中国的联通的董事长是谁?

模型输出：'ORG': ['中国的联通']

实体链接过程逻辑：

将 “中国的联通” 与 ORG 类别的实体做相似度计算，得到相似度最高的 “中国联通” ，再将“中国联通”映射到“中国联合网络通信集团有限公司”。

实体链接得到的结果：{'中国联合网络通信集团有限公司': '组织'}



代码：

- 相似度计算

/knowledge_search.py

```
def most_similarity_entity(self, entity, label=None):
    similarity_dict = {}
    if label:
        entity_list = self.entity_dict.get(label, None)
        if entity_list:
            for kg in entity_list:
                similarity = Levenshtein.jaro_winkler(u"%s" % entity, u"%s" % kg)
                if similarity:
                    similarity_dict[similarity] = (kg, label)
        else:
            return None
    else:
        for key_label, value_list in self.entity_dict.items():
            for kg in value_list:
                similarity = Levenshtein.jaro_winkler(u"%s" % entity, u"%s" % kg)
                if similarity:
                    similarity_dict[similarity] = (kg, key_label)

    res = None
    if similarity_dict:
        similarity = max(list(similarity_dict.keys()))
        most_sim = similarity_dict[similarity]
        res = (most_sim[0], similarity, most_sim[1])

    return res
```

- 同义词典映射

/entity_link.py

```
entity_res = self.searcher.transfer_name(final_label, final_entity)
# 拿到[全称]
if entity_res:
    entity_label_map[entity_res] = final_label
    entity_map[entity_res] = final_entity
# 说明本身是[全称]
else:
    entity_label_map[final_entity] = final_label
    entity_map[final_entity] = final_entity
```

/answer_search.py

```
def transfer_name(self, label_type, entity_name):
    """
    查找一个实体的全称，如果没有，None
    :param label_type:
    :param entity_name:
    :return:
    """
    transfer_sql = "MATCH (n:{0})-[r:全称]->(m:{0}) where n.name='{1}' return m.name". \
        format(label_type, entity_name)
    res = self.search_single(transfer_sql)
    if res:
        return res[0]['m.name']
    return None
```

4. Neo4j 使用

Neo4j 介绍

图形数据库也称为图形数据库管理系统或 GDBMS。图形数据库是以图形结构的形式存储数据的数据库。 它以节点，关系和属性的形式存储应用程序的数据。 正如 RDBMS 以表的“行，列”的形式存储数据，GDBMS 以“图形”的形式存储数据。

图数据库特点：

如果我们使用 RDBMS 数据库来存储更多连接的数据，那么它们不能提供用于遍历大量数据的适当性能。 在这些情况下，Graph Database 提高了应用程序性能。

