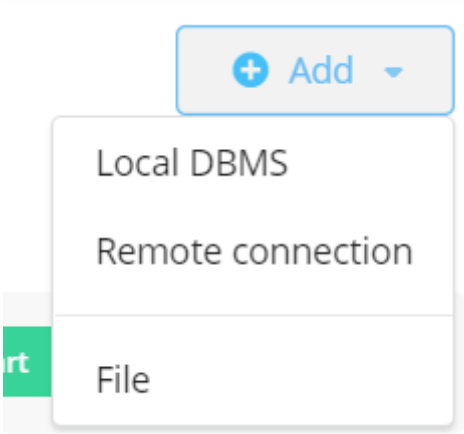
具体操作:

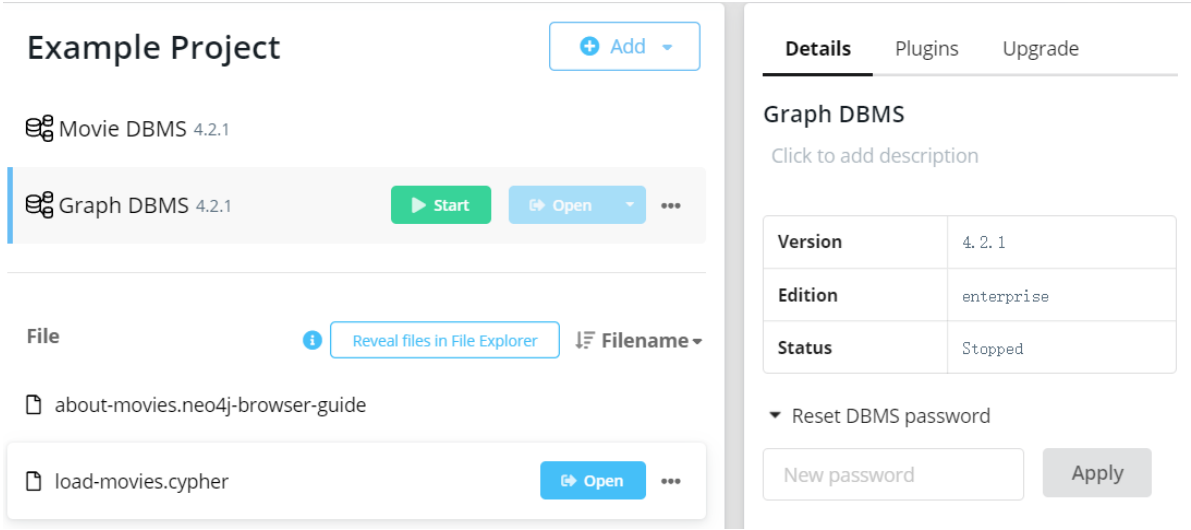
Knowledge Graph 模型

1. 准备工作， https://neo4j.com/try-neo4j/ 下载Neo4j desktop数据库，和MySQL一样的性质，一路无脑安装。

2. 打开Neo4j等它初始化，首次运行大概10分钟的样子，进去之后新建一个数据库，随便起个名字.

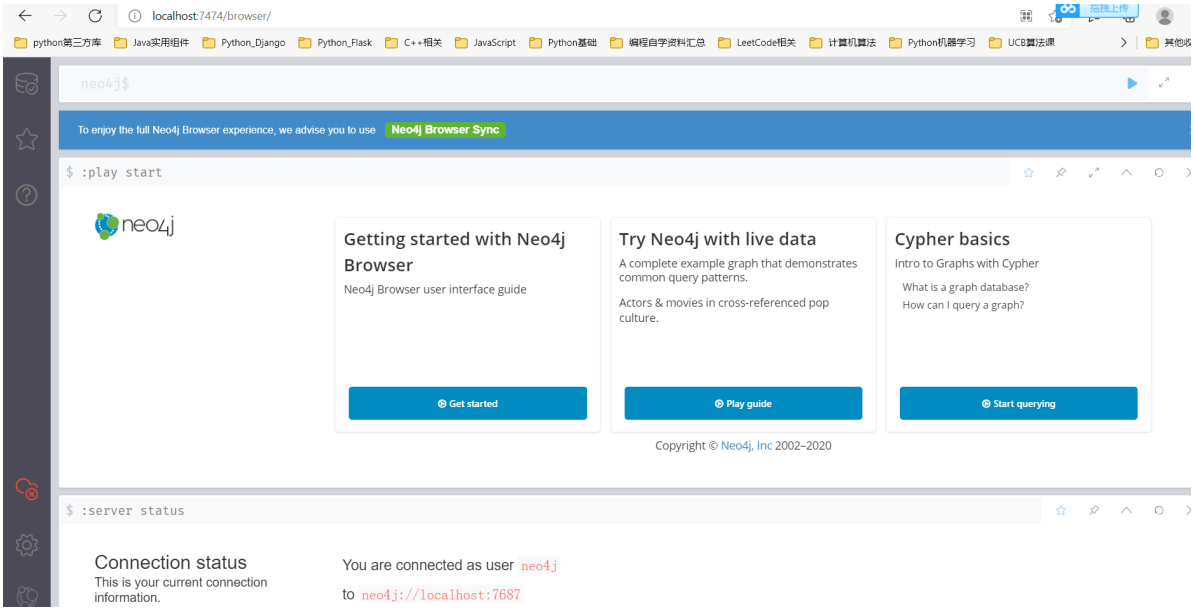


然后右下角点击重设密码, 随便设一个123456, 然后apply.

