

# 上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

## 学士学位论文

THESIS OF BACHELOR



论文题目： 知识图谱问答中的语义理解

学生姓名： 曹瑞升

学生学号： 5140219073

专业： 计算机科学与技术

指导教师： 俞凯教授

学院(系)： 电子信息与电气工程学院

# 上海交通大学

## 毕业设计（论文）学术诚信声明

本人郑重声明：所呈交的毕业设计（论文），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者签名：\_\_\_\_\_

日期：\_\_\_\_\_年\_\_\_\_\_月\_\_\_\_\_日

# 上海交通大学

## 毕业设计（论文）版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保 密 ，在\_\_年解密后适用本授权书。

本学位论文属于

不保密 。

(请在以上方框内打“√”)

作者签名: \_\_\_\_\_ 指导教师签名: \_\_\_\_\_

日 期: \_\_\_\_ 年 \_\_\_\_ 月 \_\_\_\_ 日      日 期: \_\_\_\_ 年 \_\_\_\_ 月 \_\_\_\_ 日

# 知识图谱问答中的语义理解

## 摘要

自动问答系统旨在对用户提出的问题提供直接答案，而知识图谱结构化地组织、管理和理解海量信息的能力，为计算机理解用户询问的意图提供了丰富的背景知识。知识图谱给语义理解带来了新的契机，同时也在智能问答中显示出强大活力，已经成为了智能服务的基础设施。

本文从商业应用的基点出发，首先调研了多种语义表达方式，包括语义框架 (Semantic Frame)、对话语义动作 (Dialogue Act)、类  $\lambda$  演算的逻辑形式 (Logical Form)，接着深入学习了现有的语义槽填充 (slot filling) 和意图分类 (intent classification) 任务的多种神经网络模型，然后归纳总结了基于知识图谱的问答中常见的任务类型以及最主要的解析难点，由此制定了我们的研究思路。

本论文中，我们首先创新性提出并建立了一套完善的适用于知识图谱问答中的逻辑形式，并针对这种语义表达方式，尝试了基于规则的有限状态转移机方法和基于统计的神经网络模型两种方法。针对两个不同的领域，一个规则体系较完善、上线测试并不断维护的企业信息问答 AISpeech 领域，另一个初步建立、没有任何先验知识或原始数据的实验室信息问答 SpeechLab 领域，我们逐步分析了多种神经网络模型的差异，包括双向循环网络模型 (Bi-RNN)、双向循环网络加条件随机场模型 (Bi-RNN+CRF)、局部注意力模型的变体 Focus 模型，并且比较了规则、统计两种实现方法之间的优劣。

从规则模板的编写，到有限状态机的构建、训练数据的自动生成、测试数据的半自动化众包收集，最后对各种方法模型的尝试，一步一步建立起完善的知识图谱语义解析系统，并取得了预期的实验结果。统计实验的结果明显优于传统的规则系统，并且在未来呈现出更大的发展潜力。

**关键词：**自然语言理解 知识图谱 问答系统 自然语言处理

# NATURAL LANGUAGE UNDERSTANDING IN QUESTION ANSWERING SYSTEM BASED ON KNOWLEDGE GRAPH

## ABSTRACT

The Automatic Question Answering System aims to provide direct answers to questions asked by users, and the ability of Knowledge Graph Database to organize, manage and understand massive structured information, provides a profound background knowledge for the computer to identify the intent of user's inquiries. Knowledge Graph brings new opportunities for Natural Language Understanding, and it also shows great vigor in intelligent Question Answering System. Nowadays, it has become the infrastructure of intelligent services.

Starting from the basic point of commercial application, we first investigate a variety of semantic representations, including Semantic Frame, Dialogue Act, and the Logical Form like  $\lambda$  calculus. We then delve into the existing semantic slot filling and intent classification tasks, and do some survey on the current popular neural network models. Then we summarize the common types of queries based on Knowledge Graph and list the main problems we confront with, thus formulate our research ideas and plans.

In this thesis, we first innovatively propose and establish a full set of logical form suitable for Question Answering System based on Knowledge Graph. With respect to this semantic expression, we try a rule-based Finite State Transducer method and statistics-based neural network models. Aiming at two different domains, one is AISpeech about enterprise information with a perfect rule system and continuous maintenance during on-line testing, and the other is a preliminarily established SpeechLab domain about laboratory information without any prior knowledge or raw data. We gradually analyze the differences among various neural network models, including Bi-RNN, Bi-RNN plus CRF and one variant local attention model called Focus, and compare the performances of rule-based method with statistics-based method. We point out both the advantages and disadvantages of these two methods.

From the design of rule templates, to the construction of finite state machines, the automatic generation of training data, semi-automatic crowdsourcing collection of test data, and eventually to the experiments of various methods and models, a complete Natural Language Understanding System based on Knowledge Graph is established step by step and finally we achieve the expected experimental results. The performance of statistical models are clearly superior to the rules system and it also shows greater development potential in the future.

**KEY WORDS:** Natural Language Understanding, Knowledge Graph, Question Answering System, Natural Language Processing

## 目 录

插图索引	VII
表格索引	VIII
算法索引	IX
主要符号对照表	X
<b>第一章 绪论</b>	<b>1</b>
1.1 研究背景 . . . . .	1
1.2 研究现状及相关工作 . . . . .	2
1.2.1 知识图谱与本体概述 . . . . .	2
1.2.2 语义表达方式和评价标准 . . . . .	3
1.2.3 基于知识图谱的语义理解 . . . . .	6
1.3 本课题的研究内容及贡献 . . . . .	7
<b>第二章 背景知识</b>	<b>8</b>
2.1 有限状态转移机 . . . . .	8
2.1.1 数学模型 . . . . .	8
2.1.2 有限状态转移机基本操作 . . . . .	8
2.2 条件随机场 . . . . .	9
2.2.1 概率无向图模型 . . . . .	9
2.2.2 线性链条件随机场 . . . . .	10
2.2.3 概率计算 . . . . .	11
2.2.4 预测标记序列 . . . . .	13
2.3 神经网络 . . . . .	13
2.3.1 神经元和前馈网络 . . . . .	13
2.3.2 循环神经网络及其变体 . . . . .	15
2.3.3 Sequence-to-Sequence 模型及其变体 . . . . .	18
2.3.4 Attention 模型 . . . . .	19
<b>第三章 语法规则定义及规则解析</b>	<b>21</b>
3.1 语法规则定义 . . . . .	21
3.1.1 任务分类 . . . . .	21
3.1.2 语义解析难点分析 . . . . .	22

3.1.3 规则定义 . . . . .	23
3.1.4 中间语义表示及逻辑形式 . . . . .	25
3.2 基于 FST 的规则解析原理 . . . . .	28
<b>第四章 数据集来源及神经网络统计模型</b>	<b>31</b>
4.1 数据集来源 . . . . .	31
4.1.1 AISpeech 领域 . . . . .	31
4.1.2 SpeechLab 领域 . . . . .	32
4.2 神经网络统计模型 . . . . .	34
4.2.1 子任务分类 . . . . .	34
4.2.2 结构化语义扁平化处理 . . . . .	35
4.2.3 数据预处理 . . . . .	36
4.2.4 Bi-RNN 模型 . . . . .	37
4.2.5 Bi-RNN+CRF 模型 . . . . .	38
4.2.6 Attention 模型分类及 Focus 模型 . . . . .	38
4.2.7 解码及后处理 . . . . .	41
<b>第五章 实验及结果分析</b>	<b>45</b>
5.1 实验参数设置 . . . . .	45
5.1.1 固定参数 . . . . .	45
5.1.2 超参数 . . . . .	45
5.2 评测方法 . . . . .	45
5.3 AISpeech 领域实验结果及分析 . . . . .	46
5.3.1 FST 规则解析 . . . . .	46
5.3.2 神经网络统计方法解析 . . . . .	46
5.4 SpeechLab 领域实验结果及分析 . . . . .	49
5.4.1 FST 规则解析 . . . . .	50
5.4.2 神经网络统计方法解析 . . . . .	50
5.5 实验总结分析 . . . . .	52
<b>第六章 总结及未来工作</b>	<b>54</b>
6.1 研究总结 . . . . .	54
6.2 成果展示 . . . . .	54
6.3 未来工作展望 . . . . .	56
<b>附录 A 基于 FST 的规则解析过程详解</b>	<b>57</b>
<b>附录 B SpeechLab 领域本体结构设计</b>	<b>60</b>
<b>附录 C SpeechLab 解析规则示例</b>	<b>61</b>

---

附录 D SpeechLab 问卷及测试数据收集样例	<b>62</b>
参考文献	<b>65</b>
致 谢	<b>67</b>

## 插图索引

1-1 知识图谱子图示例 . . . . .	2
1-2 ShowFlight 语义框架示意图 . . . . .	3
1-3 ShowFlight 语义框架实例化 . . . . .	4
1-4 自然语言转化为 LambdaDCS 逻辑形式树状图 . . . . .	5
1-5 基于语义解析的知识图谱智能问答示意图 . . . . .	6
2-1 有限状态转移机示意图 . . . . .	9
2-2 带权重的有限状态转移机示意图 . . . . .	9
2-3 线性链条件随机场 . . . . .	10
2-4 神经元模型 . . . . .	14
2-5 前馈神经网络示意图 . . . . .	15
2-6 长短期记忆单元示意图 . . . . .	16
2-7 循环门控单元示意图 . . . . .	17
2-8 Seq2Seq 模型示意图 . . . . .	18
2-9 注意力模型示意图 . . . . .	20
3-1 词库文件和引用格式 . . . . .	23
3-2 限制定义和引用格式 . . . . .	24
3-3 规则模板详解 . . . . .	26
3-4 逻辑形式样例 1 . . . . .	29
3-5 逻辑形式样例 2 . . . . .	30
4-1 构建训练集流程 . . . . .	33
4-2 改进后构建测试集流程 . . . . .	33
4-3 槽填充任务示例 . . . . .	34
4-4 逻辑形式扁平化处理 . . . . .	35
4-5 双向 RNN 网络模型 . . . . .	37
4-6 双向 RNN+CRF 网络模型 . . . . .	39
4-7 局部注意力模型 . . . . .	40
4-8 Focus 模型结构图 (Pre-Attention 和 Post-Attention 两种) . . . . .	40
5-1 训练过程 Epoch Loss 变化情况 . . . . .	48
5-2 训练过程中 Epoch Loss 上升 . . . . .	48
5-3 测试集不同 slot 的 P-R-F1 分数 . . . . .	48
5-4 测试集句子分类 Fscore . . . . .	49

---

6-1 SJTUSpeechLab 微信公众号智能问答系统展示 . . . . .	55
D-1 SpeechLab 领域测试数据收集问卷样例 1 . . . . .	63
D-2 SpeechLab 领域测试数据收集问卷样例 2 . . . . .	64

## 表格索引

1-1 对话语义动作示例 . . . . .	4
1-2 分类结果混淆矩阵 . . . . .	5
5-1 AISpeech 领域基于 FST 规则解析实验结果 . . . . .	46
5-2 AISpeech 领域基于神经网络解析实验结果——测试集 6274 句 . . . . .	46
5-3 AISpeech 领域基于神经网络解析实验结果——测试集 2616 句 . . . . .	47
5-4 AISpeech 领域基于神经网络解析实验结果——测试集 8890 句 . . . . .	47
5-5 SpeechLab 领域基于 FST 规则解析实验结果 . . . . .	50
5-6 SpeechLab 领域基于神经网络解析实验结果 . . . . .	50
C-1 SpeechLab 领域规则示例 . . . . .	61
D-1 SpeechLab 领域测试数据收集样例 . . . . .	62

## 算法索引

2-1 条件随机场的维特比算法 . . . . .	13
4-1 BIO-Decoder 算法 . . . . .	41
4-2 Beam Search 解码算法 . . . . .	42
4-3 提取槽值对算法 . . . . .	43
4-4 构建语义逻辑形式算法 . . . . .	44

## 主要符号对照表

<i>NLU</i>	自然语言理解 Natural Language Understanding
<i>KG</i>	知识图谱 Knowledge Graph
<i>QA</i>	问答系统 Questioning Answering
<i>aRb</i>	实体 <i>a</i> 的关系 <i>R</i> 是 <i>b</i> , 或实体 <i>a</i> 和 <i>b</i> 具有关系 <i>R</i>
<i>eAv</i>	实体 <i>e</i> 的属性 <i>A</i> 的值是 <i>v</i>
<i>RDF</i>	资源描述框架 Resource Description Framework
<i>FST</i>	有限状态转移机 Finite State Transducer
<i>I(·)</i>	指示函数, (·) 内条件满足为 1, 否则为 0
<i>RNN</i>	循环神经网络 Recurrent Neural Network
<i>LSTM</i>	长短期记忆网络 Long Short-Term Memory
<i>GRU</i>	门控循环单元 Gated Recurrent Unit
<i>Seq2Seq</i>	序列到序列模型 Sequence to Sequence Model
<i>AM</i>	注意力模型 Attention
<i>Focus</i>	专注模型, 一种局部注意力模型的变体
$\sigma(x)$	sigmoid 函数: $\sigma(x) = \frac{1}{1+e^{-x}}$
$tanh(x)$	tanh 函数: $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

# 第一章 绪论

## 1.1 研究背景

随着人工智能的发展和深度学习的盛行，车载、机器人、家居和客服等领域提供的服务已经越来越趋向智能化。其中的智能客服系统，建立在大规模知识库的基础之上，以自动问答为最终目的，运用知识融合、自然语言理解、对话管理、文本生成等相关技术，为使用客户提供专业化的查询和沟通服务，并帮助企业极大地节省呼叫中心人工服务的投入。2018 年 Google I/O 开发者大会上 CEO 桑达尔·皮查伊 (Sundar Pichai) 展示了谷歌助手 (Google Assistant) 打电话进行理发预约的场景<sup>1</sup>。智能客服人性化的服务和专业化知识已经开始逐渐取代企业重复性的问答工作。

2012 年 5 月 Google 花重金收购 Metaweb 公司，并首次提出知识图谱的概念，自此之后各种开放性领域大规模的知识图谱，诸如 NELL、FreeBase、DBpedia 等，纷纷出现。知识图谱本质上是一种语义网络，网络中的节点表示实体或概念，边表示实体或概念之间的语义关联，通过实体或概念之间的关联来管理知识。这种网络结构化的知识管理方式充分体现了实体之间的语义关联，非常适合自动问答这种对语义关联性有较强依赖的应用场景。基于知识图谱的自动问答 (Automatic Question Answering Over Knowledge Graph)，作为一种新型的问答系统，在利用知识图谱 (Knowledge Graph) 中的知识针对人们提出的自然语言问句进行回复的表现上日益突出。如无特殊指明，本论文中提及的自动问答和对话系统都是以知识图谱作为后端支持。

一个完整的自动问答系统中用户提出的语音询问，经过语音识别模块能够从音频输入转化为文字信息，但原始的自然语言文字只能被机器记录并保存，计算机并不能“理解”其中传达的含义。通常的自然语言理解 (Natural Language Understanding, NLU) 任务，就是将纯文本信息转化为一种计算机能够解析的结构化信息，称为语义表达式。比如用户说了一句

给我预订一张明天从上海到北京的火车票

其中包含的重要语义信息有

出发时间 = 明天	交通工具 = 火车
出发地 = 上海	目的地 = 北京
意图 = 订票	

语义理解模块针对目标领域定义一套语义表达形式<sup>2</sup>来组织这些语义信息并传递给对话管理模块，从而执行相应的查询检索或对话动作。自然语言理解的任务是异常复杂的——自然语言本质上具有歧义性 (比如苹果既可以指一种水果，又可以指代一家公司) 和多样性 (比如上海交通大学可以简写为上海交大或 SJTU)。人类在理解自然语言时不仅需要理解每个文字或单词的含义，还要依赖上下文并且需要“默认”的背景知识或常识作为额外的支持，而基于知识图谱的语义理解为我们提供了这

<sup>1</sup> 视频链接: <https://weibo.com/5575548730/GfZNYANew?type=comment>

<sup>2</sup> 语义信息的表达形式有很多，不同的任务、不同的领域采取的组织方式也不同，参见 1.2.2

些丰富的背景知识，以及方便查询这些知识的组织结构方式。随着深度学习在自然语言处理诸多任务上的成功应用，以知识图谱为数据支持，以自然语言理解为核心的智能客服问答系统必将提供更加专业并且人性化的服务。

## 1.2 研究现状及相关工作

### 1.2.1 知识图谱与本体概述

本体 (ontology) 是对一个领域的定义，提供了该领域的语义范围的规范化说明，关于智能语音实验室的本体结构参见附录B。将本体学应用于知识图谱上，每一个领域的知识图谱都是由概念、实体、属性、关系几个部分组成：

**概念 Concept** 对一类都具有某些属性或关系的实体的统称，可以理解为实体的类别 (type)。概念的涵盖范围取决于其下具体的实体，比如对于汽车、飞机、轮船这一类实体，它们的概念可以定义为交通工具，而对于具体的不同汽车品牌的实体，它们的概念定义为汽车

**实体 Entity** 概念的实例化体现，每一个实体都有特定的一些具体的关系对象和属性值

**属性 Attribute** 一个或一类实体共有的性质，由一个具体的实体可以确定其属性值。属性值的表示常常通过基本的数据类型，比如整数 (int，班级的人数)、浮点数 (float，企业某一年的利润值)、字符串 (string，对某个人的生平简介) 或是可枚举的类别 (性别男女)

**关系 Relation** 区别于属性，关系是定义在两个实体之间的关联，且往往具有双向性，即“逆关系”，比如一篇论文  $a$  的作者是  $b$  ( $a$ —author— $b$ )，那么人物  $b$  发表的文献就有  $a$  ( $b$ —publication— $a$ )，关系 author 和关系 publication 互为逆关系。两个实体之间可能有多个不同的关系，一个实体也可以和多个不同的实体之间有着同一种关系

以实验室的人员俞凯教授为例，与其相关的知识图谱子图的网络结构参见图1-1

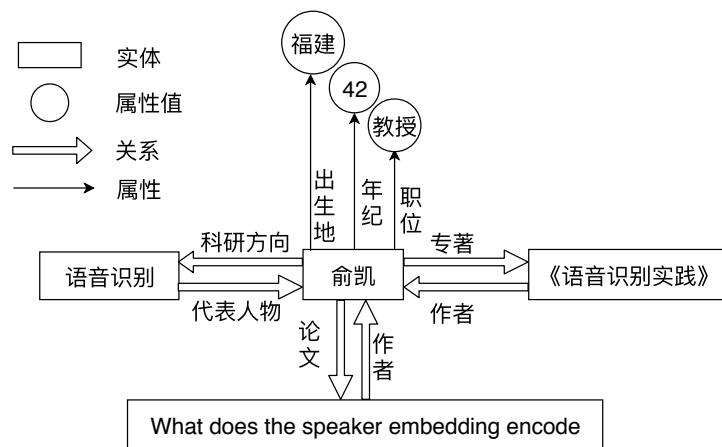


图 1-1 知识图谱子图示例

Figure 1-1 Example of A Subgraph in Knowledge Graph

知识图谱的网络结构在外部存储中通过“实体  $a$ —关系  $R$ —实体  $b$ ”或“实体  $e$ —属性  $A$ —属性值  $v$ ”的三元组来表示，这种以  $aRb$  或  $eAv$  的三元组描述知识的数据模型称为资源描述框架 (Re-

source Description Framework, RDF), 也是大多数领域相关的知识图谱采取的组织方式。

## 1.2.2 语义表达方式和评价标准

### 1.2.2.1 语义表达方式

在1.1中我们对自然语言理解的任务概括性地描述为将纯文本信息转化为一种计算机能够“理解”的语义表达式。这里介绍几种常用的语义表达方式：

1. 语义框架 (Semantic Frame)，每一个语义框架包含一系列带类型的语义槽(slot)<sup>1</sup>，语义槽的类型(type)限定了它可以被哪一类的值填充。比如图1-2中 Flight 语义框架<sup>[1]</sup> 里语义槽“DCity”(出发城市) 和 “ACity”(到达城市) 的填充类型都为 “City”<sup>2</sup>，表明这两个语义槽允许被填充的值是城市名。

```
<frame name="ShowFlight" type="Void">
  <slot name="subject" type="Subject">
    <slot name="flight" type="Flight">
  </frame>
  <frame name="GroundTrans" type="Void">
    <slot name="city" type="City">
      <slot name="type" type="TransType">
    </frame>
    <frame name="Flight" type="Flight">
      <slot name="DCity" type="City">
      <slot name="ACity" type="City">
      <slot name="DDate" type="Date">
    </frame>
```

图 1-2 ATIS 数据集 ShowFlight 语义框架

Figure 1-2 ShowFlight Semantic Frame

一个输入句子的语义表示就是相应语义框架的一个实例化，比如输入语句

Show me flights from Seattle to Boston on Christmas Eve.

的语义框架参见图1-3

上面的例子中 ShowFlight 语义框架内嵌套了 Flight 等子语义框架，这种层次化的语义表示具有强大的表示能力，支持框架之间共享一些子结构，但解析起来相对困难。另一种更加常用的变体是对语义框架进行扁平化处理，不允许语义框架包含子结构，语义表示被简化为一系列的语义槽值对(slot=value)，比如上面的句子被解析成

Show me [flights:Subject] from [Seattle:DCity] to [Boston:ACity] on [Christmas Eve:DDate].