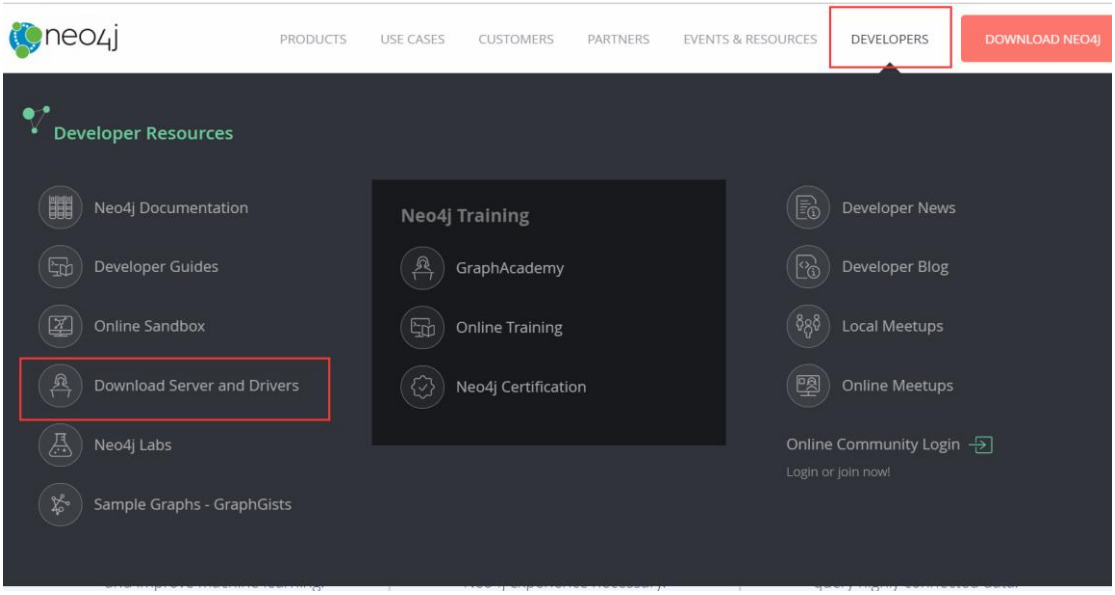
Neo4j 的优点

- 它很容易表示连接的数据
- 检索/遍历/导航更多的连接数据是非常容易和快速的
- 它非常容易地表示半结构化数据
- Neo4j CQL 查询语言命令是人性化的可读格式，非常容易学习
- 使用简单而强大的数据模型

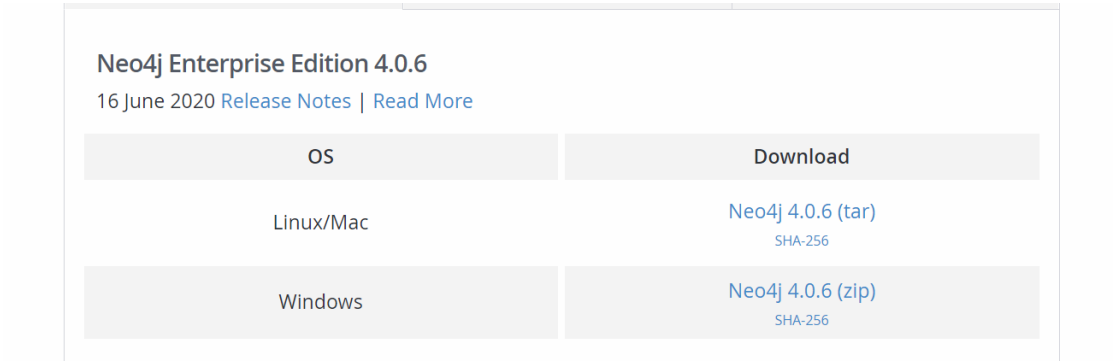
Neo4j 下载与安装

Neo4j 官网： https://neo4j.com/

下载



可根据需要选取所需版本下载。



Neo4j Enterprise Edition 4.0.6

16 June 2020 Release Notes | Read More

OS	Download
Linux/Mac	Neo4j 4.0.6 (tar) SHA-256
Windows	Neo4j 4.0.6 (zip) SHA-256

注：该系统使用的是 linux 下 Neo4j 4.0.3-community 版本。

安装

直接解压到安装目录即可。

此外，neo4j 是用 Java 语言编写的图形数据库，运行时需要启动 JVM 进程，因此，需安装 JAVA SE 的 JDK。我们安装的是 java 14 版本，安装完后，配置环境变量，输入 ‘java --version’ 命令可检查配置。(这里不展开说明)

首次使用

- 修改配置

1. 进入 neo4j 目录下的 conf 文件夹，打开 neo4j.conf 文件。

2. 修改第 9 行，去掉该行的#注释符号，可以自己定义数据库名称
   dbms.default_database=neo4j
3. 修改第 54 行，去掉该行的#注释符号，可以远程通过 ip 访问 neo4j 数据库
   dbms.connectors.default_listen_address=0.0.0.0