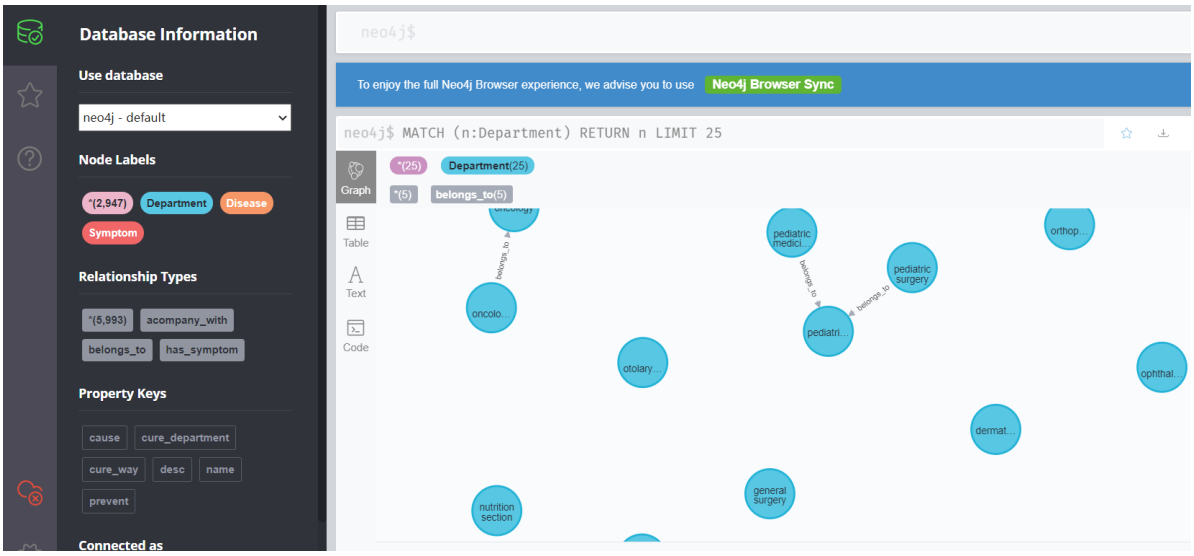


然后点击start开启后端服务，到时候demo的时候Tkinter在前端运行，这个数据库同时要在后端运行。

3. 找到Core/knowledgeGraph/buildKnowledgeGraph.py, 在**init**中修改dataPath为medical_knowledge_base.csv在你电脑中的位置。点击运行这个py文件，会自动把所有信息写入数据库。等待5分钟左右。如果想要看建立的节点结果，就浏览器打开一个网页，输入localhost:7474/broswer/会弹出数据库的前端页面.



点击左上角的数据库标志查看节点



4. 最终的Tkinter前后端调用方法在Core/KnowledgeGraph/chatbot_graph.py中，方法名是chat_main(), 你点了send就执行这个方法，方法的返回值你就展现到上面那个框里。

LSTM Model

1. 有两个版本，jupyter版本在Core/data/QA_Pairs/ReadFiles.ipynb
2. pycharm版本在Core/models/HBDA/test.py, 效果一样

   test.py和jupyter文件的最下面是我写的聊天功能，会返回答案和相关urls.

test.py

```
271
272
273     # 测试方法
274     print("Welcome to the medical QA bot, press q to quit")
275     quit = False
276     while not quit:
277
278         X_input = input("User:")
279         if X_input == "q":
280             print("Goodbye!")
281             break
282         words = text_to_word_list(flag, X_input)
283         input_seq = word_to_index(words)
284
285         count = 0
286         prob_list = []
287         for index, question in enumerate(questions_list):
288             if count < 100:
289                 words_list = text_to_word_list(flag, question)
290                 compare = word_to_index(words_list)
291                 temp_df = pd.DataFrame({"question1_n": [0], "question2_n": [0]}, dtype=
292                 temp_df.at[0, "question1_n"] = input_seq
293                 temp_df.at[0, "question2_n"] = compare
294                 temp = split_and_zero_padding(temp_df, 10)
295                 result = model([temp["left"], temp["right"]])
296                 prob_list.append((index, result.numpy()[0][0]))
297             count += 1
298         prob_list.sort(key=lambda x: x[1], reverse=True)
299
300         import random
301
302         top1 = prob_list[:1]
303         answer_list = []
```

```
    top1 = prob_list[:1]
    answer_list = []
    url_list = []
    index = 0
    for candidate in top1:
        question_index = candidate[0]
        question_sentences = list(dataset2[dataset2.question == questions_list[quest
        url_reference = list(dataset2[dataset2.question == questions_list[question_i
        answer_list.extend(question_sentences)
        url_list.extend(url_reference)

    randindex = random.randint(0, len(answer_list) - 1)
    randindex_url = random.randint(0, len(url_list) - 1)
    answer = answer_list[randindex]
    url = url_list[randindex_url]

    print("Medic:", answer)
    print("Medic:", url)
```

X_input就是输入，answer和url就是输出，你展示到页面上就行。

3. 关于embedding, 用的是GoogleNews_Negative_300预训练embedding matrix, 文件单独发你。

test.py

```
202             embedded[index] = word_index
203
204         return np.array(embedded)
205
206
207
208
209     # 修改成你的
210     TRAIN_CSV = r'C:\Users\DELL\Desktop\MIT_Final_Project\Core\data\Train_Test
211
212
213     flag = 'en'
214
215     # 这个我另外发送
216     embedding_path = r'F:\Downloads\GoogleNews-vectors-negative300.bin.gz'
217     embedding_dim = 300
218     max_seq_length = 10
219
220     # 加载词向量
```

embedding_path也要改成你的。