<sup>1</sup>语义槽是一句话中语义信息的载体，可以理解为自然语言中归一化后的关键字(keyword)

<sup>2</sup>City 也是一个语义框架，这里没有对其定义

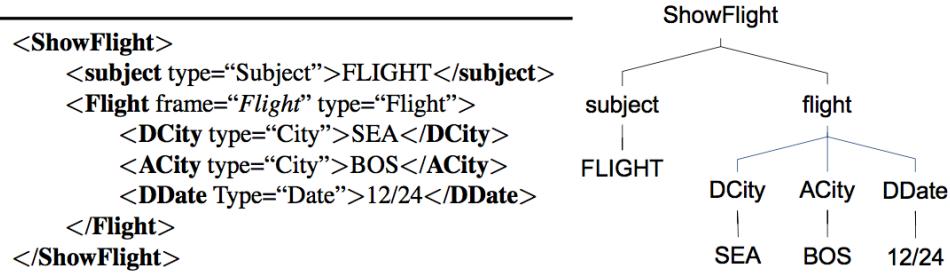


图 1-3 ShowFlight 语义框架实例化

Figure 1-3 ShowFlight Semantic Frame Example

扁平化的语义表示更加简单易懂，并且可以构造更简单的统计模型细分为槽值填充(slot filling)<sup>1</sup>和意图分类(intent classification)<sup>2</sup>两个子问题联合解决

- 对话语义动作(Dialogue Act)，相比语义框架，对话语义动作侧重于对话行为的表示上。Traum<sup>[2]</sup>分析了对话系统中行为的概念，总结归纳了请求确认(confirm)、否认(deny)、询问(request)、告知(inform)等具体对话行为。但仅凭这些分类信息不能完整传达表达者的意图，结合语义槽值对的信息，形成了一种简单有效的语义表达方式——对话语义动作<sup>[3, 4]</sup>。这里的语义槽值对不仅可以呈现 slot = value 的形式，还可以出现 slot 名为空或 value 为空的情况，参见表格1-1。由于这种表示方式更侧重对话行为的表示上，更多应用于多轮对话的场景

表 1-1 对话语义动作示例

Table 1-1 Examples of Dialogue Act

从上海到北京的机票	inform(出发城市 = 上海, 到达城市 = 北京)
您是明天上午十点出发吗	confirm(出发日期 = 明天, 出发时间 = 上午十点)
现在去北京的机票价格是多少	request(机票价格, 到达城市 = 北京)
不是，你听错了	deny()
我无所谓	inform( =dontcare)

- 逻辑形式(Logical Form)，常常作为语义解析(Semantic Parsing)任务的中间语义表示。Lambda Dependency-Based Compositional Semantics(Lambda DCS)<sup>[5]</sup>是一种经典的逻辑形式，它从 $\lambda$ 演算(Lambda Calculus)发展而来，相比 $\lambda$ 演算表示起来更容易，常常被用来处理需要对问句进行推理和计算的解析问题。语义解析就是将自然语言转化为一种逻辑形式，涉及较多语言学(Linguistic)的先验知识，这里不展开赘述。自然语言转化为Lambda DCS逻辑形式的例子参见图1-4

<sup>1</sup>一种特殊的分类问题(序列标注)，slot filling 任务详见4.2.1.1

<sup>2</sup>有时也称为句子分类，sentence classification，详见4.2.1.2

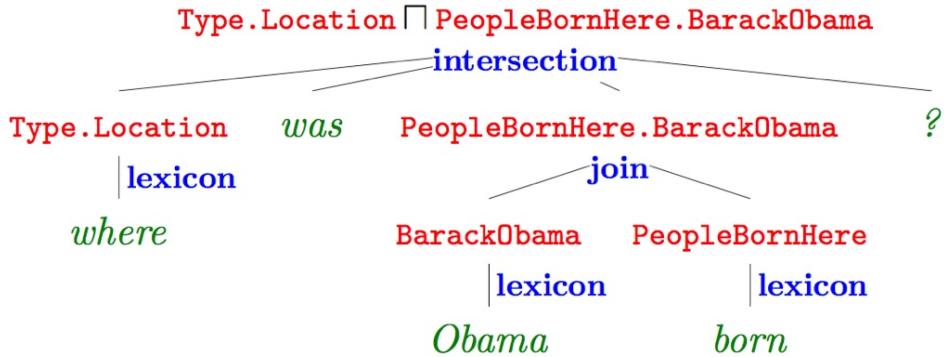


图 1-4 自然语言转化为 LambdaDCS 逻辑形式树状图

Figure 1-4 Transform Utterance to Logical Form–Lambda DCS

### 1.2.2.2 评价标准

对于一个语义理解模块的评价标准一般是 slot filling 和 intent classification 结果的准确率 (precision, P)、召回率 (recall, R)、F1(f-score)，有时还会直接使用整句解析的错误率或精度 (accuracy, acc) 作为参考。

**精度 acc** 对样例语句集记为  $D$ ,  $D$  的大小为  $m$ , 每一个样例参考的语义解析结果记为  $y^{(i)}$ (包括 slot 的识别和意图分类的结果), 由解析器解析的语义结果记为  $f(x^{(i)})$ ,  $i = 1, 2, \dots, m$ , 则整句解析的精度定义为

$$acc(f; D) = \frac{1}{m} \sum_{i=1}^m I(f(x^{(i)}) = y^{(i)})$$

**P-R-F1** 对于 slot filling 和 intent classification 这两个多分类的任务，针对每一个类别，根据真实类别与学习器预测的类别可以划分为真正例 (true positive, TP)、假正例 (false positive, FP)、真反例 (true negative, TN)、假反例 (false negative, FN) 四种情形，令 TP、FP、TN、FN 分别表示各情况对应的样例数，则  $TP + FP + TN + FN = m$  样本总数。分类结果的混淆矩阵 (confusion matrix) 参见表1-2

表 1-2 分类结果混淆矩阵

Table 1-2 Confusion Matrix

真实情况	预测结果	
	正例	反例
正例	TP(真正例)	FN(假反例)
反例	FP(假正例)	TN(真反例)

准确率  $P$ ，召回率  $R$  和  $F$ -score ( $F1$ ) 分别定义为

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN} \quad F1 = \frac{2 \times P \times R}{P + R}$$

准确率和召回率是一对矛盾的度量，当一方比较高时，另一方通常会比较低，为了综合考虑这两者，计算准确率和召回率的调和平均数就是 f-score。对于  $n$  个类别，先分别计算每个类别的准确率和召回率， $(P_1, R_1), (P_2, R_2), \dots, (P_n, R_n)$ ，再计算  $P$  和  $R$  的平均数，就得到多分类任务的宏准确率 (macro-P) 和宏召回率 (macro-R)，以及相应的宏 F1(macro-F1)<sup>1</sup>

$$\text{macro-P} = \frac{1}{n} \sum_{i=1}^n P_i \quad \text{macro-R} = \frac{1}{n} \sum_{i=1}^n R_i$$

$$\text{macro-F1} = \frac{2 \times \text{macro-P} \times \text{macro-R}}{\text{macro-P} + \text{macro-R}}$$

### 1.2.3 基于知识图谱的语义理解

目前学术界基于知识图谱的语义理解大致有三种思路：

- 基于语义解析的方法<sup>[7]</sup>，主要思想是将自然语言解析成具有一定逻辑形式的语义表达式，通过对逻辑形式自底向上的解析，通过与之匹配的知识库查询语句从预先构建的知识图谱中检索，从而得到答案。图1-5的例子<sup>[8]</sup> 中使用的逻辑形式为  $\lambda$  演算

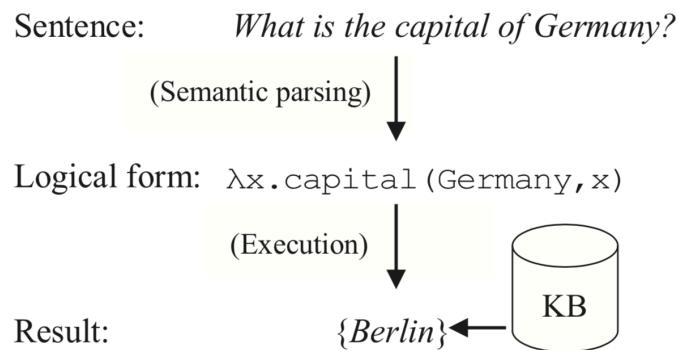


图 1-5 基于语义解析的知识图谱智能问答示意图

Figure 1-5 Procedure of Question Answering over Knowledge Graph based on Semantic Parsing

- 基于信息提取 (Information Extraction) 的方法<sup>[9]</sup>，该方法首先提取问题中的实体信息，在知识库中查询对应的实体，得到该实体的知识库子图，子图中的节点或边都可以作为候选答案，再针对整句的问题进行特征建模，得到特征向量作为输入，构建分类器从候选答案中筛选出最佳答案
- 基于向量建模 (Vector Space Modeling) 的方法<sup>[10]</sup>，该方法和基于信息提取的方法相近，区别在于它对所有候选答案也根据训练数据进行向量建模，得到问句和候选答案的分布式表达 (Distributed Embedding) 向量，通过比较问句和候选答案的相似度 (通过向量空间的距离度量方法，比如向量点乘的大小、余弦相似度)，筛选出最匹配的答案

后两种方法对于复杂、甚至需要进一步推理或计算的问句，表现效果比较差，而基于语义解析的方法得到的逻辑形式具有更加规范的层次性语义结构，可以实现更加复杂、智能的查询。随着深

<sup>1</sup>还有微准确率、微召回率、微 F1，详见 [6] 章节 2.3.2，实验中不采取此方法，这里不赘述

度学习的日益流行，深度神经网络在自然语言处理领域带来的巨大变革也为知识图谱问答中的语义理解任务带来了新的思路。本论文在这样的时代背景下进行研究，希望能够提高基于知识图谱的语义解析任务的准确率和鲁棒性，提供更加智能化的自动问答服务。

### 1.3 本课题的研究内容及贡献

本课题融合了知识图谱、自动问答系统和自然语言理解三个方向，主要研究在以知识图谱作为知识背景的支持，以智能客服问答为最终目的下，对用户提出的询问进行语义解析的多种思路及模型结构。在基于规则的解析和基于统计的解析两种方式的框架下，从语法规则的定义、解析模板的编写、训练数据的批量生成、测试数据的众包搜集，到基于有限状态转移机(Finite State Transducer, FST)的规则解析器(FSTParser)的实现、基于统计方法的神经网络诸多模型结构的改进，尝试了 Bi-RNN、Bi-RNN+CRF、Focus 多种神经网络模型，定性并且定量分析了两种方法及其下子模型的实验结果和原因。最后，我们展示了基于知识图谱的智能问答系统的成果，并为今后的工作指明了方向。

本文剩余部分的组织结构如下：第二章我们首先介绍本课题相关的背景知识，包括有限状态转移机、条件随机场和神经网络；第三章我们先引入一套自定义的、用于语义解析的语法规则以及用来查询知识库的中间语义表示的逻辑形式，接着介绍基于 FST 的规则解析方法和原理；第四章我们先介绍数据集的来源渠道，包括训练数据的获取和测试数据的收集等工作，然后逐层深入地介绍基于统计方法的实验中使用的网络模型和解码细节；第五章具体介绍各种模型的实验结果，并且详细地分析了产生这些结果的原因；第六章我们对目前的工作进行了总结、展示了基于我们语义解析工作的智能问答系统的成果并确定了未来工作的目标和方向。在附录中，我们先以一条简单的语句为例给出了基于 FST 规则解析的具体流程，然后附上 SpeechLab 领域本体结构的设计、自行编写的规则模板样例、发放的问卷样例以及收集到的部分测试集语句，最后列出了本论文相关的参考文献、谢辞和英文大摘要。

## 第二章 背景知识

### 2.1 有限状态转移机

有限状态转移机 (Finite State Transducer, 以下简称 FST)<sup>1</sup>，又称有限状态机 (Finite State Machine, FSM)、有限状态自动机、状态机，是表示有限个状态并能够在这些状态之间转移和动作的一种数学模型。通俗的描述 FST 就是定义一个状态的集合以及在这些状态之间转换的规则，当满足某些转移规则时就会执行相应的动作、跳转到另一个状态或两者兼而有之。

#### 2.1.1 数学模型

FST 可以粗略地分为接收器 (识别器) 和变换器两种类型。接收器或识别器，也叫序列检测器，产生一个二元输出，相当于回答某个输入是否被机器接受，它存在一个最终/接受状态，当输入输完后，如果机器不处于最终/接受状态，输出 False，否则输出 True。而变换器会根据输入和状态的变换给出一个输出序列，通过一个六元组  $(\Sigma, \Gamma, S, s_0, \delta, \omega)$  来定义：

- $\Sigma$  是非空输入字符表，一般用字符  $\varepsilon$  作为无条件转移的符号 (不需要任何输入就能实现状态之间的转移)
- $\Gamma$  是非空输出字符表，同样一般定义  $\varepsilon$  作为没有任何输出的标志
- $S$  是所有状态的非空有限集合
- $s_0$  是初始状态或初始状态的集合
- $\delta$  是状态转移函数  $\delta : S \times \Sigma \rightarrow S$
- $\omega$  是输出函数  $\omega : S \times \Sigma \rightarrow \Gamma$

序列检测器没有输出字符表和输出函数的定义，但比变换器多了一个  $F$ ——最终状态的集合。通常我们会把这两种类型结合起来，使之既能识别序列，又能在识别的同时给出一个输出序列，即为变换器模型也添加一个最终状态的集合，只有检测出序列时才给一个输出。图2-1给出了一个 FST 的例子，图中的圆代表状态，0 是初始状态，3 是最终或接受状态，边代表转移条件，边上的符号  $a : A$  表示在当前状态输入字符  $a$  时，FST 会输出字符  $A$ 。给定一个输入序列  $x$ ，从一个 FST 的初始状态出发，最终在接受状态停止时所有的输出字符拼接成的字符串称为输出序列  $y$ 。一个完整的 FST 甚至会给转移条件 (边) 和接受状态权重，当输入序列匹配上 FST 多个不同的输出序列时根据所有转移边和最终状态权重的和或乘积，从输出序列中进行筛选。带有权重的 FST 参见图2-2，/后的数字代表权重。

#### 2.1.2 有限状态转移机基本操作

关于有限状态机的开源库有很多，比如 C++ 版本的 Openfst 以及 Python 封装版本 Pyfst，这些第三方库都定义了有限状态机上的一些常用操作。

<sup>1</sup>严格来说，有限状态转移机是有限状态机的一种特殊子类，这里不做区分

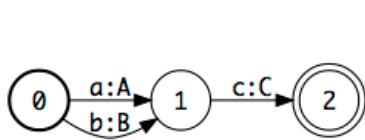


图 2-1 有限状态转移机示意图

Figure 2-1 Finite State Transducer

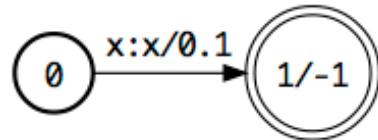


图 2-2 带权重的有限状态转移机示意图

Figure 2-2 Weighted Finite State Transducer

**拼接 concatenate** 假设 FST  $t_1$  接受输入  $xyz$  并输出  $XYZ$ , FST  $t_2$  接受输入  $uvw$  并输出  $UVW$ , 那么将这两个 FST 拼接得到新的 FST  $t = t_1 + t_2$ , 接受输入  $xyzuvw$  并输出  $XYZUVW$

**合并 merge** 二元操作, 假设 FST  $t_1$  只能接受输入  $xyz$ , FST  $t_2$  只能接受输入  $uvw$ , 那么将这两个 FST 合并得到新的 FST 既能接受  $xyx$  也能接受  $uvw$

**组合 compose** 二元操作, 假设 FST  $t_1$  接收输入  $xyz$  并输出  $abc$ , FST  $t_2$  接受输入  $abc$  并输出  $xyz$ , 那么将这两个 FST 组合得到新的 FST  $t = t_1 >> t_2$ , 接受输入  $xyz$  并输出  $xyz$

**反转 inverse** 单元操作, 假设 FST  $t$  接收输入  $xyz$  并输出  $abc$ , 那么反转  $t$  得到新的 FST  $t_{inversed}$  接受输入  $abc$  并输出  $xyz$

**确定化 determinize** 单元操作, 确定化一个 FST 得到的 FST 和原来的 FST 等价, 但从新的 FST 任何一个状态出发, 没有两条输入相同的转移边

**其他操作** 其他操作包括最小化状态数 (minimize, 单元操作得到一个等价的 FST 但在所有等价的 FST 中状态数目是最少的)、删除无条件转移边 (remove\_eps, 单元操作得到一个等价的 FST 但不包含任何无条件转移的边) 等

## 2.2 条件随机场

### 2.2.1 概率无向图模型

首先我们定义概率无向图模型 (probabilistic undirected graphical model), 又称为马尔可夫随机场, 一种通过无向图描述随机变量之间依赖关系, 从而定义它们的联合概率分布的模型。概率无向图中的节点表示随机变量, 边表示两个随机变量之间的依赖关系。

任何一个概率无向图都满足以下三个性质 (完全等价):

**成对马尔可夫性** 假设  $u$  和  $v$  是概率无向图中的两个没有边直接相连的节点, 它们分别表示随机变量  $X_u$  和  $X_v$ , 其他所有节点构成的集合用  $O$  表示, 它们对应的随机变量集合为  $X_O$ , 那么在给定  $X_O$  的条件下  $X_u$  和  $X_v$  条件独立, 记为

$$P(X_u, X_v | X_O) = P(X_u | X_O)P(X_v | X_O)$$

**局部马尔可夫性** 假设  $u$  是概率无向图中任何一个节点, 代表随机变量  $X_u$ ,  $N$  是所有和节点  $u$  有边直接相连的节点集合, 代表随机变量集合  $X_N$ ,  $O$  是除了  $u$  和  $N$  之外的所有节点构成的集合, 代表随机变量集合  $X_O$ , 实际上  $u$  和  $X_O$  中的任何一个节点都满足成对马尔科夫性, 那么在给定随机变量集合  $X_N$  的条件下  $X_u$  和  $X_O$  条件独立, 记为

$$P(X_u, X_O | X_N) = P(X_u | X_N)P(X_O | X_N)$$

**全局马尔可夫性** 将局部马尔可夫性中的节点  $u$  扩展为节点集合  $U$ , 我们就得到了全局马尔可夫性: 假设节点集合  $U$  和节点集合  $O$  是被节点集合  $N$  分开的任意两个集合(不经过  $N$  中的节点,  $U$  中任意节点都不能和  $O$  中节点相连, 反之亦然), 它们对应的随机变量集合分别为  $X_U$ 、 $X_O$ 、 $X_N$ , 那么给定随机变量  $X_N$  的条件下  $X_U$  和  $X_O$  条件独立, 记为

$$P(X_U, X_O | X_N) = P(X_U | X_N)P(X_O | X_N)$$

为了求一个概率无向图表示的随机变量的联合概率分布, 可以依据节点集合之间的条件独立性将联合概率分布因子分解为若干联合概率分布的乘积。

**定义 2.1.** 无向图中任何两个节点都有边直接相连的节点集合称为团 (clique), 若  $C$  是无向图中一个团, 且加入任何一个新的节点都会使  $C$  不再是团, 则称  $C$  为极大团 (maximal clique)。

根据定义2.1, 概率无向图的随机变量联合概率分布可以表示为定义在所有极大团  $C$  上的函数  $\Psi_C(X_C)$  的归一化乘积。其中  $Z$  是规范化因子 (normalization factor),  $\Psi_C(X_C)$  称为势函数, 要求严格大于 0, 因此常呈现指数函数的形式。

$$\begin{aligned} P(X) &= \frac{1}{Z} \prod_C \Psi_C(X_C) \\ Z &= \sum_X \prod_C \Psi_C(X_C) \\ \Psi_C(X_C) &= e^{\varphi(X_C)} \end{aligned} \tag{2-1}$$

## 2.2.2 线性链条件随机场

条件随机场 (Conditional Random Field, CRF) 是在给定一组输入随机变量  $X$  的条件下, 另一组输出随机变量  $Y$  的条件概率分布模型, 它作出的一个重要假设是输出随机变量  $Y$  在给定输入变量  $X$  的条件下构成一个马尔科夫随机场。本课题关注的一个重要任务是 slot filling 这种序列标注问题, 这里只讲述线性链 (linear chain) 条件随机场 (图2-3)。

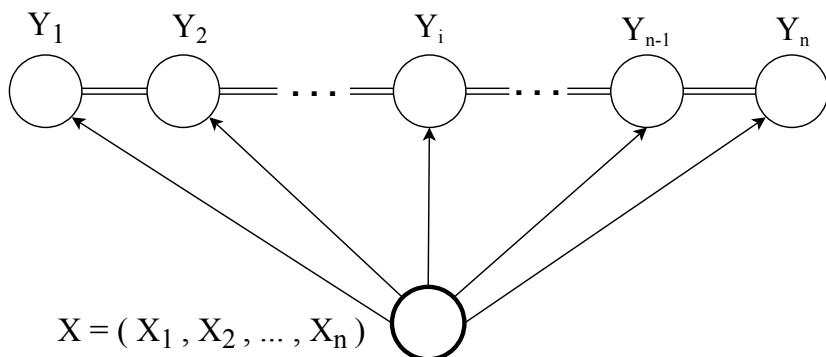


图 2-3 线性链条件随机场

Figure 2-3 Linear Chain Conditional Random Field

在条件概率模型  $P(Y|X)$  中  $Y = (Y_1, Y_2, \dots, Y_n)$  是输出变量, 表示标记的序列,  $X = (X_1, X_2, \dots, X_n)$  是输入变量, 表示需要标注的观测序列。给定输入变量  $X$ , 根据线性链条件随机场的局部马尔可夫

性，可以得到任意位置  $i$  输出标记的概率分布：(在  $i = 1$  和  $n$  时只取一条边)

$$\begin{aligned} \frac{P(Y_1, \dots, Y_{i-2}, Y_i, Y_{i+2}, \dots, Y_n | X, Y_{i-1}, Y_{i+1})}{P(Y_1, \dots, Y_{i-2}, Y_{i+2}, \dots, Y_n | X, Y_{i-1}, Y_{i+1})} &= P(Y_i | X, Y_{i-1}, Y_{i+1}) \\ P(Y_i | X, Y_1, \dots, Y_{i-1}, Y_{i+1}, \dots, Y_n) &= P(Y_i | X, Y_{i-1}, Y_{i+1}) \quad (i = 1, 2, \dots, n) \end{aligned}$$

如果具体的输入序列为  $\mathbf{x} = (x_1, \dots, x_n)$ ，则根据公式2-1，输出序列为  $\mathbf{y} = (y_1 \dots, y_n)$  的概率为

$$\begin{aligned} p(\mathbf{y} | \mathbf{x}) &= \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{i,k} \lambda_k t_k(y_{i-1}, y_i, \mathbf{x}, i) + \sum_{i,l} \beta_l s_l(y_i, \mathbf{x}, i)\right) \\ Z(\mathbf{x}) &= \sum_y \exp\left(\sum_{i,k} \lambda_k t_k(y_{i-1}, y_i, \mathbf{x}, i) + \sum_{i,l} \beta_l s_l(y_i, \mathbf{x}, i)\right) \end{aligned} \quad (2-2)$$