此外还有一些参数可以自行设置。

- 常用命令

进入 bin 目录可执行如下命令：

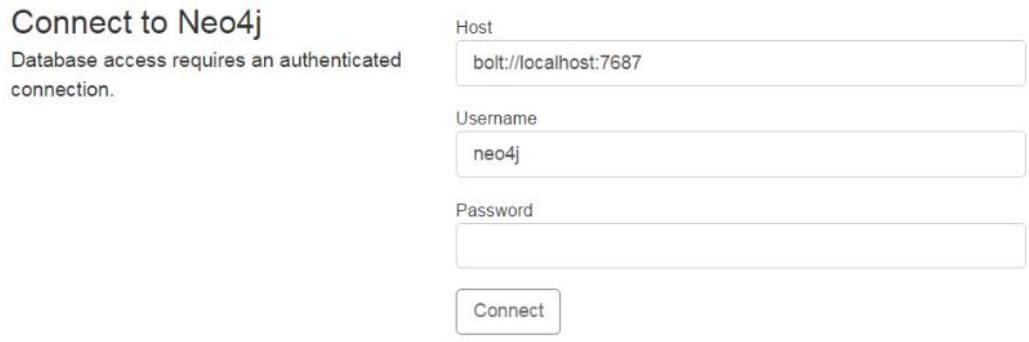
后台启动：neo4j start
前台启动：neo4j console
查看状态：neo4j status
停止：neo4j stop
重启：neo4j restart

- 客户端访问

1. 进入 bin 目录后，执行 neo4j console 命，我们可以看到如下界面：

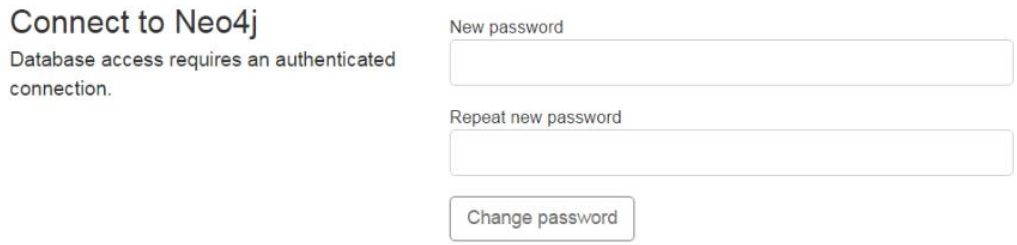
```
020-06-18 06:49:58.330+0000 INFO  ======== Neo4j 4.0.3 ========
020-06-18 06:49:58.437+0000 INFO  Starting...
020-06-18 06:50:30.127+0000 INFO  Bolt enabled on localhost:7687.
020-06-18 06:50:30.128+0000 INFO  Started.
020-06-18 06:50:33.289+0000 INFO  Remote interface available at http://localhost:7474/
```

2. 如上图所示，已经开启了 Neo4j 数据库，可以在浏览器中登录 http://localhost:7474 网址查看数据库。
3. 首次登录会看到如下界面：



默认的账号和密码均为 neo4j。

登陆后如下图：



重新设置自己的账户名和密码后，则可进入主页面，至此 neo4j 配置成功。

数据导入

1. 导入工作前，需要关闭 neo4j 服务。
2. 准备好导入的 csv 文件，各个类型的实体表，以及通过实体定义的 relation 表。
3. 切换到 neo4j 解压缩目录下的 bin 文件夹，windows 使用 neo4j-admin.bat，linux 使用 neo4j-admin。

命令格式如下：

```
neo4j-admin.bat
import --database lt   # 数据库名字
--nodes graph_data/称号.csv --nodes graph_data/国家.csv  # 实体节点文件
--relationships graph_data/relation.csv   # 关系文件
--skip-duplicate-nodes=true --ignore-extra-columns=true --ignore-empty-strings=true  #可选

# 以上语句全是连接的并不需要分行
# graph_data 只是数据存放路径，视情况定
```