其中  $t_k(\cdot)$  和  $s_l(\cdot)$  分别表示转移特征函数和状态特征函数，需要遍历输出序列的每一个位置，只有当输出和输入满足一定条件时特征函数为 1，否则为 0。 $\lambda_k$  和  $\beta_l$  分别是对应的权重，也是我们需要学习的参数， $Z(\mathbf{x})$  是规范化因子。条件随机场分为概率计算和预测最大概率的标记序列两个重要任务。

### 2.2.3 概率计算

为了方便推导，我们将公式2-2中转移特征、状态特征及其相应的权重系数采用统一的符号改写。

假设有  $K_1$  个转移特征和  $K_2$  个状态特征， $K = K_1 + K_2$ ，它们的系数都用  $\omega_i$  表示， $i = 1, 2, \dots, K$ ，公式2-2可以改写为：

$$\begin{aligned} f_k(y_{i-1}, y_i, \mathbf{x}, i) &= \begin{cases} t_k(y_{i-1}, y_i, \mathbf{x}, i) & k = 1, 2, \dots, K_1 \\ s_l(y_i, \mathbf{x}, i) & l = K_1 + 1, \dots, K \end{cases} \\ \omega_k &= \begin{cases} \lambda_k & k = 1, 2, \dots, K_1 \\ \beta_l & l = K_1 + 1, \dots, K \end{cases} \\ P(\mathbf{y} | \mathbf{x}) &= \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{i=1}^{n+1} \sum_{k=1}^K \omega_k f_k(y_{i-1}, y_i, \mathbf{x}, i)\right) \\ &= \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{i=1}^{n+1} \vec{\omega} \cdot \vec{f}(y_{i-1}, y_i, \mathbf{x}, i)\right) \\ &= \frac{1}{Z(\mathbf{x})} \exp(\vec{\omega} \cdot \vec{f}(\mathbf{y}, \mathbf{x})) \\ Z(\mathbf{x}) &= \sum_y \exp\left(\sum_{i=1}^{n+1} \sum_{k=1}^K \omega_k f_k(y_{i-1}, y_i, \mathbf{x}, i)\right) \\ &= \sum_y \exp\left(\sum_{i=1}^{n+1} \vec{\omega} \cdot \vec{f}(y_{i-1}, y_i, \mathbf{x}, i)\right) \\ &= \sum_y \exp(\vec{\omega} \cdot \vec{f}(\mathbf{y}, \mathbf{x})) \end{aligned}$$

其中

$$\vec{w} = (w_1, w_2, \dots, w_K)^T$$

$$\vec{f}(y_{i-1}, y_i, \mathbf{x}, i) = (f_1(y_{i-1}, y_i, \mathbf{x}, i), f_2(y_{i-1}, y_i, \mathbf{x}, i), \dots, f_K(y_{i-1}, y_i, \mathbf{x}, i))^T$$

$$\vec{f}(\mathbf{y}, \mathbf{x}) = \sum_{i=1}^{n+1} \vec{f}(y_{i-1}, y_i, \mathbf{x}, i)$$

假定输入序列的长度是  $n$ , 对  $y$  引入两个特殊标记  $start$  和  $end$  后  $y$  的标记数目为  $m$ , 并令  $y_0 = start$ ,  $y_{n+1} = end$ , 对于序列的每一个位置  $i$ , 对  $K$  个特征函数求和, 可以得到一个  $m$  阶的矩阵 ( $y_{i-1}$  和  $y_i$  的取值个数都是  $m$  个, 所以是  $m$  阶)

$$M_i(\mathbf{x}) = [ M_i(y_{i-1}, y_i | \mathbf{x}) ]_{m \times m}$$

$$M_i(y_{i-1}, y_i | \mathbf{x}) = \exp\left(\sum_{k=1}^K \omega_k f_k(y_{i-1}, y_i, \mathbf{x}, i)\right)$$

$$= \exp(\vec{\omega} \cdot \vec{f}(y_{i-1}, y_i, \mathbf{x}, i))$$

其中对于序列位置  $i$  的矩阵  $M_i$ , 第  $\alpha$  行第  $\beta$  列的值可以理解为当前位置  $i$  的标记为  $\beta$  且前一个位置  $i-1$  的标记为  $\alpha$  的非规范化概率, 据此, 我们可以通过矩阵对应位置元素的乘积  $\prod_{i=1}^{n+1} M_i(y_{i-1}, y_i | \mathbf{x})$  得到整个给定的输出序列  $\mathbf{y}$  的非规范化概率, 而规范化因子  $Z(\mathbf{x})$  可以通过矩阵乘积  $\prod_{i=1}^{n+1} M_i(\mathbf{x})$  的第 ( $start$ ,  $end$ ) 位置的值得到。在实际计算  $Z(\mathbf{x})$  中, 先求出每个位置  $i$  的矩阵  $M_i$  并计算它们乘积的开销比较大, 常常采用动态规划思想, 定义一个  $m$  维的前向向量  $\vec{a}_i(y_i | \mathbf{x})$  ( $y_i$  的取值有  $m$  个, 所以维度是  $m$ ), 它的第  $j$  个元素表示从前向后一直统计到位置  $i$  并且位置  $i$  处的标记  $y_i$  为  $j$  的非规范化概率, 根据递推公式2-3求出  $Z(\mathbf{x})$

$$\vec{a}_0(y_0 | \mathbf{x}) = \begin{cases} 1 & y_0 = start \\ 0 & otherwise \end{cases}$$

$$\vec{a}_i^T(y_i | \mathbf{x}) = \vec{a}_{i-1}^T(y_{i-1} | \mathbf{x}) M_i(\mathbf{x})$$

$$Z(\mathbf{x}) = \vec{a}_n^T(y_n | \mathbf{x}) \cdot \vec{1} \quad (2-3)$$

有了非规范化概率  $\prod_{i=1}^{n+1} M_i(y_{i-1}, y_i | \mathbf{x})$  和规范化因子  $Z(\mathbf{x})$ , 就可以求得给定输入序列  $\mathbf{x}$  输出序列为  $\mathbf{y}$  的条件概率

$$P(\mathbf{y} | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{i=1}^{n+1} M_i(y_{i-1}, y_i | \mathbf{x})$$

## 2.2.4 预测标记序列

给定一个条件随机场 (参数已知), 根据输入序列  $\mathbf{x}$  求条件概率最大的输出标记序列一般采用动态规划思想的维特比算法 (Viterbi Algorithm)。

$$\begin{aligned}\mathbf{y}^* &= \arg \max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \\ &= \arg \max_{\mathbf{y}} \frac{\exp(\sum_{i=1}^n \vec{w} \cdot \vec{f}(y_{i-1}, y_i, \mathbf{x}, i))}{Z(\mathbf{x})} \\ &= \arg \max_{\mathbf{y}} \sum_{i=1}^n \vec{w} \cdot \vec{f}(y_{i-1}, y_i, \mathbf{x}, i)\end{aligned}$$

定义维特比向量  $\vec{v}_i$  为输出序列在位置  $i$  处各个标记上的非规范化概率, 第  $j$  个维度 ( $j = 1, 2, \dots, m$ ) 的值用  $v_i(j)$  表示, 回溯向量  $\vec{k}_i$  用来追溯非规范化概率最大的标记路径, 用  $\kappa_i(j)$  记录输出序列在位置  $i$  处标记为  $y_i = j$  并使之非规范化概率最大时  $y_{i-1}$  的标记, 具体流程参见算法2-1

---

### 算法 2-1 条件随机场的维特比算法

---

**输入:** 经过训练学得的权重向量  $\vec{w} = (w_1, w_2, \dots, w_K)^T$ ; 标记集合大小  $m$ ; 观测或输入序列  $\mathbf{x} = (x_1, x_2, \dots, x_n)$ ; 模型特征函数向量  $\vec{f}(y_{i-1}, y_i, \mathbf{x}, i) = (\vec{f}_1(y_{i-1}, y_i, \mathbf{x}, i), \dots, \vec{f}_K(y_{i-1}, y_i, \mathbf{x}, i))^T$ ;

**输出:** 条件概率最大的输出路径  $\mathbf{y}^* = (y_1^*, y_2^*, \dots, y_n^*)$ ; 非规范化的最大条件概率  $maxProb$ ;

```

1:  $v_1(j) = \vec{w} \cdot vecf(y_0 = start, y_1 = j, \mathbf{x}, 1) \quad j = 1, 2, \dots, m$ 
2: for  $i = 2 \rightarrow n$  do
3:    $v_i(l) = \max_{1 \leq j \leq m} \{ v_{i-1}(j) + \vec{w} \cdot \vec{f}(y_{i-1} = j, y_i = l, \mathbf{x}, i) \} \quad l = 1, 2, \dots, m$ 
4:    $\kappa_i(l) = \arg \max_{1 \leq j \leq m} \{ v_{i-1}(j) + \vec{w} \cdot \vec{f}(y_{i-1} = j, y_i = l, \mathbf{x}, i) \} \quad l = 1, 2, \dots, m$ 
5: end for
6:  $maxProb = \max_{1 \leq j \leq m} v_n(j)$ 
7:  $y_n^* = \arg \max_{1 \leq j \leq m} v_n(j)$ 
8: for  $i = n - 1 \rightarrow 1$  do
9:    $y_i^* = \kappa_{i+1}(y_{i+1}^*)$ 
10: end for
11: return  $\mathbf{y}^* = (y_1^*, y_2^*, \dots, y_n^*), maxProb$ 
```

---

## 2.3 神经网络

### 2.3.1 神经元和前馈网络

神经网络的基本组成单位是神经元 (neuron), 它接受一个多维的输入向量, 经过线性变换将其映射成一个实数, 该实数再通过一个激活函数 (activation function) 得到输出结果。一个神经元可以理解为从向量到标量的函数, 示意图参见2-4。

神经网络每一层由多个神经元并行构成, 它们接受原始输入或网络上一层的输出向量作为这一层的输入, 并行计算后将它们的结果拼接起来一起构成下一层的输入向量。一个典型的前馈神经网

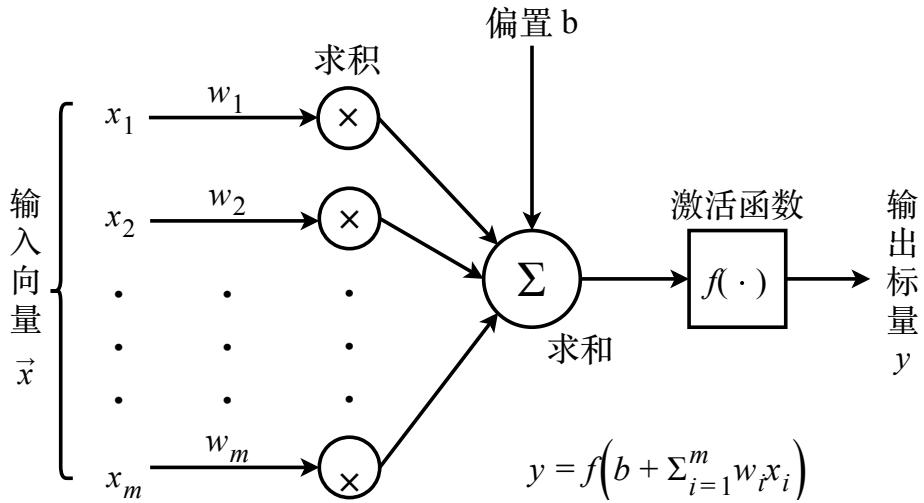


图 2-4 神经元模型

Figure 2-4 A Neuron Model

**络 (Feedforward Neural Network)** 由输入层 (input layer)、隐藏层 (hidden layer) 和输出层 (output layer) 三个部分构成。输入层接受最原始的数据样本构成的向量，隐藏层的层数决定了网络的深度，而每一个隐层神经元的数目称为隐藏层大小 (hidden size)，它决定了网络模型的宽度。神经网络的非线性功能由激活函数实现，常用的激活函数有 *sigmoid*, *tanh*, *relu*, *softplus*。不同的激活函数在不同深度学习的任务表现上各有差异，无法评定谁好谁坏，实验中应多次尝试选出最适合本任务的非线性函数。

$$\begin{aligned} \text{sigmoid}(x) &= \frac{1}{1 + e^{-x}} \\ \text{softplus}(x) &= \log(1 + e^x) \end{aligned}$$

$$\begin{aligned} \tanh(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \\ \text{relu}(x) &= \max(0, x) \end{aligned}$$

一个典型的前馈神经网络参考图2-5

1. 输入层的向量来自于原始数据的变换 (非数值转化为数值向量)，不同任务前期处理过程大相径庭。以自然语言处理任务为例，需要预先构建一个词表映射，把每一个符号 (单词或特殊字符) 映射到一个长整形的整数，具体的处理方式参见4.2.3.2
2. 理论上，即使只有一层隐藏层，只要隐层数目足够多，该前馈网络可以实现几乎任意的目标函数。但加深网络的层数可以避免很多冗余的计算，使得网络具有更多的非线性，在参数数目相同的情况下，相比单层网络它可以实现更多更复杂的函数。隐层之间的神经元连接方式有很多，全连接、局部连接 (卷积神经网络，CNN)、循环连接 (循环神经网络，RNN) 等，不同的连接方式决定了网络的结构
3. 输出层获取网络的输出用于解决具体的任务，常见的任务主要是回归 (regression) 和分类 (classification) 两类。对于回归任务，需要根据输入预测一个具体的数值，一般采取基于仿射变换的线性输出单元， $\vec{y} = W^T \vec{h} + \vec{b}$ ， $\vec{h}$  是紧连输出层的隐层的输出向量。这种线性输出层的结果常被用来产生高斯条件分布的均值，这时最大化对数似然等价于最小化均方误差。而对于分类任务，输出层在线性输出单元的基础上会再经过一个 softmax 层，得到在  $n$  个类别上的概

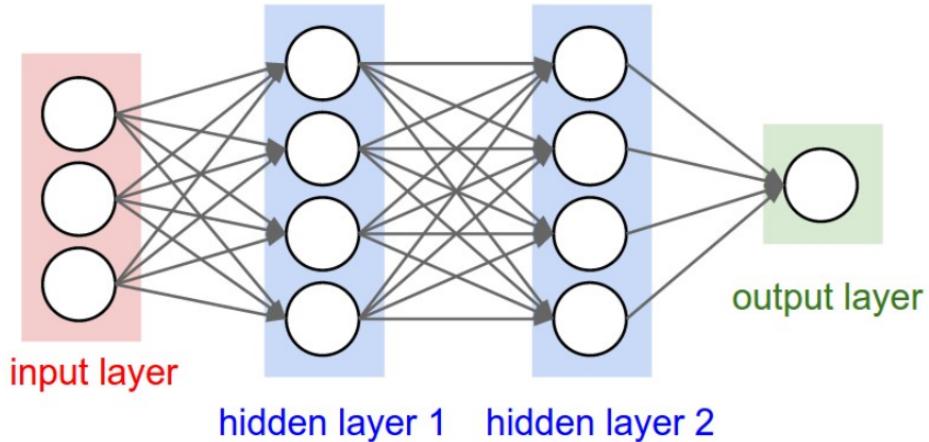


图 2-5 前馈神经网络示意图

Figure 2-5 Example of A FeedForward Neural Network

率分布 (多项分布, Multinoulli Distribution)

$$\text{softmax}(\vec{y})_i = \frac{\exp(y_i)}{\sum_j \exp(y_j)}$$

关于神经网络的计算 (前向传播) 和训练 (误差反向传播算法)、损失函数 (极大似然估计、交叉熵)、正则化措施 (dropout、早停) 以及优化器 (随机梯度下降, SGD) 等问题参见 [11]，本论文主要关注网络的结构。

### 2.3.2 循环神经网络及其变体

循环神经网络 (Recurrent Neural Network, RNN) 是一种用于处理序列化数据的网络模型，自然语言的一句话就是一种典型的序列。序列化的输入存在时间上的依赖关系，一个后面出现的概念或词汇很可能依赖之前或之后出现的一个概念或词汇。RNN 与前馈神经网络最大的区别在于隐藏层神经元对循环连接的引入，循环连接会将前一时刻神经元的输出作为下一时刻该神经元输入的一部分。因此，隐藏层神经元的输入不仅取决于当前的输入，还会受到前一时刻自身输出的影响。数学形式地描述为：

$$h^{(t)} = \tanh(W h^{(t-1)} + U x^{(t)} + b)$$

$h^{(t)}$  为  $t$  时刻神经元的输出或称之为隐状态 (hidden state)， $x^{(t)}$  为  $t$  时刻的输入

#### 2.3.2.1 长短期记忆单元

传统循环神经网络的权重是固定的，存在长期依赖的问题：经过时间序列上长时间的传递后，很久之前的输入对当前的影响甚微，丢失重要的特征，学习训练时出现梯度爆炸或消失的情况。门控类 RNN(即将要讨论的 LSTM 和 GRU) 作出的重大改变在于将循环连接的权重变成上下文相关，每个时间步的自连接权重都可能改变，不再是固定不变的参数。

长短期记忆单元 (Long Short-Term Memory, LSTM) 对传统神经元进行修改，引入了控制阀门的概念并对输入特征和中间状态的信息作出变换，示意图参见2–6

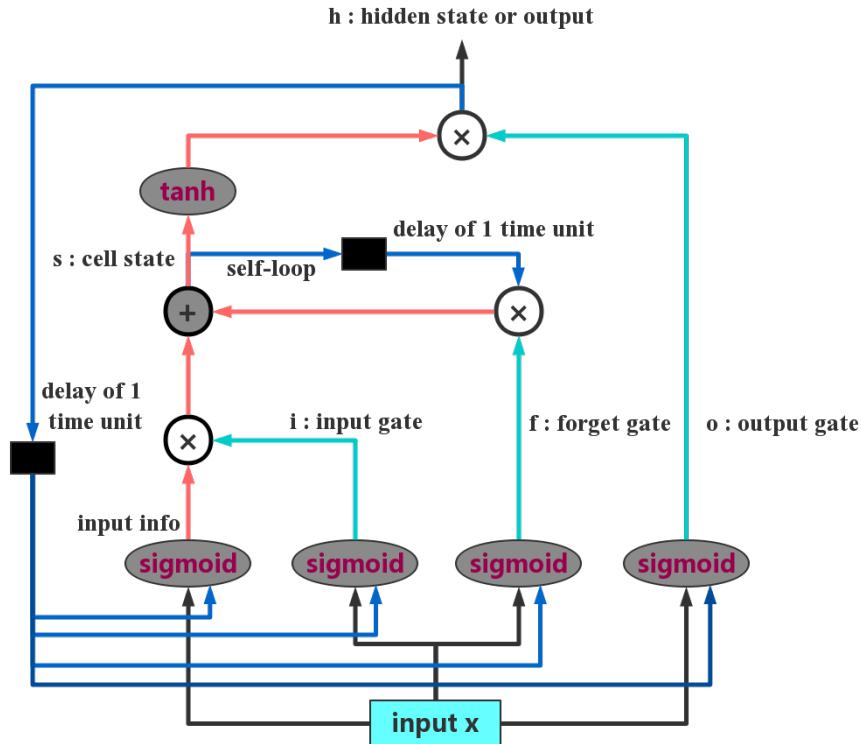


图 2–6 长短期记忆单元示意图  
Figure 2–6 Long Short-Term Memory Unit

- LSTM 引入了三个控制阀门 (gate)，输入门 (input gate,  $i$ )、遗忘门 (forget gate,  $f$ ) 和输出门 (output gate,  $o$ )，分别控制输入特征的更新，历史信息的保留和最终结果的输出，它们具有相同的结构和类似的计算公式 (以下均以时间  $t$  为例)

$$\begin{aligned} i^{(t)} &= \sigma(b_i + U_i x^{(t)} + V_i h^{(t-1)}) \\ f^{(t)} &= \sigma(b_f + U_f x^{(t)} + V_f h^{(t-1)}) \\ o^{(t)} &= \sigma(b_o + U_o x^{(t)} + V_o h^{(t-1)}) \end{aligned}$$

- 对于输入特征  $x^{(t)}$ ，我们也同样做出变换，得到变换后的  $input info$

$$input info = \sigma(b + U x^{(t)} + V h^{(t-1)})$$

- 对于状态单元 (cell state)  $s^{(t)}$ ，由历史信息 ( $s^{(t-1)}$ ) 和变换后的新信息 ( $input info$ ) 组成，并分别

受到遗忘门 (forget gate) 和输入门 (input gate) 的控制

$$\begin{aligned}s^{(t)} &= f^{(t)} s^{(t-1)} + i^{(t)} (\text{input info}) \\ &= f^{(t)} s^{(t-1)} + i^{(t)} \sigma(b + Ux^{(t)} + Vh^{(t-1)})\end{aligned}$$

- $t$  时刻的输出或隐状态  $h^{(t)}$  由状态单元经过非线性变换并受输出门的控制得到

$$h^{(t)} = \tanh(s^{(t)})o^{(t)}$$

### 2.3.2.2 门控单元变体

门控循环单元 (Gated Recurrent Unit, GRU)<sup>[12]</sup> 是一种简化版的 LSTM 单元，将原来的三个控制阀门减少到两个，示意图参见 2-7

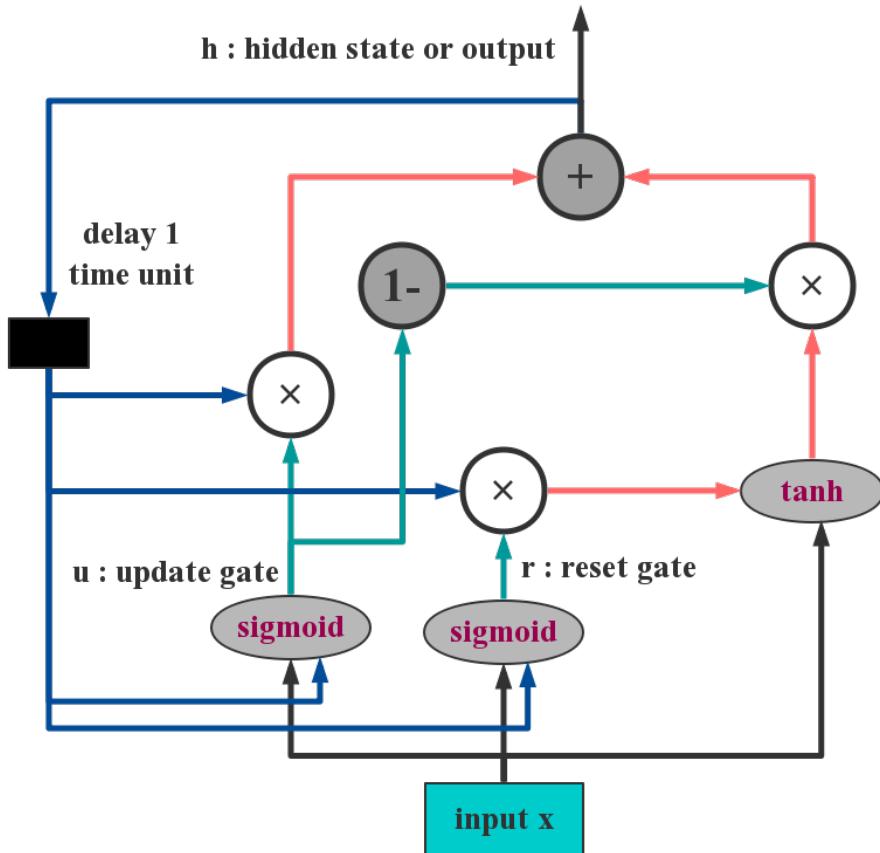


图 2-7 循环门控单元示意图

Figure 2-7 Gated Recurrent Unit

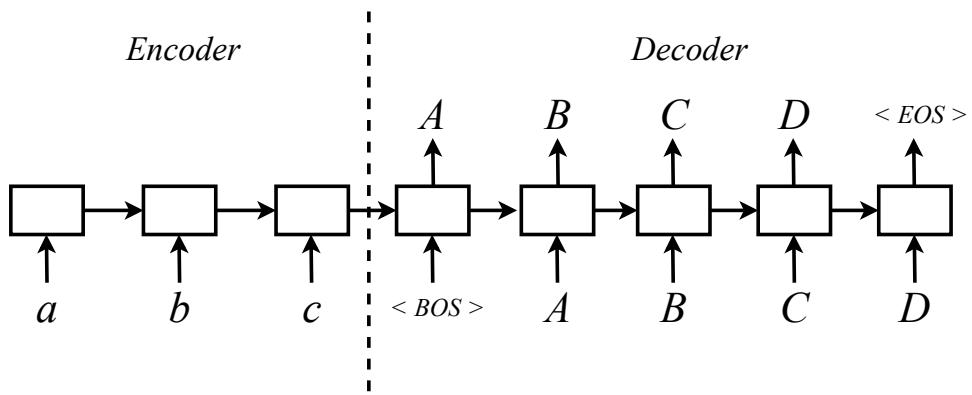
- GRU 不记录状态单元，只记录隐状态或输出，而 LSTM 中每一时刻的隐状态都由该时刻的状态单元和输出控制门确定

- 只有两个阀门 (删除了输出控制的阀门), 一个更新门 (update gate,  $u$ ), 它同时控制输入信息的更新和历史信息的保留, 另一个重置门 (reset gate,  $r$ ), 辅助控制历史信息在输入特征的变换中起到的作用, 计算公式

$$\begin{aligned} u^{(t)} &= \sigma(W_u x^{(t)} + V_u h^{(t-1)} + b_u) \\ r^{(t)} &= \sigma(W_r x^{(t)} + V_r h^{(t-1)} + b_r) \\ \hat{h}^{(t)} &= \tanh(W x^{(t)} + V(r^{(t)} h^{(t-1)}) + b) \\ h^{(t)} &= u^{(t)} h^{(t-1)} + (1 - u^{(t)}) \hat{h}^{(t)} \end{aligned}$$

### 2.3.3 Sequence-to-Sequence 模型及其变体

序列到序列 (Sequence-to-Sequence, Seq2Seq) 模型, 又称编码-解码 (Encoder-Decoder) 模型, 2014 年最早在 [13] 和 [12] 中分别独立被提出。最早是被用来解决机器翻译的问题, 超越了当时最好的基于短语翻译的系统, 后来被广泛应用于输入序列和目标输出序列长度不相等的任务, 诸如语音识别、文本摘要、对话生成等从一个模态转换成另一个模态的系统。Seq2Seq 的核心思想在于将整个网络架构分成两个部分, 一个编码器 (Encoder) 和一个解码器 (Decoder), 编码器将原始序列输入特征编码成一个固定长度的中间语义表示向量<sup>1</sup>  $c$ , 它包含汇总了整个序列重要的信息, 接下来的解码器接受这个语义表示向量  $c$ , 按照时间序列逐个生成目标序列, 前一时刻的输出作为下一个时刻的输入。训练的目标在于最大化目标序列的条件概率  $p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^n p_\theta(y_i|y_1, \dots, y_{i-1}, \mathbf{x})$ 。图示参见 2-8。



### 2.3.4 Attention 模型

传统的 Seq2Seq 模型无论输入多长，都把它编码成一个固定长度的向量，且一旦编码成语义向量  $c$ ，编码器部分的特征信息就不再使用。编码器和解码器之间的连接仅仅通过一个固定长度的中间语义向量，一旦遇到较长的输入，固定长度的语义向量  $c$  无法包含所有必要的信息，必然存在着信息的缺失。注意力模型 (Attention Model, AM)<sup>[14]</sup> 的核心思想在于解码器使用与当前时刻相关的中间语义表示  $c^{(t)}$ ，从而让中间语义  $\mathbf{C} = (c^{(1)}, c^{(2)}, \dots, c^{(t)})$  成为了一个可变长度的序列，同时引入了序列  $\mathbf{C}$  的元素和编码器输出结果相关联的注意力机制。<sup>[11]</sup> 将编码-解码过程细分为以下三步：

- 编码器读取原始输入，并将其转换为分布式表示  $(h_{encoder}^{(1)}, h_{encoder}^{(2)}, \dots, h_{encoder}^{(t)})$ ，其中每一个特征向量  $h_{encoder}^{(i)}$  分别与每个词相关联
- 一个存储器保存所有和输入词对齐的特征向量列表  $(h_{encoder}^{(1)}, h_{encoder}^{(2)}, \dots, h_{encoder}^{(t)})$ ，以供解码器从中检索使用
- 解码器顺序执行输出任务，每个时间步聚焦于存储器不同的元素的内容 (具有不同权重)

记  $t$  时刻编码器隐状态或输出为  $h_{encoder}^{(t)}$ ， $t$  时刻解码器输入标记为  $y^{(t-1)}$ 、输入语义向量为  $c^{(t)}$ ，前一时刻的解码器隐状态为  $s_{decoder}^{(t-1)}$ ， $[a; b]$  表示向量拼接操作， $\alpha_{tj}$  是权重系数， $T_x$  表示输入序列的长度， $f(\cdot)$  表示解码器部分的 LSTM/GRU 网络，则 AM 模型解码器在  $t$  时刻隐状态  $s_{decoder}^{(t)}$  的计算公式为：

$$\begin{aligned}s_{decoder}^{(t)} &= f(s_{decoder}^{(t-1)}, [c^{(t)}; y^{(t-1)}]) \\ c^{(t)} &= \sum_{j=1}^{T_x} \alpha_{tj} h_{encoder}^{(j)} \\ \alpha_{tj} &= \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})} \\ e_{tj} &= align(s_{decoder}^{(t-1)}, h_{encoder}^{(j)})\end{aligned}$$

$align(\cdot)$  是对齐模型 (align model)，模型的输入是编码器  $j$  时刻的输出隐状态和解码器  $t-1$  时刻的输出隐状态，输出是一个标量，衡量解码器即将要计算的第  $t$  个输出隐状态和编码器第  $j$  个输出隐状态之间的相关性，作为计算解码器  $t$  时刻输入中间语义向量  $c^{(t)}$  的加权系数。对齐模型的实现方法有很多，具体讨论参见4.2.6。<sup>[14]</sup> 给出了 AM 模型的示意图2-9

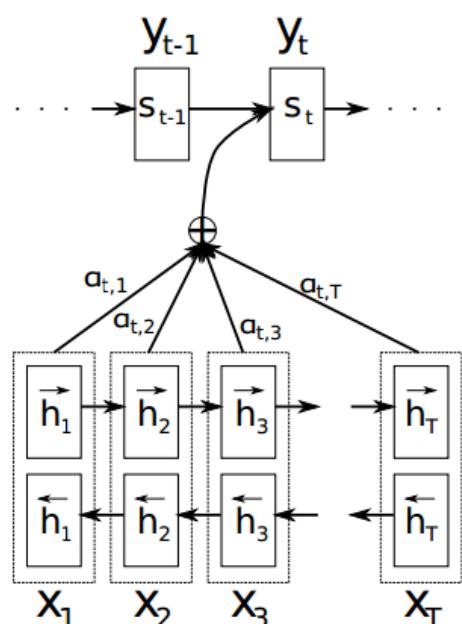


图 2–9 注意力模型示意图

Figure 2–9 Attention Model

## 第三章 语法规则定义及规则解析

本章先介绍基于知识图谱的语法解析的规则定义和用于知识库查询的逻辑形式，然后给出基于FST的规则解析的原理。

### 3.1 语法规则定义

#### 3.1.1 任务分类

一个领域相关的基于知识图谱的问答任务，根据常见的询问目标可以细分为以下几种：

1. 已知实体  $e$ ，询问其某个属性  $A$ ，符号表示为  $eA?$ ，比如张三今年多大了，实体为张三，询问属性年龄
2. 已知实体  $a$ ，询问其某个关系  $R$  的对应实体有哪些，符号表示为  $aR?$ ，比如李四发表了什么论文，实体为人李四，询问关系发表文献
3. 已知实体  $a$ ，询问其某个关系  $R$  的对应实体，且这些对应实体满足关系  $R'$  为  $c$  的限制条件，符号表示为  $aR(?R'c)$ ，比如微软公司和语音识别有关的产品有哪些，实体为微软公司，询问关系产品，且产品的关系限制为相关技术是语音识别
4. 已知实体  $a$ ，询问其某个关系  $R$  的对应实体，且这些对应实体满足属性  $A$  为  $v$  的限制条件，符号表示为  $aR(?Av)$ ，比如李四去年发表了什么论文，实体为李四，询问关系发表文献，且文献的属性限制是发表时间为去年
5. 第 3 和第 4 种情况的限制条件能够以任意数目进行组合，提出更复杂的问句，比如谷歌公司去年申请的和深度学习相关的家居类的专利有哪些
6. 已知某种关系  $R$  对应的实体为  $b$ ，询问满足要求的实体，符号分别表示为  $?Rb$ ，比如做图像识别的实验室有哪些啊，要求返回实体类别为实验室，且关系限制研究方向为图像识别
7. 已知某种属性  $A$  的值为  $v$ ，询问满足要求的实体，符号分别表示为  $?Av$ ，比如今年上映的电影有哪些，要求返回实体类别电影，且属性限制是上映时间为今年
8. 第 6 和第 7 种情况的限制条件能够以任意数目进行组合，提出更复杂的问句，比如问今年年初上映的和喜剧相关的冯导拍摄的电影有哪些
9. 直接询问某个实体  $e$ ，但并不指明任何限制条件或是解析不出询问目的，比如直接询问姚明篮球球星

除此之外，还有诸如已知两个实体  $a$  和  $b$ ，询问它们之间的关系  $R$ ， $a?b$  这种提问方式一般出现在开放性领域的知识图谱问答中，领域相关的知识图谱问答一般需要以关系或属性为主导，本课题中不关注此类问题。对以上各种提问方式进行汇总，可以概括为三种：

- 询问关系或属性，对应提问方式 1-5。在实际提问中，属性与关系的唯一区别在于询问实体的属性时不会有限制条件，可以看作限制条件始终为空的情况，而询问实体的关系时限制条件可有可无

- 询问实体，对应提问方式 6-8。这种情况只知道限制条件，要求返回满足条件的实体，此时限制条件不能为空，否则会返回知识库中所有实体（数目太多，没有询问的意义）
  - 只能解析出实体，不知具体询问意图，对应情况 9，用特殊符号 *nil* 表示这种情形
- 本课题针对以上三类问题进行语法规则的设计并给出中间语义的逻辑形式。

### 3.1.2 语义解析难点分析

本小节中我们详细列举基于知识图谱的语义解析相比于传统语义理解任务的难点：

**多实体的识别** 在章节3.1.1中我们把询问目的概括为以下三种：询问关系或属性、询问实体、*nil*。

语义信息的载体主要是实体和限制条件，而限制条件无论是关系还是属性都必定和句中某个实体相捆绑。为此，一句话中可能出现多个实体，但最多一个主实体，其余可以看作限制条件的次实体。对于第一种询问目的，在思必驰公司生产的和语音识别相关的产品有哪些一句中，询问的最终目的是关系产品，主实体是思必驰公司，语音识别虽然也是实体，但只用来做限制条件的次实体；对于第二种询问目的，询问中没有主实体，比如做语音识别的公司有哪些，虽然句中也有实体语音识别，但同样只是用来做限制条件，主实体正是需要返回的内容，即我们询问的目的；对于第三种询问目的，无法解析出询问目的，也就没有主实体，识别出的所有实体处于同等地位，没有主次之分。基于知识图谱的解析一大难点在于如何区分这些实体的主次之分，并把次实体和对应的关系限制联系到一起

**关系或属性的提取** 这类提取不同于实体的识别，无法在输入问句的序列中准确将关系给逐字标注出来，比如询问李四今年多大，老李几岁了，李老头年龄大小，实体李四能够标注并通过归一化的操作识别出来，但对于属性年龄，并非所有句子都出现该词汇，无法从字层面提取出来；此外，递归解析中会涉及到多个关系和属性，这也极大地增加了我们解析的难度

**操作符** 之前我们举出的例子中的限制条件，默认都是判定相等（操作符 =），但实际上可能出现实验室年龄最小的人是谁，实验室发表论文最多的人是谁这类需要对限制条件进行比较大小 (<, >) 或者求和求平均 (*sum*, *avg*)、求最大最小值 (*max*, *min*) 等聚合操作符运算的问句。识别出限制条件的操作符也成了解析的任务之一

**一句多解现象** 章节1.2.1中提及关系可能存在逆关系，比如张三是张五的父亲，那么张五就是张三的儿子，同一问句的解析方式可能有多种，这就造成了解析的逻辑形式或语义表达式有多个可以接受的结果。比如语音实验室发表的论文有哪些，既可以理解为询问关系发表论文，主实体是语音实验室，没有限制条件，也可以理解为查询实体论文，限制条件是关系发表机构对应的实体是语音实验室。解析的结果有多种，不同的解析结果有不同的语义表达形式很容易造成错乱解析

**递归解析** 基于规则模板的解析方式不可能涵盖所有的解析规则，比如用户提问思必驰公司的 CEO 毕业于哪所大学，而规则模板库中只有以下两个规则模式：

---

\${#company} (公司)? 的 CEO (是谁)?  
\${#person} 毕业于哪一? 所 (大学 | 学校)

---

没有任何一个规则模式能够完全匹配上提问语句，但第一个规则模式能局部匹配子句思必驰公

司的 CEO。非递归解析只对输入语句进行一次规则模板的匹配操作，而递归解析在没有完全匹配的模板时，允许局部匹配的情况出现，先针对局部匹配的子句进行查询，并将知识库查找到的实体结果替换原语句的匹配部分，再次进行规则模板的匹配工作，直到没有任何一个模板能够匹配或知识库的返回值为属性值为止。比如前面的例子中知识库根据匹配上的子句查找到思必驰的 CEO 是高始兴，然后替换原语句匹配部分得到新的查询语句高始兴毕业于哪所大学，刚好能够与第二条规则模式匹配上，再次执行知识库的检索就能得到目标答案。这种递归解析的思想在基于规则的解析方法中很容易实现，极大地简化了我们规则的书写，即使用户针对性地提出思必驰老板的老婆的父亲的兄弟的儿子今年多大了这种问句，在编写规则时也只需要编写非递归的模板，就能解决这种多轮递归查询的问题，但在基于统计的解析方法中如何实现却是一大难点。

### 3.1.3 规则定义

基于知识图谱的语法规则主要可以分为词法、短语、句法三个组成部分：

**词法规则** 词法规则定义了询问中能够被识别出来的实体名称，分为词库 (Lexicon) 文件和引用格式两部分。一份词库文件定义一种概念的实体数据库，文件中每一行都是该概念下实体的名称，由纯文本构成；引用时使用类似于 \${#company} 的形式 (company 是实体类别，也就是概念名，词库文件的命名就采用本文件内实体的类别)，词库文件及其引用示意图参见3-1

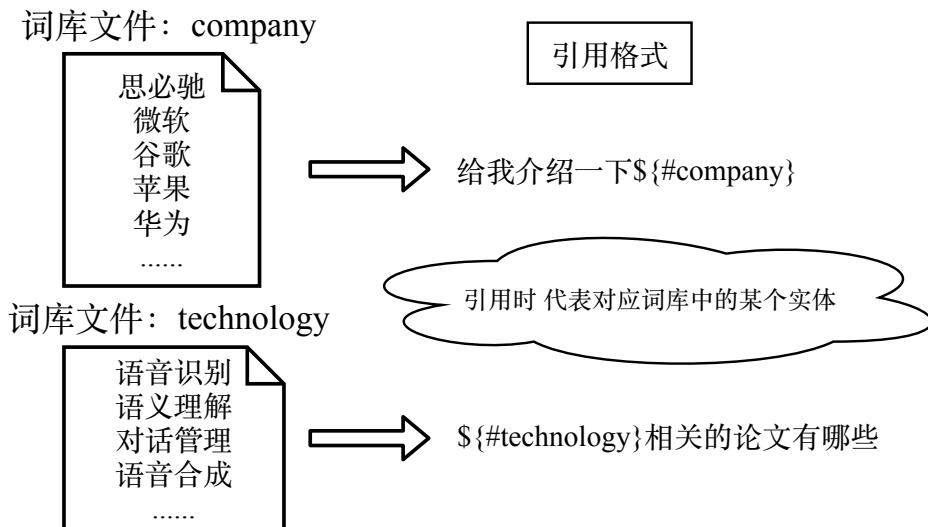


图 3-1 词库文件和引用格式

Figure 3-1 Lexicon File and Reference Format

**短语规则** 短语规则定义了询问中可能会使用的属性值限制条件，也分为限制 (constraint) 定义和引用格式两部分。

$$\text{限制名称} = \text{正则表达式模式} \Rightarrow (\text{"映射值"})$$

限制的定义由限制名称 (constraint name)、限制模式 (constraint pattern) 和映射值 (mapped value) 三部分构成。限制名称可以是任意互不冲突的命名，尽量体现该限制条件的目的；限制模式

通过正则表达式描述一个限制条件可以呈现的具体的值(目前我们支持的通用正则表达式符号有?, | 和 ()), 该部分也可以引用已经定义的其他限制条件, 但不能出现互相引用的情况(限制 a 的定义中引用限制 b, 限制 b 的定义里又引用了限制 a); 映射值是对限制条件进行值的归一化操作, 它有三种定义方式:

- 限制定义中直接省略 => ("映射值"), 代表使用匹配上的字符串本身作为映射值
- 通过索引, => ("\$ $\alpha$ ") 代表使用限制模式中第  $\alpha$ (整数) 个括号内的内容作为映射值
- 直接定义映射值, => ("string") 表示该限制模式匹配上的所有字符串都映射成统一的字符串值 string

对限制的引用和词法的引用类似, 但为了区分实体和属性值, 使用 @ 而非 #, 比如 \${@city}。短语规则的定义及其引用参见图3-2

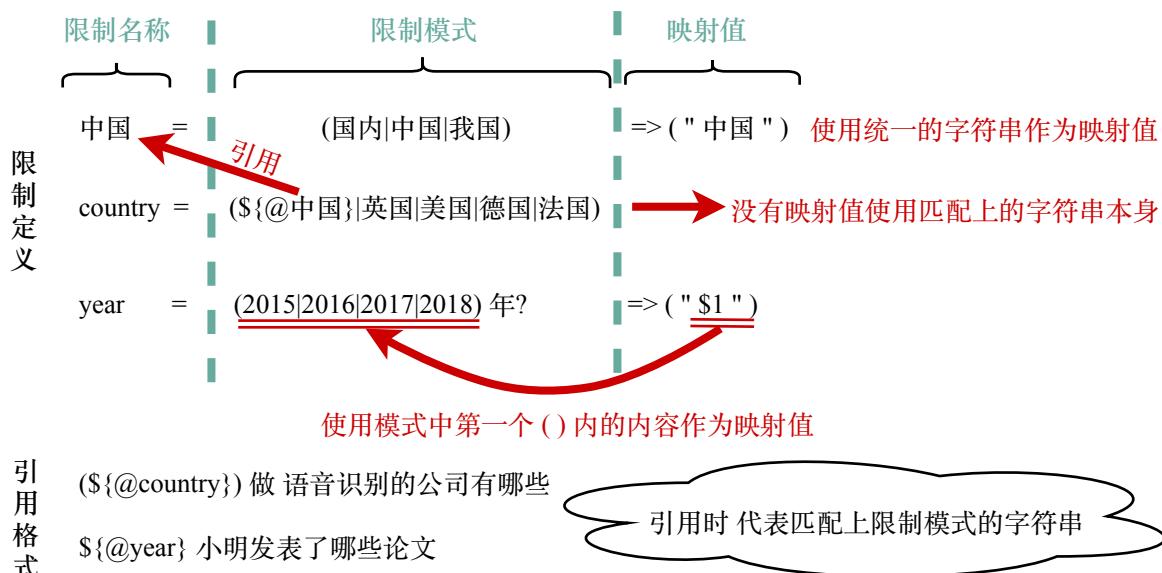


图 3-2 限制定义和引用格式

Figure 3-2 Constraint Definition and Reference Format

**句法规则** 作为规则定义的主体, 该部分定义了用于 FST 解析的规则。主要由两个部分构成, 宏定义 (macro definition) 和规则模板 (rule template)。宏定义是我们编写规则模板时的语法糖 (syntactic sugar), 将一些可能会反复使用的正则表达式语法模式通过宏定义封装起来, 比如 ask\_time = (什么时候 | 什么时间 | 哪一? 年)。在编写解析规则时可以直接引用, 省去我们编写大量规则时的重复性工作。调用宏时和调用实体词库和限制条件类似, 不过此时不需要 # 和 @ 表征实体和限制, 直接通过 \${宏名称} 调用。规则模板分为规则模式 (rule pattern) 和语义表示 (semantic representation) 两个部分:

规则模式 => 语义表示

- 规则模式类似于限制模式, 由正则表达式构成, 可以引用实体词库、短语限制还有宏定义, 用来定义文本的匹配模板, 比如

宏定义	<code>what=(哪一?些 啥 什么)</code>
规则模式	<code> \${#company} \${@year} (发布 发行 研发 生产) 了? \${what} 产品</code>

- 语义表示用规范的语义结构定义了规则模式中的语句传达了用户什么询问的意图，可以看作对规则模式的一种语义标注。具体介绍参见下一小节3.1.4。关于具体的规则示例可以参见附录C。

### 3.1.4 中间语义表示及逻辑形式

#### 3.1.4.1 中间语义表示

本小节我们详细地介绍上一小节3.1.3中句法规则的语义表示部分。中间语义表示由返回值类型(*return type*)和过滤模板(*filter template*)两个部分组成。

(返回值类型, [过滤条件三元组列表])  
过滤模板

返回值类型分别针对章节3.1.1中总结归纳的三种询问类型：

- 询问属性或关系，返回值类型的形式为( $\#\alpha$ , 关系名或属性名, ?)的三元组形式，其中 $\#\alpha$ 代表规则模式中第 $\alpha$ 个 $\${\#xxx}$ 形式的实体，?是占位符
- 询问实体，返回值类型使用对应实体的概念名
- 只能解析出实体的采用(*nil*)标记

过滤模板是由过滤条件的三元组形成的列表，三元组格式：

(限制条件的操作符, 关系名或属性名,  $\#\alpha$ 或 $@\beta$ )

- 限制条件的操作符表示该限制条件是如何对查询目标进行限制的，比如对于年份类的属性，是刚好看在这一年，即相等(eq)，还是在这一年之前(小于，lt)或之后(大于，gt)，对于发表论文的关系，是要根据发表论文的总数量筛选(计数，sum)还是取每年平均发表数量(平均，avg)
- 关系名和属性名都由本体结构预先定义，参见附录BSpeechLab领域本体结构
- $\#\alpha$ 或 $@\beta$ 表示规则模式中第 $\alpha$ 个 $\${\#xxx}$ 形式的实体(此时为关系限制)或第 $\beta$ 个 $\${@xxx}$ 形式的属性值(此时为属性限制)。注意#和@分别单独计数

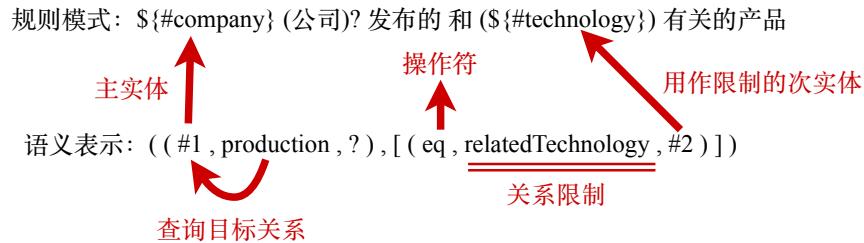
将规则模式以及中间语义表示的返回值类型、过滤模板结合起来，对规则模板的图解参见3-3。

目前定义的中间语义表示虽然能够涵盖足够多的询问情况，但这种表示方式还不足以解决3.1.2小节递归解析的问题，因此我们需要一种更加强有力的语义表达方式(逻辑形式)和知识库交互。

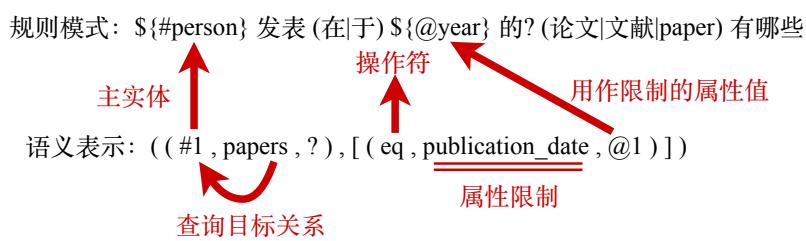
#### 3.1.4.2 基于知识图谱的语义逻辑形式

按照3.1.4.1小节定义的中间语义表示，在执行多轮递归解析时，只能一轮一轮地和知识库进行交互，获取了目标实体值、替换上一轮匹配的语句之后，再次寻找规则模板进行匹配，当递归次数

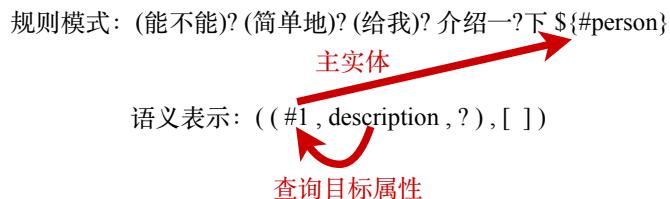
### 询问关系(加关系限制)



### 询问关系(加属性限制)



### 询问属性(一般没限制条件)



### 询问实体(一般需要限制条件)



### 特殊 nil



图 3-3 规则模板詳解

Figure 3-3 Explanation of Rule Template

较多时会浪费不少时间在与知识库的交互上。为此，我们探寻一种逻辑形式能够将我们每一轮递归查询的目标依次封装起来，并将这种拓扑结构，一次性地传递给知识库。实验中我们采用的结构为

```
{
    "ne_0": { 实体0 }, ..., "ne_k": { 实体k },
    "er_(k + 1)": { 查询目标1 }, ..., "er_(k + m)": { 查询目标m }
}
```

其中  $ne_i$  形式的键名按  $i$  从小到大依次表示问句中识别出的实体， $er_j$  形式的键名按  $j$  从小到大依次表示每一轮递归时的查询目标。

实体  $i$  ( $ne_i$ ) 的字典结构

```
{
    "type": 实体类型, 即概念名
    "value": string值, 即实体名
}
```

查询目标  $j$  ( $er_j$ ) 的字典结构

```
{
    "expr_type": "entity" 或者 "rel_attr",
    "pred": 关系名或属性名或概念名,
    "filter": filter_list,
    "ent_idx": "ne_(数字)"、"er_(数字)"的形式或null
}
```

$ent\_idx$  是指向主实体的链接， $pred$  是我们查询的目标， $expr\_type$  表示返回值类型，它只有两种取值，为  $rel\_attr$  时，询问关系或属性，此时  $ent\_idx$  指向主实体的实体字典，主实体用  $ne_{(数字)}$  或  $er_{(数字)}$  的形式表示，如果是  $er_{(数字)}$  的形式，就用到了递归查询，知识库需要先检索到  $er_{(数字)}$  对应的实体才能继续当前的查询； $expr\_type$  为  $entity$  时，询问实体，此时没有主实体， $ent\_idx$  为  $null$ ；如果中间语义表示的返回类型是  $nil$ ，则逻辑形式里就不会有查询目标  $er_j$  这一类的字典。 $filter\_list$  是限制条件的列表，每一个限制条件也通过一个字典结构来表示：

```
{
    "op": 操作符比如"eq",
    "constraint": 关系名或属性名如"technology",
    "value": string(属性值)或"ne_{(数字)}"、"er_{(数字)}"的形式(指向关系对应的实体字典)
}
```

逻辑形式相比中间语义表示作出的重要改变是对实体的值通过  $ne_{-}(\text{数字})$  或  $er_{-}(\text{数字})$  的形式来引用之前出现的实体字典或前几轮递归查询获取的实体。

具体的逻辑形式的例子参见图3-4和3-5

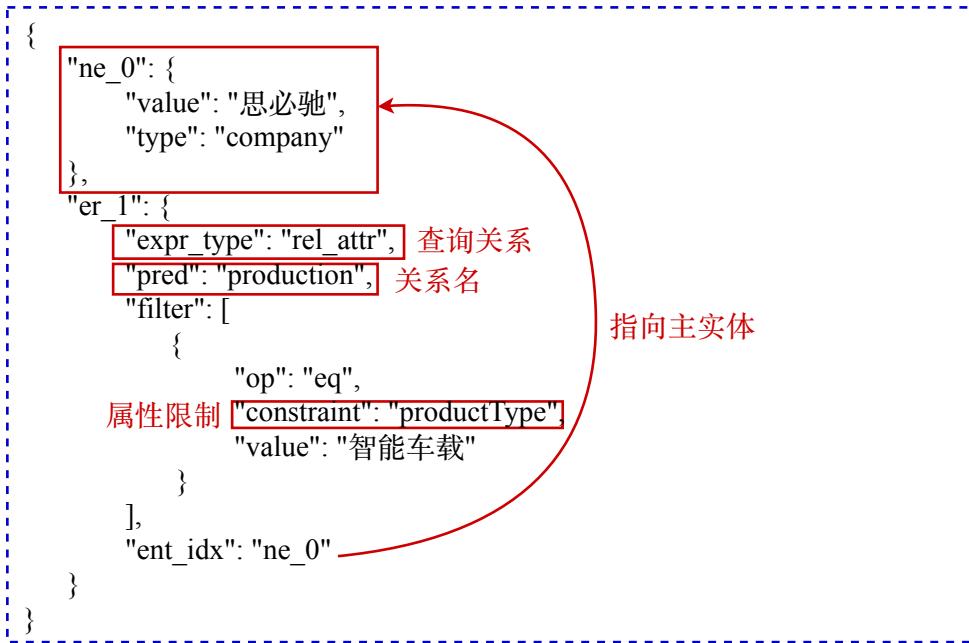
## 3.2 基于 FST 的规则解析原理

给定一个输入句子，基于有限状态转移机 (FST) 的规则方法单轮解析分为以下几个步骤：

1. 前期资源准备，将词法规则（实体词汇）、短语规则（属性限制条件）和句法规则分别构建合并的有限状态转移机
2. 根据词法规则的有限状态转移机，匹配句子中所有的实体词汇
3. 根据短语规则的有限状态转移机，在原始文本和步骤 2 的基础上匹配所有可能的短语限制条件
4. 根据句法规则的有限状态转移机，在原始文本和步骤 2、3 的基础上匹配所有的句法规则
5. 将步骤 4 匹配上句法规则全部输出（包括全局匹配和局部匹配），根据某种原理排序后筛选出最合适的规则（一种启发式的方法是根据匹配上规则的长度来排序并筛选）并提取对应规则的中间语义表示

具体的解析原理参考附录A

思必驰有哪些智能车载类产品



微软公司cto的名字叫什么

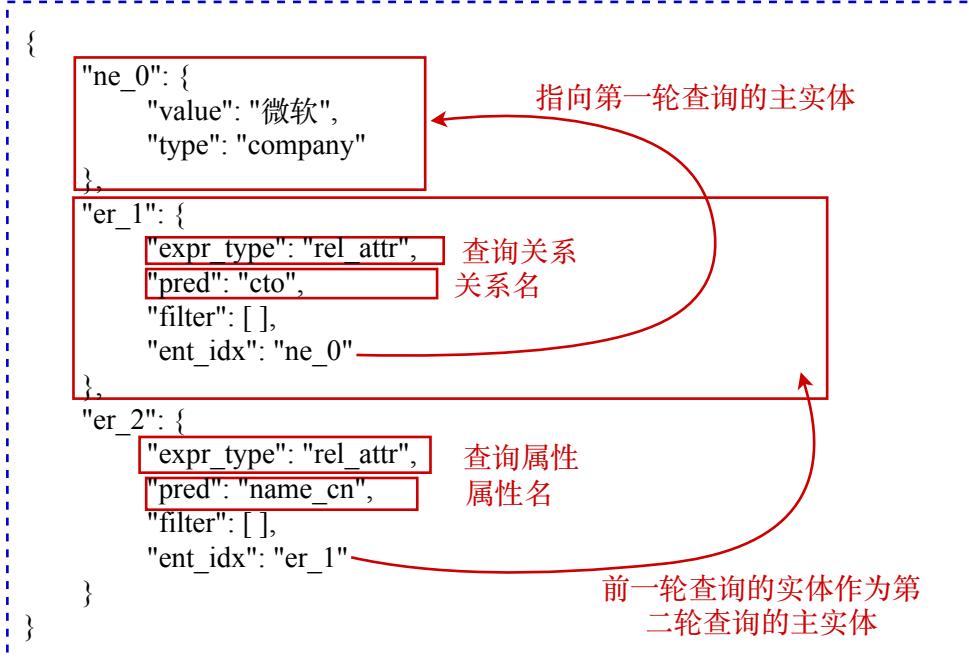


图 3-4 逻辑形式样例 1

Figure 3-4 Example 1 of Logical Form

### 国内做语音识别的公司有哪些

```
{
  "ne_0": {
    "value": "语音识别",
    "type": "technology"
  },
  "er_2": {
    "expr_type": "entity", 查询实体
    "pred": "company", 实体类型，即概念名
    "filter": [
      {
        "op": "eq",
        "constraint": "nation", 属性限制
        "value": "国内"
      },
      {
        "op": "eq",
        "constraint": "field_of_work", 关系限制
        "value": "ne_0" 指向用作关系限制的实体
      }
    ],
    "ent_idx": "null" 查询实体，没有主实体，null
  }
}
```

### 思必驰公司的小明

```
{
  "ne_0": {
    "value": "思必驰",
    "type": "company"
  },
  "ne_1": {
    "value": "小明",
    "type": "person"
  }
}
```

**无法判断查询意图，nil**  
**只有实体字典，没有"er\_"形式的字典**

图 3-5 逻辑形式样例 2

Figure 3-5 Example 2 of Logical Form

## 第四章 数据集来源及神经网络统计模型

本章先介绍实验中的数据来源以及预处理的工作，然后逐步深入地介绍基于统计方法的实验中我们使用的神经网络模型以及解码的实现。

### 4.1 数据集来源

本课题最终目的是实现智能问答系统，实验中的数据来自于两个领域，AISpeech 和 SpeechLab，分别针对的是思必驰公司和上海交通大学智能语音实验室的信息问答。两个领域数据来源渠道不同，本节具体介绍两个领域的数据获取途径，主要分为训练数据和测试数据两个部分。

#### 4.1.1 AISpeech 领域

AISpeech 领域的数据主要来自于思必驰公司员工的已有工作：

- 本体结构有概念(实体类别)5种，不同的属性27种，不同的关系23种(有些属性或关系意义相近，在不同的实体类别里命名方式相同，算作同一种)
- 人工编写的解析规则500条，按语义表达式类别汇总分类，一共有95种不同语义表达式的规则(比如询问某人、某公司的简介，虽然针对不同实体类型，但语义表达形式相同，算作同一种)
- 编写的规则中并没有覆盖所有的属性(有些概念的属性仅仅用来唯一标识实体；有些属性属于元属性，不用于查询，仅用于组织知识库结构；还有些属性很少在实际问答中被提及)，规则中只涵盖了概念5种，属性21种，关系23种

##### 4.1.1.1 训练数据集

统计模型需要海量训练数据，实验中的训练集生成方式如下：

1. 根据词法规则为所有概念构建一个概念到实体的字典 *concept\_to\_entity*
2. 根据短语规则构建有限状态机，然后输出有限状态机所有路径，构建属性名到属性限制取值的字典 *attr\_to\_value*
3. 为每类规则分别构建有限状态机，展开每条规则的正则表达式部分
4. 对于每类规则，根据步骤3构建的状态机，随机生成有限状态机的路径，并根据步骤1概念到实体的字典 *concept\_to\_entity* 和步骤2属性名到属性值的字典 *attr\_to\_value*，随机挑选并替换路径中 \${#xxx} 和 \${@xxx} 的成分，得到原始的语料问句并自动取得语义标注(解析规则的语义表示部分就是标注信息)，以此批量生成训练数据

由于每一类规则展开正则表达式后可以生成的问句总数都不相同，随机生成问句时不同语义类别的样例数不宜相差太大，否则可能有些语义类别的训练样本不够，导致模型学习不到位，在这部分数据上欠拟合。实验中我们采取的平衡语义类别分布的方法：规范每一类规则生成的总样例数，限制在1000到5000之间。若处于这个范围之内，将所有路径枚举；若大于5000句，则只随机挑选其

中 5000 句；若小于 1000 句，则所有样例复制多份直到大于 1000 句。最终我们一共生成 AISpeech 领域训练集 139004 句。

#### 4.1.1.2 测试数据集

AISpeech 领域的训练集由两部分组成，第一部分由 4.1.1.1 小节的方法从规则生成，然后经过员工粗糙地检查改写后使部分拗口的语句更加口语化、容易被接受，这部分测试数据有 6274 句，第二部分由员工模拟问答情形，抛开所有编写的规则，直接编写测试语句，这部分测试集来源更加接近现实情况，但编写和标注也更加费时费力，这部分测试数据一共有 2616 句。将这两部分的测试数据合并，测试集大小为 8890。

### 4.1.2 SpeechLab 领域

SpeechLab 领域所有数据均来自上海交通大学智能语音实验室内部人员。本体结构由实验室成员共同设计：

- 本体结构有概念(实体类别)7 种，不同的属性 33 种，不同的关系 14 种
- 自行编写的解析规则 506 条，按语义表达式类别汇总分类，一共有 98 种不同语义类型的规则，部分规则示例参考附录 C
- 编写的规则中涵盖了概念 7 种，属性 30 种，关系 14 种

SpeechLab 本体结构设计参见附录 B

#### 4.1.2.1 训练数据集

SpeechLab 领域训练数据集生成方式和 AISpeech 领域相同，最终一共生成 371761 条训练语句。

#### 4.1.2.2 测试数据集

一个新领域的测试集标注问题常常耗时且耗力，尤其是语义理解的任务，由于中间语义表示形式复杂，需要一定的先验知识，专业之外的标注人员常常要花较长时间学习领域相关的语义表达方式，即便如此，标注的句子中不合规范或是错误的标注依旧较多。为此，SpeechLab 领域测试数据的收集方式借鉴 [15] 中训练数据的构建方法。

[15] 中提出一种功能驱动 (functionality-driven) 的构建语义解析器的框架，将数据的收集分成两类任务，一类是领域相关 (domain-specific)，另一类属于领域无关 (domain-general)。数据集构建者先根据领域任务定义目标领域的本体结构和词库、谓语库，然后根据领域无关的文法，由词库和谓语库生成大量的规范语句 (canonical utterance) 及对应的标注 (Lambda DCS logical form)。这部分原始的语句并非日常询问语句，不能直接用作训练数据，比如 *article that has the largest publication date*，但却很容易理解它的询问意图。接下来，构建者将这些规范语句通过众包 (crowdsourcing) 的方式分发给标注者改写句子的询问方式，但不改变询问意图。由于标注在通过领域无关文法生成规范语句时已经生成，所以不需要标注者特别去标注，而且改写问句的任务即便对于非专业人员一样可以手工完成。具体流程参见图 4-1

本课题的实验中借鉴并改进了上述方法，用来收集 SpeechLab 领域的测试数据，图 4-2：

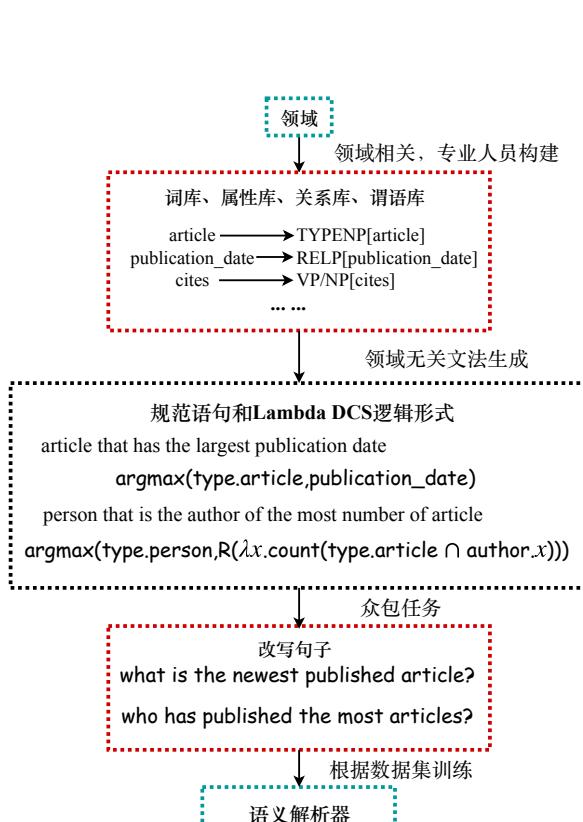


图 4-1 构建训练集流程

Figure 4-1 Procedure of Constructing TrainingSet



图 4-2 改进后构建测试集流程

Figure 4-2 Improved Procedure of Constructing TestSet

1. 仿照训练集生成的方法，生成测试样例的规范语句库以及对应的标注，注意类别分布平衡
2. 由步骤 1 的规范语句库生成问卷分发给标注者进行语句的改写，每个标注者被随机分发语句库中数十条；若是标注者改变了原语句中语义槽(slot)的值，需要标注者在问卷提供的 slot 替换接口中填写对应的值(比如原语句是张三在 2018 年发了啥子论文，改写时如果用户将 slot 张三换成了李四，需要在提供的填空框 \${#person} = \_\_\_\_ 中填写李四)
3. 测试数据回收，并将原来标注中 slot 值重新赋值为标注者替换的 slot 值

在我们的方法中，生成的规范语句相比 [15] 更加容易理解(我们的规范语句根据领域相关的规则生成，而非领域无关的文法)，且我们提供给标注者改写 slot 的接口，收集到的数据不仅语法结构上更合理，内容上也更具多样性，而 [15] 中只是改写句子的表达方式，只能在语句结构上改写。通过上述方法，在智能语音实验室内部分发标注任务，一共收集到测试集数据 478 句。数据收集时的问卷题目样例和整理后的测试数据样例参考附录D。