实际上，我们已经提供了自动化导入的脚本文件，可参考第 2 节数据构建的导入 Neo4j 数据库部分。

数据管理

1. 选择数据库：
   在 conf 目录下，更改 neo4j.conf 文件的 `dbms.default_database=默认数据库名称`，来启动所需的数据库。

2. 删除数据库：
   在 data/databases 和 data/transactions 目录下删除同名数据库。

Cypher 查询语言介绍

可参考资料：https://www.w3cschool.cn/neo4j/neo4j_cql_introduction.html

Neo4j CQL - CREATE 命令

- 创建没有属性的节点
- 使用属性创建节点
- 在没有属性的节点之间创建关系
- 使用属性创建节点之间的关系
- 为节点或关系创建单个或多个标签

CREATE命令语法

```
CREATE (<node-name>:<label-name>)
```

语法说明

语法元素	描述
CREATE	它是一个Neo4j CQL命令。
<node-name>	它是我们要创建的节点名称。
<label-name>	它是一个节点标签名称

Neo4j CQL - MATCH 命令

- 从数据库获取有关节点和属性的数据
- 从数据库获取有关节点，关系和属性的数据

MATCH命令语法：

```
MATCH
(
   <node-name>:<label-name>
)
```

语法说明

语法元素	描述
<node-name>	这是我们要创建一个节点名称。
<label-name>	这是一个节点的标签名称

Neo4j CQL - RETURN 子句

- 检索节点的某些属性
- 检索节点的所有属性
- 检索节点和关联关系的某些属性

- 检索节点和关联关系的所有属性

RETURN命令语法：

```
RETURN
   <node-name>.<property1-name>,
   ........
   <node-name>.<propertyn-name>
```

语法说明:

语法元素	描述
<node-name>	它是我们将要创建的节点名称。
<Property1-name>...<Propertyn-name>	属性是键值对。 <Property-name>定义要分配给创建节点的属性的名称

举例如下：

1.

```
def parse_fact(self, entity_dict, sub_intention):
    '''
    事实类 forward
    '''
    sqls = []
    for entity, label in entity_dict.items():
        sqls.append("MATCH (n:{0})-[r:{1}]->(m) where n.name = '{2}' return n.name, m.name, labels(m)".format(label, sub_intention, entity))

    return sqls
```

"MATCH (n:{0})-
[r:{1}]->(m) where n.name = '{2}' return n.name, m.name, labels(m)".format(label, sub_intention, entity)

该语句根据一个实体的标签和名字以及一个关系来找到另外的实体。

2.

```
def parse_describe(self, entity_dict, sub_intention):
    '''
    描述类
    '''
    sqls = []
    for entity, label in entity_dict.items():
        sqls.append("MATCH (n:{0}) where n.name = '{1}' return n.name, n.{2}".format(label, entity, sub_intention))

    return sqls
```

"MATCH (n:{0}) where n.name = '{1}' return n.name, n.{2}".format(label, entity, sub_intention)

该语句查询实体相关属性。

py2neo 介绍

Py2neo 是一个客户端库和工具包，可从 Python 应用程序内部和命令行使用 Neo4j。

可参考资料：https://py2neo.org/v5/

```python
from py2neo import Graph

import kbqa_config as fileConfig


class AnswerSearcher:
    def __init__(self):
        self.g = Graph(
            host=fileConfig.neo4j_host_ip,
            http_port=fileConfig.neo4j_host_port,
            user=fileConfig.neo4j_username,
            password=fileConfig.neo4j_pwd)
```

根据实际情况配置好参数，例子如下图所示：

```python
    # self.g = Graph(
    #     host="127.0.0.1",
    #     http_port=74,
    #     user="neo4j",
    #     password="aa123456")
```

初始化后，可按照如下语句进行 cypher 语句查询

```python
def search_single(self, sql):
    '''
    单句cypher查询
    '''
    res = self.g.run(sql).data()
    return res
```