为了更好地衡量模型的鲁棒性(robustness)和泛化(generalization)性能，对于收集到的测试数据，我们不经过任何数据清洗的工作，直接使用。

## 4.2 神经网络统计模型

在介绍实验中使用的各种统计模型之前，在1.2.2.1小节的语义框架中，我们提出把扁平化后的语义框架分解为两个子任务，本节先介绍这两个子任务的具体定义。

### 4.2.1 子任务分类

#### 4.2.1.1 slot filling 任务

槽填充(slot filling)，实质上是一个序列标注的任务，它的输入是一个序列，输出是相同长度的另一个标记序列，输出的每一个标记都是在标记空间的一个类别(对应到神经网络中输出是多项分布)，因此，整个任务可以看做一系列的分类任务，只是这些分类任务之间互相依赖，标记之间存在某种相关性。

对于slot filling任务来说，输出的标记采取通用的begin-in-out(BIO)标记方式<sup>[16]</sup>。没有语义信息的输入元素标记为O(out)；某一个新的语义槽开始用B-[slotName]标记，由于一个实体名称的长度可能会很长，比如一篇论文的名字，将第一个单词之后的部分都用I-[slotName]标记，表示还处于该语义槽范围之内。具体的标记例子参见4-3

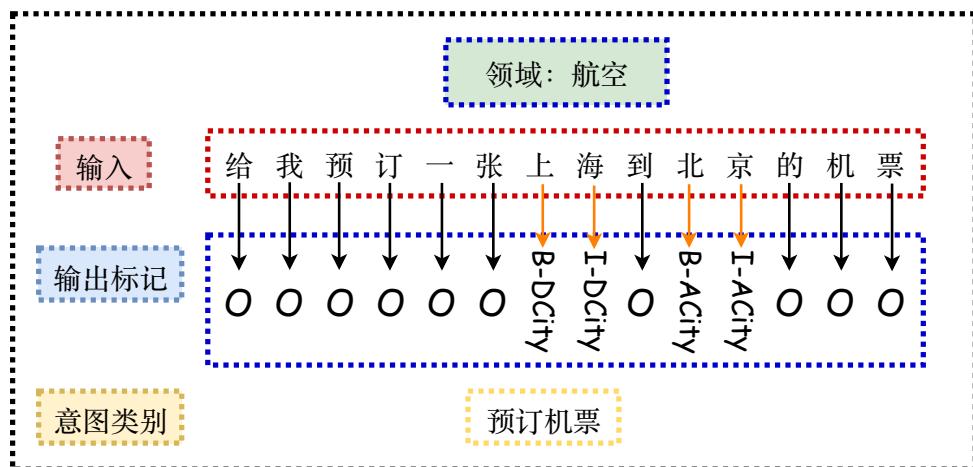


图 4-3 槽填充任务示例

Figure 4-3 Example of Slot Filling Task

#### 4.2.1.2 intent classification 任务

意图分类是一个典型的多分类任务，每一个输入句子都对应一种意图的类别，比如在1.2.2.1的对话语义动作中，每一种对话动作就是一个类别。该任务的输入是一整句话，神经网络的输出是在所有句子类别上的多项分布，由于句子类别和句中语义槽信息存在一定相关性，因此常常和slot filling任务联合学习，共享网络参数。

## 4.2.2 结构化语义扁平化处理

语义输出的最终要求是3.1.4.2小节的逻辑形式，它具有树状的层次化结构，而神经网络的输出往往是扁平化的。因此，我们需要定义一种扁平化的逻辑形式，能实现和最终逻辑形式之间相互转化。由于大多数问答只会用到一轮递归，且训练数据中也只包含单轮解析的语料，统计模型的方法中我们只考虑运用一轮递归解决的问答。

扁平化的语义结构表示见图4-4

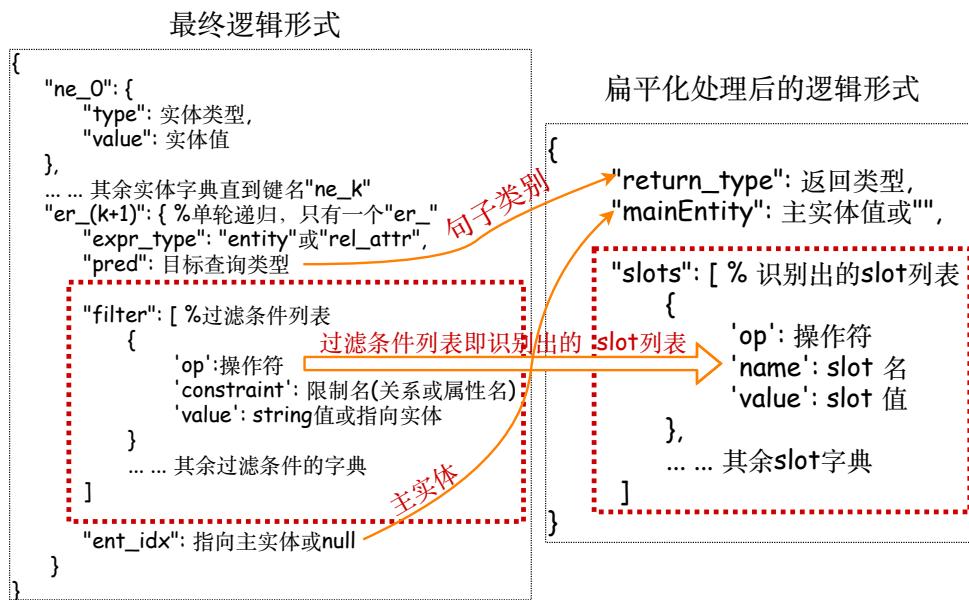


图 4-4 逻辑形式扁平化处理

Figure 4-4 Flatten Logical Form

### 4.2.2.1 从神经网络输出到语义逻辑形式

- 由于我们只考虑一轮递归的情况，目标的查询类型  $er_{-}$  只有一个，直接将其作为句子分类任务的结果；
- 而逻辑形式中所有的限制条件和实体的识别都作为 slot filling 的任务，用对应的限制名(即属性名或关系名)和实体类型名来标注；
- 对于对主实体和其他实体的区分问题，将其并入到 slot filling 的任务中，在 slot 的识别中，主实体之外的其他实体用对应的关系名来标注(或不标注，如果没有对应关系限制)，而主实体的 slot 标记用一套特殊的符号 \${#实体类型} 来表示主实体，最后从神经网络输出中查找 \${#实体类型} 形式的标记，提取为主实体(解码后处理，参见4.2.7)；
- 本课题中训练集和测试集都只有判定相等的操作符，因此操作符默认都是  $eq$ ，未来如果引入其他操作符，可以将其和关系、属性捆绑在一起，定义诸如  $le$ - $year\_of\_join$ (加入时间小于等于某年)、 $max$ - $publication\_date$ (最新发表的)的标签或作为独立的另一个 slot filling 的任务

### 4.2.3 数据预处理

#### 4.2.3.1 输入词表构建

由于两个领域的训练数据中英文交叉，对于输入文本，我们对整个训练集分词并构建词表索引的原则为：

- 英文以单词为单位，中文以字为单位划分，其他特殊字符以字符为单位划分
- 英文标点保留下来（论文英文标题中常常出现连字符），中文字符删除
- 预留特殊符号：
  - <PAD> padding symbol，用来补齐不同长度的输入句子，作占位符
  - <UNK> unknown word，测试集中词汇不一定全都出现在训练集中，该标记表示所有训练集未出现的词
  - <BOS> begin-of-sentence，用来标记一句输入的开始（常常用在 Seq2Seq 类模型中）
  - <EOS> end-of-sentence，用来标记一句输入的结束（常常用在 Seq2Seq 类模型中）
  - <BEOS> Seq2Seq 类模型 Decoder 部分用来生成第一个标记的输入，也可以用 <BOS> 替代，本课题的实验中将两者区分

通过以上原则，AISpeech 领域和 SpeechLab 领域构建的词表大小分别为 950 和 1310

#### 4.2.3.2 词表索引到词向量 word2vec

由于神经网络的输入是多维特征的向量，构建完了输入的词表，还需要将每个单词映射为对应的词向量。常用的映射方法有三种：

- One-hot 词向量表示方法，输入向量维度就是词表大小，一个单词对应一个维度，每一个单词的向量表示都是一个稀疏的向量，只在本身单词对应的维度为 1，其余维度上都是 0
- 预训练的词向量，网上有很多开源的基于很大语料库训练的词向量的映射库，常见的有 Word2Vec、GloVe，可以直接从中查询使用
- 神经网络自动学习词向量的表示，神经网络每一个隐藏层都可以看作学习输入的特征表示，词向量也是单词的一种特征表示方法，完全可以交由神经网络自主学习，只需输入 One-hot 的表示方法，把经过第一个线性层的输出看作网络学习到的词向量；还有一种更好的方法是输入预训练的词向量，在此基础上神经网络继续学习并完善词向量的表示

在本课题的实验中，我们采取第三种方法，利用 PyTorch 第三方开源库的 Embedding Layer(词嵌入层) API 由单词的 One-Hot 表示学习对应的词向量表示。

#### 4.2.3.3 输出标记空间

有了扁平化的逻辑形式，我们就可以根据训练集出现过的标注分别定义 AISpeech 领域和 SpeechLab 领域句子分类和槽填充标记的类别：

##### AISpeech

**句子类别数目** 查询关系 21 种，查询属性 23 种，查询实体 2 种，加上特殊类别 *nil*，一共  $(21 + 23 + 2 + 1 = 47)$  种

**槽填充标记类别数目** 关系限制 3 种，属性限制 2 种，主实体类型标记 5 种，每种各有 *B/I* 两种前缀，再加上特殊标记 *O*，一共 ( $3 * 2 + 2 * 2 + 5 * 2 + 1 = 21$ ) 种

### SpeechLab

**句子类别** 查询关系 14 种，查询属性 30 种，查询实体 5 种，加上特殊标记 *nil*，一共 ( $14 + 30 + 5 + 1 = 50$ ) 种

**槽填充标记类别** 关系限制 3 种，属性限制 12 种，主实体类型标记 7 种，每种各有 *B/I* 两种前缀，再加上特殊标记 *O*，一共 ( $3 * 2 + 12 * 2 + 7 * 2 + 1 = 45$ ) 种

#### 4.2.4 Bi-RNN 模型

由于双向 RNN 模型近乎在所有任务中都比单向模型取得了更好的结果，实验中我们就不再构建单向 RNN 的模型。

- 模型的网络主体结构是一个 Embedding 层、一个双向的 RNN 层、两个独立的线性层和输出 softmax 层的顺序连接
- 两个线性层一个用于 slot filling 任务，另一个用于 intent classification 任务
- 句子分类任务经过线性层之前需要将 BiRNN 层两个方向的最终输出结果先拼接 (concatenate) 起来
- 两个子任务在双向 RNN 层共享参数，但线性层有着各自的私有参数
- 双向 RNN 层的神经元一般使用 2.3.2 小节介绍的 LSTM<sup>[17]</sup> 或 GRU 单元，缓和输入序列长期依赖的问题

Bi-RNN 模型具体网络结构参见图 4-5

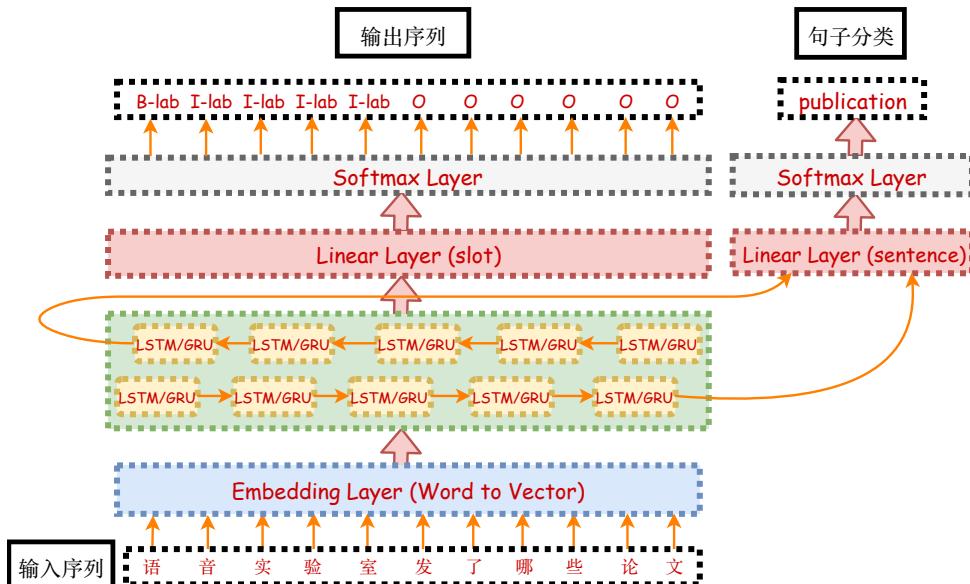


图 4-5 双向 RNN 网络模型

Figure 4-5 Bi-directional RNN Neural Network Model

#### 4.2.5 Bi-RNN+CRF 模型

Bi-RNN+CRF 模型为了解决输出 slot 标记之间的依赖问题(比如  $I - [slotName]$  标记之前只能是  $B - [slotName]$  标记), 在 BiRNN 模型线性层之后添加了一个条件随机场(CRF)的层<sup>[18]</sup>。和传统的条件随机场需要自己构造特征函数不同, Bi-RNN+CRF 模型输入给 CRF 层的特征由 BiRNN 模型的输出提供。2.2.2 小节式 2-2 给出了条件随机场概率计算公式, 本模型中将其改写为

$$\begin{aligned} P(\mathbf{y}|\mathbf{x}) &= \frac{\exp(score(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp(score(\mathbf{x}, \mathbf{y}'))} \\ score(\mathbf{x}, \mathbf{y}) &= \sum_i (\log \Psi_{emit}(y_i \rightarrow x_i) + \log \Psi_{trans}(y_{i-1} \rightarrow y_i)) \\ &= \sum_i (h_i(y_i) + \mathbb{P}_{y_i, y_{i-1}}) \end{aligned}$$

特征函数分为两种, 分别是发射特征函数(emission feature function)和转移特征函数(transition feature function), 为方便处理, 都定义成  $\log$  形式的势函数  $\log \Psi_{emit}$  和  $\log \Psi_{trans}$ 。其中输入序列在位置  $i$  处的发射特征来自于 BiRNN 网络在时刻  $i$  的输出  $h_i(y_i)$ , 转移特征存储在一个  $T \times T$  的矩阵  $\mathbb{P}$  中,  $T$  是输出标记集合大小,  $\mathbb{P}_{j,k}$  表示从标记  $k$  转换为标记  $j$  的分数。定义了条件随机场的特征函数, 就可以根据 2.2.3 和 2.2.4 小节分别计算概率和预测标记序列, 这里不再赘述。

关于前向向量的计算, 这里介绍一个注意事项或技巧—Log-Sum-Exp。在计算前向向量  $\alpha$  时, 记输出时刻为  $i$ , 输出标记为  $j$  的非归一化概率为  $\alpha_i(j)$ , 输出标记集合为  $T$ , 则

$$\alpha_i(j) = \sum_{j' \in T} \Psi_{emit}(j \rightarrow x_i) \times \Psi_{trans}(j' \rightarrow j) \times \alpha_{i-1}(j')$$

这种计算是数值不稳定的, 可能会发生数值下溢(underflow)的情况, 即接近零的数被四舍五入为零。当对  $\alpha_i(j)$  取对数  $\log$  时会得到结果  $-\infty$ 。实际操作中, 我们直接计算  $\log \alpha_i(j)$ ,

$$\begin{aligned} \log \alpha_i(j) &= \log \underbrace{\sum_{j' \in T} \exp(\log \Psi_{emit}(j \rightarrow x_i) + \log \Psi_{trans}(j' \rightarrow j) + \log \alpha_{i-1}(j'))}_{\text{乘法转化为加法}} \\ &= \max(j^*) + \log \sum_{j' \in T} \exp(\log \Psi_{emit}(j \rightarrow x_i) + \log \Psi_{trans}(j' \rightarrow j) + \log \alpha_{i-1}(j') - \max(j^*)) \end{aligned}$$

其中

$$\max(j^*) = \max_j \{ \log \Psi_{emit}(j \rightarrow x_i) + \log \Psi_{trans}(j' \rightarrow j) + \log \alpha_{i-1}(j') \}$$

Bi-RNN+CRF 模型具体网络结构参见图 4-6

#### 4.2.6 Attention 模型分类及 Focus 模型

论文 [19] 中将 Attention Model 分为全局注意力(Global Attention)和局部注意力(Local Attention)两种类型。2.3.4 小节介绍的就是全局注意力, 局部注意力在  $t$  时刻计算中间语义向量  $c^{(t)}$  时只会考虑编码器的部分输出隐向量, 而非全部。Local Attention 参考图 4-7。此外, Local Attention 模型中还有两个重要的改变:

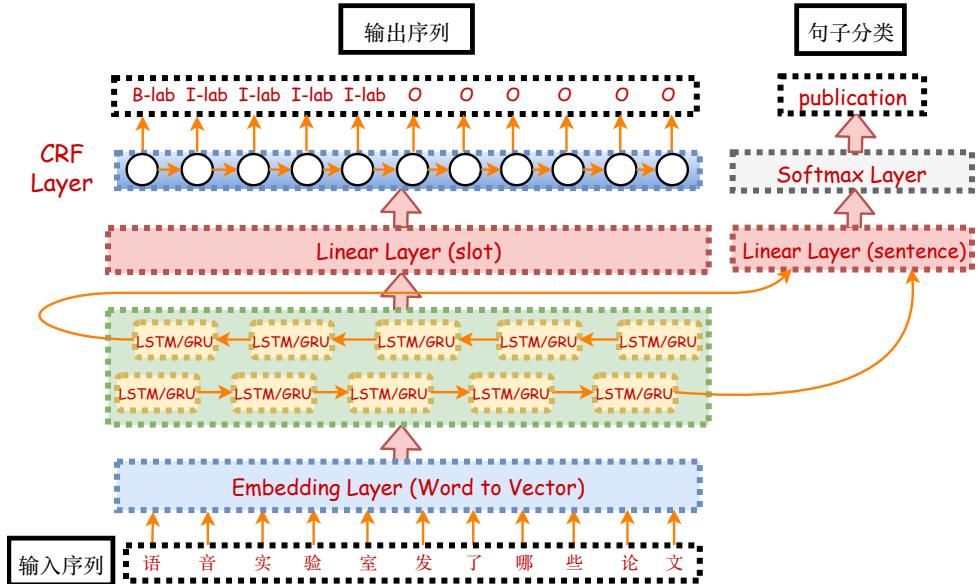


图 4-6 双向 RNN+CRF 网络模型

Figure 4-6 Bi-directional RNN Plus CRF Neural Network Model

- 中间语义向量  $c^{(t)}$  在  $t$  时刻解码器 LSTM/GRU 网络输出之后才和隐状态拼接(记为 post\_attention), 再经过线性层获取输出标记的分布, 而非和输入标记的 embedding 一同作为解码器 LSTM/-GRU 的输入(记为 pre\_attention), 数学形式描述为

$$\begin{aligned} h_{decoder}^{(t)} &= f(y^{(t)}, h_{decoder}^{(t-1)}) \\ \tilde{h}_{decoder}^{(t)} &= \tanh(W[c^{(t)}; h_{decoder}^{(t)}]) \\ p(y^{(t)} | y^{(<t)}, x) &= \text{softmax}(W_{out} \tilde{h}_{decoder}^{(t)}) \end{aligned}$$

- align model 除了前馈网络 ( $e_{tj} = v_{align}^T \tanh(W_{align}[h_{decoder}^{(t)}; h_{encoder}^{(j)}])$ ), 还提出了多种对齐模型

$$\begin{aligned} c^{(t)} &= \sum_{j=1}^{T_x} \alpha_{tj} h_{encoder}^{(j)} \\ \alpha_{tj} &= \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})} \\ e_{tj} &= align(h_{decoder}^{(t)}, h_{encoder}^{(j)}) \quad (\text{由于采取post\_attention, 注意是 } h_{decoder}^{(t)} \text{ 而不是 } h_{decoder}^{(t-1)}) \\ &= \begin{cases} h_{decoder}^{(t)}^T h_{encoder}^{(j)} & dot \\ h_{decoder}^{(t)}^T W_{align} h_{encoder}^{(j)} & general \\ W_{align} h_{decoder}^{(t)} & location \end{cases} \end{aligned}$$

其中,  $W_{align}$  和  $v_{align}$  都是网络中需要学习的参数

由于这些模型结构针对的是输入输出序列长度一般不相等的机器翻译问题, 而我们的任务中输入输出序列长度必须相等, 因此需要对 Local Attention 模型作出一些修改, 使用 Focus 模型, 网络结构图参见4-8

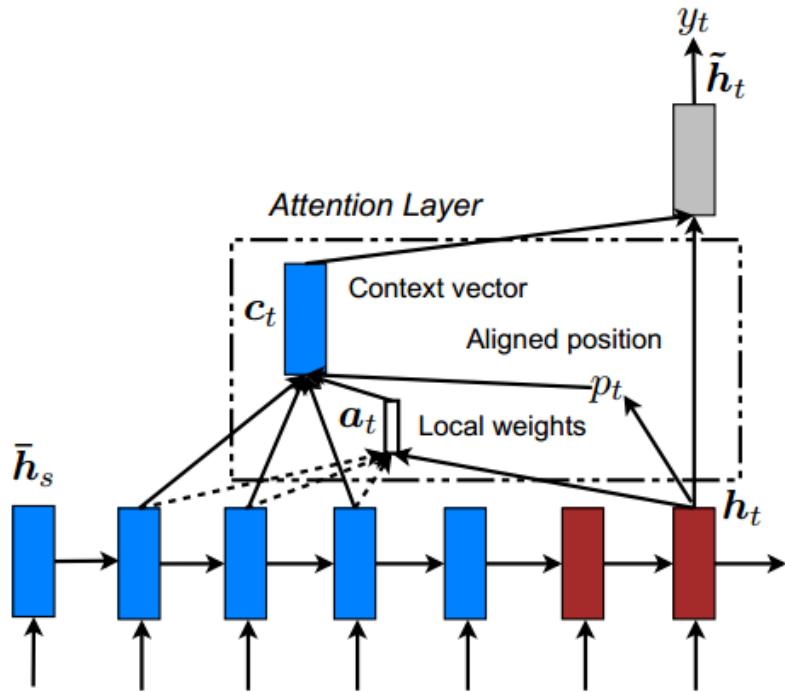


图 4-7 局部注意力模型

Figure 4-7 Local Attention Model

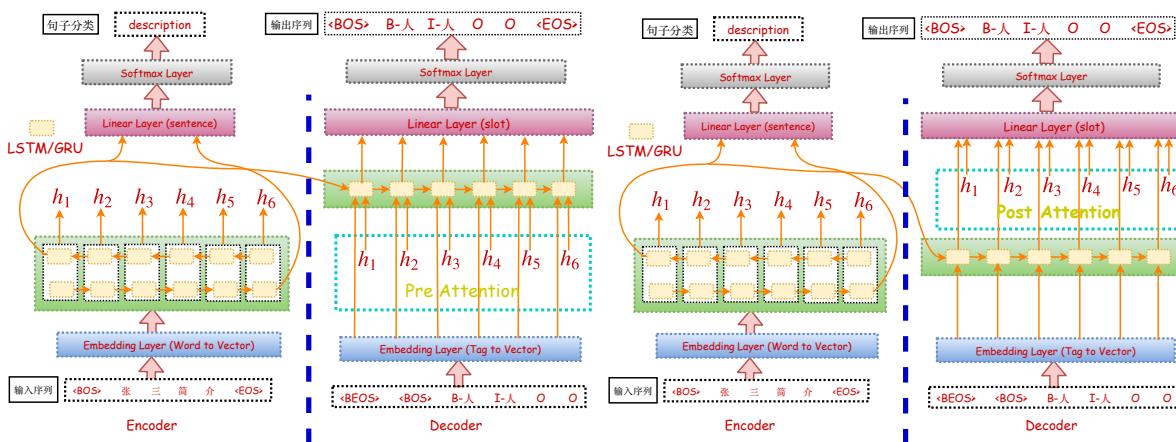


图 4-8 Focus 模型结构图 (Pre-Attention 和 Post-Attention 两种)

Figure 4-8 Focus Model(Pre-Attention and Post-Attention)

- Focus 模型可以看作 Local Attention 模型的一种变体，encoder 部分每一个阶段的输出分别各自作为语义向量，对齐地和 decoder 部分的输入拼接在一起，数学表示为

$$c^{(t)} = h_{encoder}^{(t)}$$

- 编码器部分是双向 LSTM/GRU，而解码器部分是单向的，解码器隐状态初始化时使用编码器逆向的最终隐状态
- 实验中我们尝试了在解码器 LSTM/GRU 网络输入阶段 (Pre-Attention) 和输出阶段 (Post-Attention) 分别使用注意力的情形
- < BEOS > 是输出序列开始的标志，固定用作产生第一个输出标记 < BOS >

## 4.2.7 解码及后处理

### 4.2.7.1 神经网络输出解码

解码时，对于句子分类的结果，从中直接挑选最大概率的类别标记作为返回值类型；而 slot 输出的结果是序列每一个位置在各个标记上多项分布，由于各个位置处标记的互相依赖关系，还需要从中挑选出最合理的输出标记路径。

**Bi-RNN 模型** 最简单的做法是直接取每个位置最大概率的分布，但容易得到 (*O, I-person*)、(*B-person, I-company*)、(*I-person, I-company*) 这三种不合理的连续输出标记，为避免这种情况发生，仿照 CRF 模型的 Viterbi 解码算法，我们使用 BIO-Decoder 算法，原理是将输出标签的前后依赖关系考虑进去，若是前后两个标记出现前面三种不匹配的情况之一，计算概率时给出一定的惩罚因子，动态规划地选出最合适的  $k$  条输出标记路径，详见算法4-1

---

#### 算法 4-1 BIO-Decoder 算法

---

**输入：**输出最优的路径数目  $k$ ； 标记集合大小  $T$ ； 惩罚因子  $penalty$  和序列长度  $n$ ；

**输出：**概率最大的  $k$  条输出标记路径

- 1: 初始化  $k$  个序列均为空，记为  $kbestPaths$
  - 2: **for**  $i = 1 \rightarrow n$  **do**
  - 3:   将当前时刻  $i$  的  $T$  个 slot 的候选值和已存在的  $kbestPaths$  组合形成  $k \times T$  条路径，称为  $probPaths$
  - 4:   对  $probPaths$  进行排序打分，打分时若当前标记为 I-xxx 而前一标记为 *O*，或前后标签名不匹配则施加惩罚因子  $penalty$
  - 5:   取分数最高的  $k$  个  $probPath$  作为新的  $kbestPaths$
  - 6: **end for**
  - 7: **return**  $kbestPaths$
- 

**Bi-RNN+CRF 模型** 使用2.2.4小节的 Viterbi 解码算法即可

**Focus 模型** 同样，可以直接选取解码器每一个输出阶段最大概率的标记，但可能形成局部最优但非全局最优的情况，因此我们使用 Beam Search 解码算法<sup>[13]</sup>，每个输出阶段都保留概率最大的  $k$  条输出路径，算法原理见4-2

---

#### 算法 4-2 Beam Search 解码算法

---

**输入:** 解码器网络 *Decoder*; 序列长度  $n$  和 beam size 取值  $k$ ; 标记集合大小  $T$ ;

**输出:** 概率最大的  $k$  条输出标记路径

- 1: 初始化  $k$  个序列均为空, 记为 *beamPaths*
  - 2: **for**  $i = 1 \rightarrow n$  **do**
  - 3:   将 *Decoder* 时刻  $i$  输出的所有  $T$  个候选值和已存在的 *beamPaths* 组合形成  $k \times T$  条路径, 称为 *probPaths*
  - 4:   对 *probPaths* 进行排序打分, 取分数最高的  $k$  个 *probPath* 作为新的 *beamPaths*
  - 5: **end for**
  - 6: **return** *beamPaths*
- 

#### 4.2.7.2 构建语义逻辑形式

基于知识图谱的语义理解和传统语义理解任务最大的区别在于构建语义逻辑形式时需要额外检查提取出的槽值对的合理性。构建语义逻辑形式分为两个阶段, 第一个是从输出标记中提取槽值对(slot-value pair), 第二个是根据所有识别出的槽值对和句子分类的结果构建合理的逻辑形式, 包括提取主实体(如果查询类型是关系或属性)、检查 slot 名是否是目标实体类型的关系或属性等要求。两个阶段的算法分别参见4-3和4-4。

算法4-3中 *if* 语句中几种条件的定义分别是:

**标记不匹配** 对应以下三种情况:

1. 当前标记为 I-xxx 且前一标记为 *O*
2. 当前标记为 I-xxx 且前一标记为 Byyy
3. 当前标记为 I-xxx 且前一标记为 Iyyy

**标记匹配** 只有在当前标记为 I-xxx 且前一标记为 I-xxx 或 B-xxx 时称之为标记匹配

**标记无关** 对应以下两种情况:

1. 当前标记为 *O*
2. 当前标记为 B-xxx 形式

算法4-4中挑选主实体没有固定的解决方案, 这里提供一种启发式方法:

- 查找 *slotValueList* 中 \${#xxx} 形式的 slot 且能作为 *returnType* 的主实体, 如果找到了返回该 slot, 否则跳转下一步
- 查找 *slotValueList* 中的关系且该关系对应的实体类型能作为 *returnType* 的主实体, 如果找到了, 将该 slot 重命名为 \${#xxx} 形式的 slot 后返回, 否则跳转下一步
- 如果前面两步都没有找到能作为返回类型的主实体, 将 *logicalForm* 赋值为

```
{"return_type": "nil", "mainEntity": "", "slots": []}
```

作为最终的语义逻辑形式直接返回

---

#### 算法 4-3 提取槽值对算法

---

**输入:** 序列长度  $n$ ; 原始输入序列  $\mathbf{x} = (x_1, x_2, \dots, x_n)$ ; 输出标记序列  $\mathbf{y} = (y_1, y_2, \dots, y_n)$ ;

**输出:** 所有槽值对列表  $slotValueList$

```
1: slotValueList 初始化为空列表
2: slotName,slotValue 都初始化为空字符串
3: prevSlot 初始化为  $O$  标记
4: for  $i = 1 \rightarrow n$  do
5:   if  $y_i$  和 prevSlot 标记无关 then
6:     if prevSlot 不是  $O$  then
7:       将 (slotName,slotValue) 元组添加进 slotValueList
8:     end if
9:     prevSlot =  $y_i$ 
10:    if  $y_i$  是  $O$  then
11:      slotName,slotValue 都再次初始化为空字符串
12:    else
13:      slotName,slotValue =  $y_i, x_i$ 
14:    end if
15:    else if  $y_i$  和 prevSlot 标记匹配 then
16:      将  $x_i$  拼接到 slotValue 后面
17:      prevSlot =  $y_i$ 
18:    else  $y_i$  和 prevSlot 标记不匹配
19:      if prevSlot 不是  $O$  then
20:        将 (slotName,slotValue) 元组添加进 slotValueList
21:      end if
22:      将  $y_i$  由 I-xxx 改为 B-xxx 的形式
23:      slotName,slotValue =  $y_i, x_i$ 
24:      prevSlot =  $y_i$ 
25:    end if
26:  end for
27:  if slotName 不是空字符串 then
28:    将 (slotName,slotValue) 元组添加进 slotValueList
29:  end if
30: return slotValueList
```

---

---

#### 算法 4-4 构建语义逻辑形式算法

---

输入：所有的槽值对列表  $slotValueList$ ; 句子分类结果  $returnType$ ; 领域的本体结构  $ontology$

输出：语义逻辑形式  $logicalForm$

```
1: 初始化  $logicalForm = \{“return\_type” : returnType, “mainEntity” : “”, “slots” : []\}$ 
2: if  $returnType$  是返回实体 then
3:   for  $slot \in slotValueList$  do
4:     if  $slot$  在  $ontology$  中是实体类型  $returnType$  的关系或属性名 then
5:       将  $slot$  添加进  $logicalForm[“slots”]$  列表中
6:     end if
7:   end for
8: else if  $returnType$  是  $nil$  then
9:   for  $slot \in slotValueList$  do
10:    if  $slot$  在  $ontology$  中是实体类型 then
11:      将  $slot$  添加进  $logicalForm[“slots”]$  列表中
12:    else if  $slot$  在  $ontology$  中是关系 then
13:      将  $slot$  重命名为该关系对应的实体类型名
14:      将  $slot$  添加进  $logicalForm[“slots”]$  列表中
15:    end if
16:  end for
17: else
18:   先从  $slotValueList$  中挑选一个作为主实体并赋给  $logicalForm[“mainEntity”]$ 
19:   for  $slot \in slotValueList$  do
20:     if  $slot$  在  $ontology$  中是  $logicalForm[“mainEntity”]$  的关系或属性名 then
21:       将  $slot$  添加进  $logicalForm[“slots”]$  列表中
22:     end if
23:   end for
24: end if
25: return  $logicalForm$ 
```

---

## 第五章 实验及结果分析

实验部分使用 Python 深度学习第三方开源框架 PyTorch, 工程搭建操作系统是 Ubuntu LTS 16.04, 源码与远程仓库 github 同步<sup>1</sup>

### 5.1 实验参数设置

#### 5.1.1 固定参数

基于统计的实验中，我们固定以下参数：

- 由于双向模型在各类任务中普遍更具优势，实验中始终使用双向网络模型 Bi-directional
- 根据 [11] 章节 7.12 中的建议，dropout 比例，输入单元设置为 0.2，隐层输出单元设置为 0.5
- 为了让 <UNK> 字符在训练时也能得到训练，构建词表索引时只取所有单词的前 93.9%，不在该范围内的单词用 <UNK> 代替
- 由于两个领域都比较具体（小），网络层数都设置为 1
- 验证集大小为训练集的 1/3
- 训练时损失函数使用交叉熵  $Cross\ Entropy = -E_{x \sim p_{data}} [\log p_{model}(x)]$
- 优化器选用随机梯度下降 SGD，学习率设置为 0.1
- 最大 Epoch(训练轮数) 设置为 50，早停 (Early stop) 策略的耐心值 (patience) 设置为 3，一个 mini-batch 的大小设置为 64
- Focus 模型训练时 teacher force learning(解码时直接输入参考标记) 的概率设置为 0.5

#### 5.1.2 超参数

实验中我们通过验证集选取的超参数有：

- 隐层单元大小，hidden size ∈ {256, 512}
- RNN 单元选取，cell ∈ {LSTM, GRU}
- 词向量维度大小，embedding size ∈ {128, 256}
- 训练时的 Epoch 数根据验证集早停策略的 Epoch 数决定
- Focus 模型 Pre-Attention 和 PostAttention 的选取

### 5.2 评测方法

基于 FST 的规则解析只要匹配上了模板就能解析正确，我们只需要考虑整个句子解析正确的精度 acc。

---

<sup>1</sup>[git@github.com:RhythmCao/KGNLU.git](https://github.com/RhythmCao/KGNLU.git)

基于统计的解析有着不同粒度的评价标准<sup>1</sup>，分为 charFscore、slotFscore、sentenceFscore 和整句解析正确的精度 acc。charScore 是逐字符计算每个标签（包括 O）的 f-score，这些标签会带有 B-/I- 的前缀；slotScore 是计算每种 slot 的 f-score（将 B-/I- 的标签组合起来，不考虑标签 O），将 slot 的值和参考结果比较；sentenceFscore 是计算每种返回类型的 f-score；精度 acc 则要求解析出的语义逻辑形式完全正确，这也是最终的目的。

## 5.3 AISpeech 领域实验结果及分析

### 5.3.1 FST 规则解析

根据两种方法收集到的测试数据（章节4.1.1.2），实验结果参见表5-1：

表 5-1 AISpeech 领域基于 FST 规则解析实验结果

Table 5-1 Experiment Results of FST Parser in AISpeech Domain

测试样例数	精度 acc(%)
6274	98.52
2616	85.55
合并起来 8890	94.70

第一部分 6274 句是由规则生成的基础上经由专业人员将一些拗口的说法简单修改得到，大多数的句子沿用了原始的规则语句，所以解析正确率更高；第二部分 2616 句不经过规则生成直接编写而得，更接近现实问答情况，相对来说精度就下降了很多。

### 5.3.2 神经网络统计方法解析

省略验证集对超参数的选择过程，直接给出三个模型在各自的验证集最优参数下，分别在两个部分的测试集以及合并的测试集上的结果，见表5-2、表5-3、表5-4

表 5-2 AISpeech 领域基于神经网络解析实验结果——测试集 6274 句

Table 5-2 Experiment Results of Neural Network Models in AISpeech Domain——TestSet Size 6274

model	embedding size	hidden size	cell	Epoch	slotFscore	sentenceFscore	acc
Bi-RNN	256	512	GRU	50	98.12	93.64	<b>93.43</b>
Bi-RNN+CRF	128	256	LSTM	21	99.12	97.15	<b>98.18</b>
Focus(PostAttention)	256	256	GRU	35	99.04	97.91	<b>98.29</b>

对比三种神经网络模型，将三份表格一起比较，无论是哪份测试数据集上，Focus (PostAttention) 模型都优于 Bi-RNN+CRF 模型，而 Bi-RNN+CRF 模型也都比 Bi-RNN 模型更加有效，这些都与模型网络结构的复杂程度有关，一定程度上我们可以得出结论，网络结构越复杂，能够学习到的模型空间就越大，学习到更近似于目标函数的可能性就越高。

<sup>1</sup>评价标准的定义参见小节1.2.2.2

表 5-3 AISpeech 领域基于神经网络解析实验结果——测试集 2616 句

Table 5-3 Experiment Results of Neural Network Models in AISpeech Domain——TestSet Size 2616

model	embedding size	hidden size	cell	Epoch	slotFscore	sentenceFscore	acc
Bi-RNN	256	256	GRU	50	87.10	94.12	<b>86.01</b>
Bi-RNN+CRF	128	256	LSTM	21	87.42	95.67	<b>88.15</b>
Focus(PostAttention)	256	256	GRU	35	87.58	95.92	<b>88.42</b>

表 5-4 AISpeech 领域基于神经网络解析实验结果——测试集 8890 句

Table 5-4 Experiment Results of Neural Network Models in AISpeech Domain——TestSet Size 8890

model	embedding size	hidden size	cell	Epoch	slotFscore	sentenceFscore	acc
Bi-RNN	256	256	GRU	50	91.68	94.64	<b>91.25</b>
Bi-RNN+CRF	128	256	LSTM	21	92.17	96.65	<b>95.23</b>
Focus(PostAttention)	256	256	GRU	35	92.45	96.61	<b>95.39</b>

表5-2选取的测试集来自于第一部分 6274 句，这部分测试数据和由规则生成的语句非常相似，可以用来衡量模型学习的情况。从表中可以看出，即便是表现最优的 Focus (PostAttention) 模型，在该部分的精度也不如基于规则解析的精度(分别是 98.29 和 98.52)，不过这个分数对于大量的测试数据来说是能够接受的，我们完全可以声称基于统计方法的解析器在近似由规则生成的测试数据上的学习表现达到了我们预期的效果，不比规则解析器差；

表5-3选取的测试集来自于第二部分 2616 句，该部分测试语句更接近真实的情况，可以用来衡量模型的泛化能力。从表中可以看出，即便是效果最差的 Bi-RNN 模型在该部分测试集上的精度都比规则解析器要高(分别是 86.01 和 85.55)，Focus 模型的精度更是达到了 88.42，足足高出了三个百分点，由此我们能够归纳出结论：基于统计方法的解析器在泛化能力上远远超过规则解析器，这也是统计模型相比规则模型最大的优势。将两份测试数据合并在一起，Focus 模型的精度达到了 95.36%，比规则模型高出了近 0.7 个百分点，这更进一步说明统计模型无论是学习情况还是泛化能力都取得了较理想的结果。

从训练过程和学习速度上来看，尽管 Bi-RNN+CRF 的模型初始的 Epoch Loss 很大，但它学习的速度也是最快的(斜率绝对值最大)，且 21 轮就已经收敛，loss 不再明显降低，而 Focus 和 Bi-RNN 学习曲线比较相似，收敛时的轮数也相近。此外，训练的后期，图5-2，学习速度明显减小，甚至出现 Epoch Loss 增大的情况，由此也可以看出，我们在实验中采取的正则化措施——dropout 和 early stop 还是非常有必要的，防止过拟合在训练时至关重要，并非 loss 越低越好。训练过程中三种模型 Epoch Loss 变化情况参见图5-1

以表现最好的 Focus(PostAttention) 模型为例，在测试集上各语义槽的 P-R-F1 分数、各种句子类别的 f-score 分数的柱状图分别参见图5-3和图5-4

图5-3中大多数的 slot 在 fscore 上都达到了 90 以上的分数，唯一的两个例外是 concept 和 technology 两个 slot，这两个语义槽标注的是同一类实体(比如语音识别、语义理解之类的概念或技术名词)，区别在于当这类实体在问句中做主实体时用 concept，当用作限制关系时用 technology，在句中不同的用法造成了同一个实体可能有不同的标注 slot，这从侧面体现了同一实体对应多种标注的情况会降低识别的准确率。

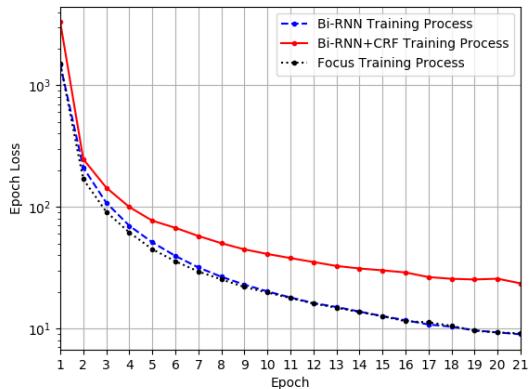


图 5-1 训练过程 Epoch Loss 变化情况

Figure 5-1 Loss Changes During Training Process

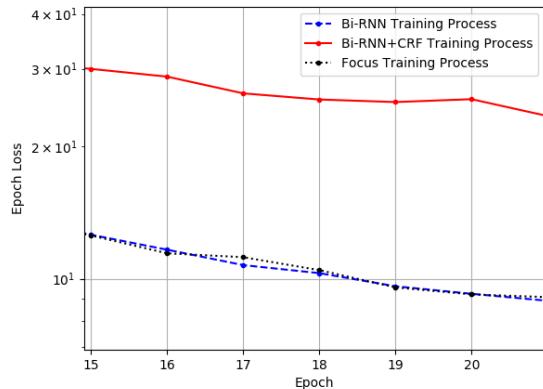


图 5-2 训练过程中 Epoch Loss 上升

Figure 5-2 Phenomenon of Increase in Epoch Loss

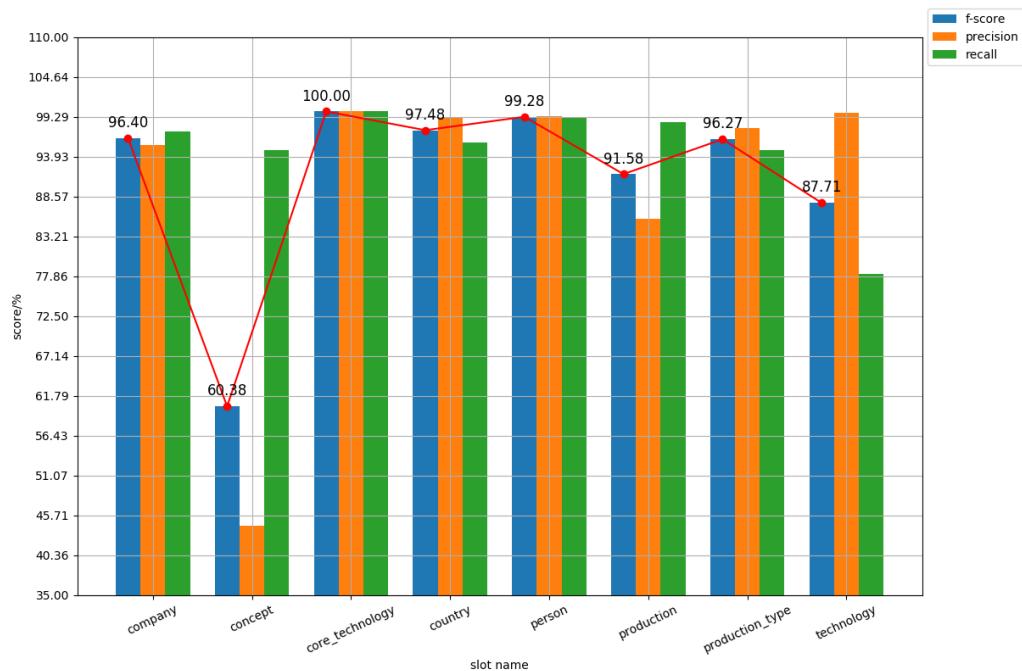


图 5-3 测试集不同 slot 的 P-R-F1 分数

Figure 5-3 Slot Fscore in TestSet

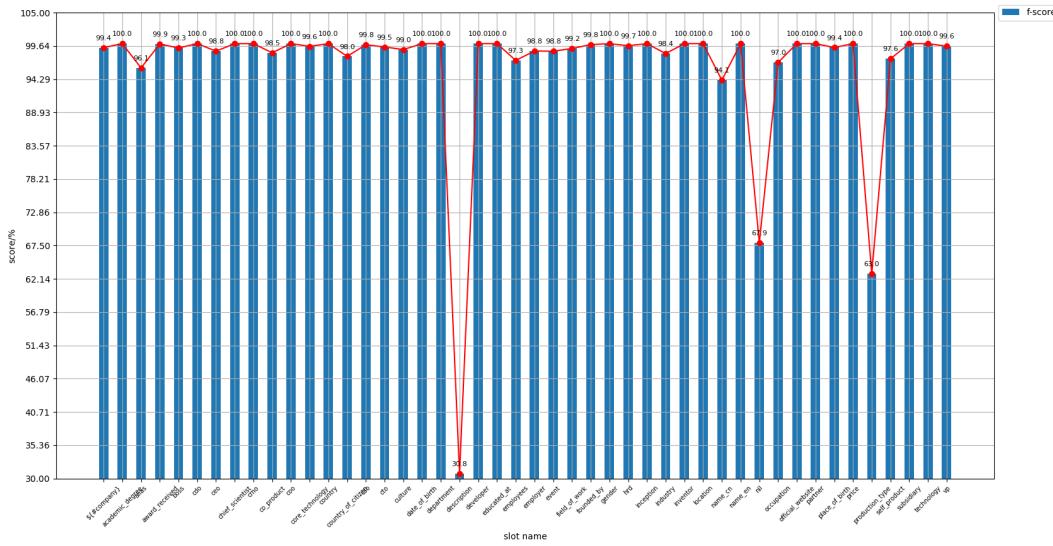


图 5-4 测试集句子分类 Fscore

Figure 5–4 Sentence Fscore in TestSet

图5-4中绝大多数句子种类(查询的返回类型)也都取得了较好的fscore, 达到了95以上的分数。分数较低的有三种返回类型, 分别是description(30.8)、nil(67.9)和production\_type(63.0)。原因分析如下:

- **description** 是所有实体都具有的一个属性(用来描述该实体),在规则模板中,关于这种返回类型的句子定义形式丰富多样,比如论文提出的模型结构、人物的生平简介、书的大概内容等,都视作查询实体的 **description** 属性,因此,很多只能识别出实体,无法确认询问意图的句子在一定程度上,模型可能“理解”为用户意图知晓对实体的介绍,这在数据上体现为 **description** 的准确率很低(44.44%),同时,由于规则模式形式多样,模型没能从中学出固定的规律,这就造成了 **description** 的召回率非常低(23.53%);
  - **nil** 分数较低的原因在于解码过程中找不到主实体的句子我们都会作为 **nil** 处理,相比其他 slot, **nil** 的准确率会比较低(很多不应做 **nil** 处理的句子解码时我们都作为 **nil** 处理了);
  - **production\_type** 分数较低的一部分原因在于对应该类别的训练数据量比较少(尽管在生成训练集时我们做了平衡类别的处理,但也只是让不同句子类别的数目不至于相差太大),另一部分原因在于产品类型在实际使用中可能呈现的词汇很多,而我们在短语规则的定义中使其能够呈现的属性值比较少,这就造成了测试集没有出现过的产品类型很难准确识别出来。

@production\_type=(智能车载|车载|机器人|智能家居|家居|智能芯片|芯片|APP|对话平台)

## 5.4 SpeechLab 领域实验结果及分析

由于 AISpeech 领域由专业人员构建，规则经过不断修改和上线测试，体系较为完善，所以统计方法的结果虽然优于基于规则的方法，但没有明显的提升（最好的模型在整个测试集上也只高出

0.66%)。为了衡量一个全新领域(没有任何先验知识和测试反馈)两种方法的结果，我们在新构建的SpeechLab领域也做了同样的实验。

### 5.4.1 FST 规则解析

根据章节4.1.2.2收集的测试数据，实验结果参见表5-5：

表 5-5 SpeechLab 领域基于 FST 规则解析实验结果

Table 5-5 Experiment Results of FST Parser in SpeechLab Domain

测试样例数	精度 acc(%)
478	39.33

新领域 SpeechLab 没有任何上线测试结果反馈的情况下，直接根据编写的规则进行 FST 解析精度非常低，甚至不到 40%。错误原因分析如下：

- 测试集语句来自最原始的模拟用户提问，形式、内容各异，且没有任何的限制
- 编写的规则涵盖率还不够广(实际上无论领域多小，由于自然语言表达的复杂性，基于规则的解析不可能覆盖所有的用语规则)
- 解析原理本质上决定的，基于规则模板的 FST 解析即使是局部匹配也需要连续的一段字符串完全匹配上(不能跳跃式解析)，比如询问年龄张三多大了，若是在张三后加入一个语气词(实际问答中很可能出现)，张三额今年多大了，规则模板几乎不可能覆盖这样的句子，就解析不出来
- 实体和属性限制条件都来源于预定义的词库和短语库，一旦出现没有定义的实体或限制条件，就解析不出来，甚至只需要对原来的实体名做一点修改，比如词库中一篇论文名字很长，*What does the speaker embedding encode*，用户偷懒用 *speaker embedding* 来指代该论文，词库中没有这个实体，就不能识别出来

由此可见，基于 FST 的规则解析虽然有一套规范的流程和实现系统，但面对一个新领域都需要经过精细的规则编写和反复上线测试并完善这些规则，耗时耗力，还需要丰富的经验、领域知识、语言学知识，才能实现一个完善的解析系统，而这些都不能在短时间之内实现。

### 5.4.2 神经网络统计方法解析

省略验证集对超参数的选择过程，直接给出三个模型在各自的验证集最优参数下，于测试集上的结果，表5-6

表 5-6 SpeechLab 领域基于神经网络解析实验结果

Table 5-6 Experiment Results of Neural Network Models in SpeechLab Domain

model	embedding size	hidden size	cell	Epoch	charFscore	slotFscore	sentenceScore	acc
Bi-RNN	256	512	GRU	42	83.23	75.89	80.07	<b>69.04</b>
Bi-RNN+CRF	256	256	LSTM	15	87.97	82.98	80.44	<b>71.13</b>
Focus(PreAttention)	256	256	LSTM	37	78.54	75.87	82.19	<b>74.06</b>

三种模型中 Focus 模型的精度比 Bi-RNN+CRF 模型高出了近 3 个百分点，Bi-RNN+CRF 模型比 Bi-RNN 模型高出了 2 个百分点。由此可见，Bi-RNN+CRF 由于考虑了额外的特征(输出标记之间的转移概率)，依旧明显优于简单的 Bi-RNN 模型；而 Focus 模型相比 Bi-RNN+CRF 模型网络结构更复杂，挖掘出的隐藏规则也更多，在测试集上表现也更好。从收敛速度上说，Bi-RNN+CRF 依旧学习得最快，15 轮后就收敛了，其次是 Focus 模型，而 Bi-RNN 模型学习收敛最慢，花了 42 轮。

通过分析解析错误的句子，我们总结出以下几条出错原因：

- 尽管统计模型能够识别出许多训练集没有出现过的实体或属性限制，训练集词库和短语库没有对应的实体或属性限制依旧是出错最主要的原因之一。SpeechLab 针对的是上海交通大学智能语音实验室的问答，但由于给了改写者内容上很大的发挥空间，收集到的测试数据中出现贝尔实验室、科创中心之类训练集没有出现过的实体

问句	科创中心发表过的文章都是哪一些
参考标注	{"return_type": "papers", "mainEntity": "科创中心", "slots": [{"op": "eq", "name": "\${#institute}", "value": "科创中心"}]}
解析结果	{"return_type": "publication", "mainEntity": "科创", "slots": [{"op": "eq", "name": "\${#person}", "value": "科创"}]}

- 由于没进行数据清洗，有些测试集的语句由于改写者没有很好的理解原本句子的询问意图，错误地改写造成标注和句子之间不匹配，比如测试样例中有一句你们出国比例，但我们的知识图谱里并没有关于这条问句的关系或属性名，说明改写者错误理解原来句子的询问意图，过度发挥想象

问句	你们出国比例
参考标注	{"return_type": "member", "mainEntity": "智能语音实验室", "slots": [{"op": "eq", "name": "\${#institute}", "value": "智能语音实验室"}, {"op": "eq", "name": "country_of_citizen", "value": "出国"}]}
解析结果	{"return_type": "member", "mainEntity": "智能语音实验室", "slots": [{"op": "eq", "name": "\${#institute}", "value": "智能语音实验室"}, {"op": "eq", "name": "place_of_birth", "value": "国比例"}]}

- 相似的规则对 slot 误分类，比如错把安徽识别为国家

问句	你们实验室里有哪些人来自安徽
参考标注	{"return_type": "member", "mainEntity": "智能语音实验室", "slots": [{"op": "eq", "name": "\${#institute}", "value": "智能语音实验室"}, {"op": "eq", "name": "place_of_birth", "value": "安徽"}]}
解析结果	{"return_type": "member", "mainEntity": "智能语音实验室", "slots": [{"op": "eq", "name": "\${#institute}", "value": "智能语音实验室"}, {"op": "eq", "name": "country_of_citizen", "value": "安徽"}]}

- 本应识别出来的句子识别错误，可能是句子分类错误或 slot 没识别出来，这种现象不可避免，即便在验证集（由训练集划分得到）上解析也不是百分百正确

问句	你们实验室有哪些在上海交通大学读书的成员
参考标注	{"return_type": "member", "mainEntity": "智能语音实验室", "slots": [{"op": "eq", "name": "\${#institute}", "value": "智能语音实验室"}, {"op": "eq", "name": "educated_at", "value": "上海交通大学"}]}
解析结果	{"return_type": "member", "mainEntity": "智能语音实验室", "slots": [{"op": "eq", "name": "\${#institute}", "value": "智能语音实验室"}]}

- 改写句子的人出现错别字，测试语句中出现一条那几本是实验室的代表性书籍的问句，哪错用成那，网络没能理解这种错误（这类错误对模型纠错能力和鲁棒性要求较高）

问句	那几本是实验室的代表性书籍
参考标注	{"return_type": "books", "mainEntity": "智能语音实验室", "slots": [{"op": "eq", "name": "\${#institute}", "value": "智能语音实验室"}]}
解析结果	{"return_type": "books", "mainEntity": "那几本", "slots": [{"op": "eq", "name": "\${#institute}", "value": "那几本"}]}

尽管有上述几点错误，对于 SpeechLab 这一短时间构建起来的全新问答领域来说，统计模型不仅在验证集上能达到 99.97%<sup>1</sup> 的精度（学习能力很强），在测试集上的表现也明显优于规则 FST 模型（74.06% 相比于 39.33%，精度近乎 2 倍），更进一步说明统计模型泛化能力远远大于规则模型。这种泛化能力主要体现在对训练集未出现过的实体、属性限制和规则模式的识别上，不是僵硬地套用模板，而是挖掘潜在的规律。

## 5.5 实验总结分析

本论文的多个实验中，我们基于两个领域，AISpeech 和 SpeechLab，尝试了两种方法，基于 FST 的规则解析和基于神经网络的统计解析，在统计方法中实验了 Bi-RNN、Bi-RNN+CRF、Focus 模型三种框架。AISpeech 领域是一个规则较完善的领域，而 SpeechLab 领域是针对上海交通大学智能语音实验室在短时间构建的新领域，规则体系尚未经过测试和检验。在规则解析方法中，对于规则较完善的 AISpeech 领域达到了上线使用的要求，而在 SpeechLab 这一冷启动问题上，表现却很差，有待进一步对规则的完善修改；在统计方法中，三种模型的实验结果符合模型结构的复杂程度，无论是哪个领域，Focus 模型优于 Bi-RNN+CRF 模型，Bi-RNN+CRF 优于 Bi-RNN 模型，在 AISpeech 领域，统计模型就初步呈现出较强的学习能力和优于规则解析器的泛化推理能力，在 SpeechLab 领域更是进一步体现了它在泛化能力上的优势，对于形式内容各异的、训练预料或规则模板没有覆盖到的问句，解析的正确率远远高于规则模型。

此外，综合两个领域多种网络模型，在超参数的选择，LSTM/GRU 单元、Focus 模型 Pre-Attention 和 Post-Attention，并没有统一的哪一种选择更优的结论。比如，在 AISpeech 领域 Focus 模型 PostAt-

<sup>1</sup>根据验证集调参数据太多，实验表格中就没有给出

tention 方法表现更突出，而在 SpeechLab 领域 Pre-Attention 模型比 PostAttention 表现更突出，需要结合具体领域、具体模型分析。而对于词向量维度和隐层单元的大小，也并非参数越大模型表现就越好，由此可见，某一参数的选择在多个领域之间没有明显优劣的情况下，验证集对于超参数的选择还是起着一定的优化作用。

尽管统计模型在实验中无论是学习能力还是泛化能力都优于规则解析器，但并不意味着统计解析器目前就能完全替代规则解析器。3.1.2小节介绍的递归解析问题，基于规则的解析可以非常容易地解决，而基于统计方法的解析器由于神经网络的扁平化输出结构目前很难实现这一功能。神经网络结构化的语义输出表示问题一直是深度学习应用于语义理解领域的一大难点。

## 第六章 总结及未来工作

### 6.1 研究总结

本文针对知识图谱问答中的语义理解问题进行了研究，最主要的贡献是提出了一种新颖的、易于理解的逻辑形式（或语义表达式），构建了相关的基于神经网络的统计方法模型，并对比分析了统计模型和规则模型的实验结果。在学习情况上统计模型表现良好，并且在泛化能力上比规则模型的基线系统有着较大的提升。

本文首先介绍了研究课题的背景，指出知识图谱的提出和发展为语义理解领域提供了丰富的背景知识并且已经逐渐发展成为未来智能问答系统的趋势。近几年深度学习技术的发展更是为这一领域提供了多种多样的解决思路。接着，简要论述了知识图谱本体结构的定义和语义理解的核心——语义表达方式，并给出了常用的评价标准。第二部分，本文介绍了相关的背景知识，简要地说明了有限状态转移机的定义和常用操作，条件随机场在序列标注问题上的运用、神经网络的基础知识以及流行的 Sequence-to-Sequence、Attention 模型结构，本文使用的方法和模型都与之紧密关联。

本文的第三部分介绍了一套自定义的规则、逻辑形式以及基于 FST 的规则解析原理。总结归纳了常用的提问类型作为我们的任务目标，并且分条陈述了基于知识图谱的语义理解任务的几大难点，针对这些难点，我们详细论述了在实验中我们定义并使用的语义逻辑形式。在附录中更是一步一步抽丝剥茧般给出了利用 FST 进行解析的具体例子。论文的第四部分我们先是详细介绍了数据集的来源渠道，包括训练集生成和测试集的收集工作。接着逐步深入地阐述了统计实验中我们所使用的神经网络模型，包括 Bi-RNN、Bi-RNN+CRF 和 Attention 模型的变体 Focus 模型。为了实现网络的输出到逻辑形式的转换，我们还具体介绍了逻辑形式的扁平化表示以及解码过程的诸多算法。

在正文第五部分，我们根据两个领域，从两种方法、多个模型，具体地分析了实验的结果，以及产生这些结果的原因。在规则较为完善的 AISpeech 领域，统计模型的效果就已经略优于规则模型，为了进一步验证统计模型的泛化能力，我们实验室内部还自行构建了一个全新的 SpeechLab 领域，包括知识图谱的本体结构、规则模板以及测试数据。实验结果表明，统计模型在识别模板没有覆盖到的规则和实体、属性限制方面具有独特的优势，泛化性能远远优于规则系统。不过规则系统在递归解析方面却暂时无法通过扁平化的逻辑形式被替代。

总而言之，本文在总结前人研究经验的基础上，尝试了规则解析的框架并构建了崭新的逻辑形式下的神经网络统计语义理解模型，统计模型实验结果大幅超出了基线规则系统。而且统计方法的模型还具有快速适应性和灵活可扩展性，且有着较大的提升空间，有待进一步的研究。

### 6.2 成果展示

为了更好地检验我们的工作成果，我通过爬虫爬取部分实验室的内部数据并生成 RDF 资源文件，简单构建了一个远程知识图谱的数据库。基于我们语义理解的工作，以及思必驰公司员工技术平台上的支持，实现了 SJTUSpeechLab 智能问答的微信公众号，问答成果展示参见图6-1

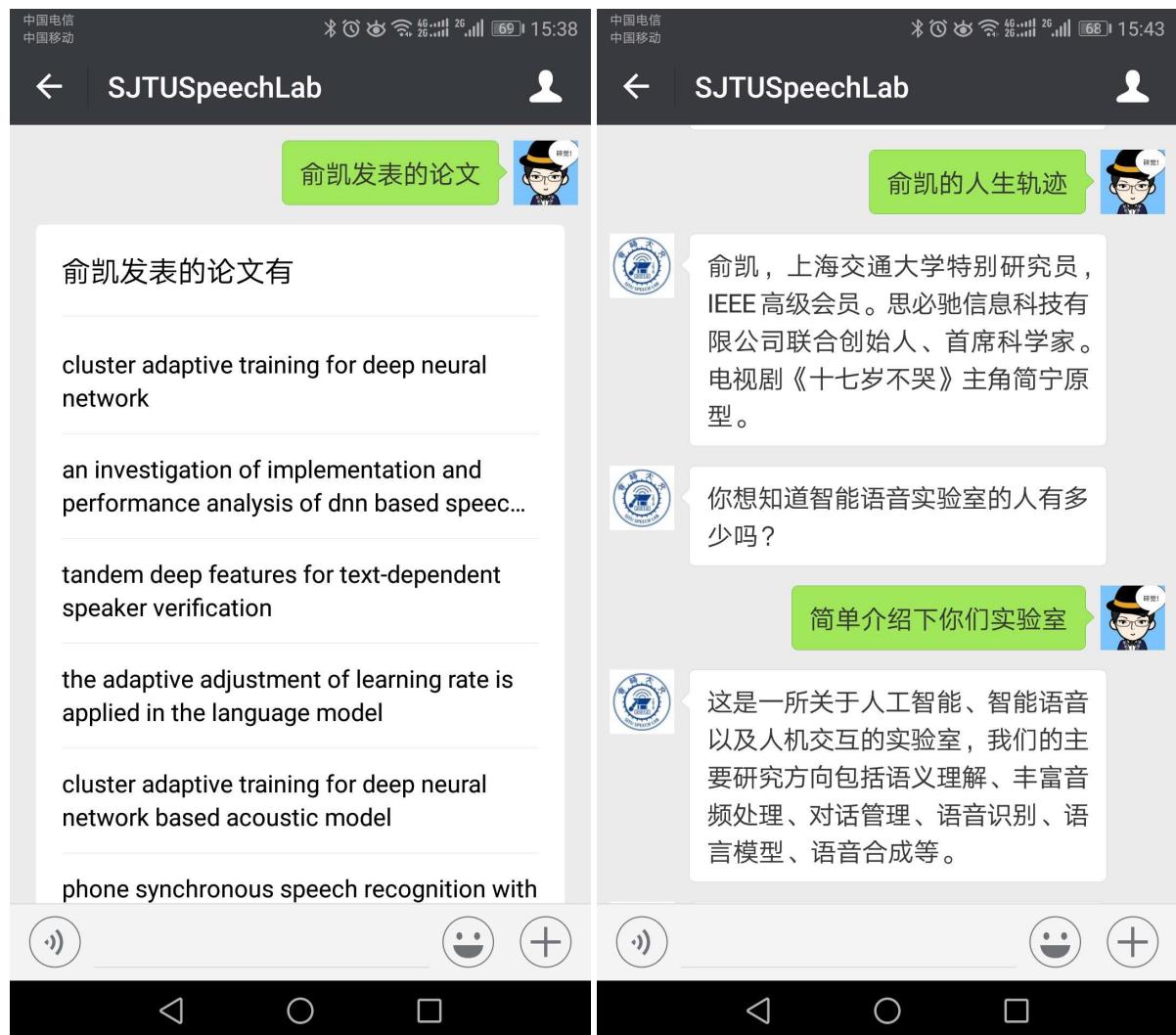


图 6-1 SJTUSpeechLab 微信公众号智能问答系统展示  
Figure 6-1 SJTUSpeechLab Question Answering System Demo

## 6.3 未来工作展望

基于本文统计模型在两个领域的优异表现，神经网络的方法在知识图谱问答中体现出极大的研究价值和发展空间。未来还有很多值得挖掘、深入研究的空间，在本校读研期间，我计划从以下几个角度入手：

**端到端生成** 从神经网络的输出到语义逻辑形式的转换要经过复杂的解码过程，为此可以探索一种直接由网络生成语义逻辑形式的端到端生成方式，省去繁重且容易出错的语义逻辑形式转换过程

**递归解析** 规则系统相比于统计模型唯一的优势在于递归解析上，实现层次上的多轮解析，针对多关系问句的语义解析是未来基于知识图谱的语义理解任务的重中之重。从本质上看，无论是端到端方法还是递归解析，都是为了解决结构化的语义输出这一根本问题，语义表达式的复杂结构以及不同成分之间相互依赖关系是最主要的羁绊，为此我尚需要调研其他任务更多更复杂的网络框架

**半监督学习** 当面临一个新领域时总要面对冷启动的问题，无论是训练数据的生成还是测试数据的收集都需要耗费极大的人力，对训练语句的标注任务更是雪上加霜，学习如何从少量标注语料和大量无标注语料中学习到一个可以使用的模型也是未来工作之一

**领域自适应** 当基于知识图谱的语义理解在某一具体领域能取得较为理想的结果时，我们常常考虑系统在不同领域之间的鲁棒性和领域之间学习能力，这也是迁移学习的一种应用

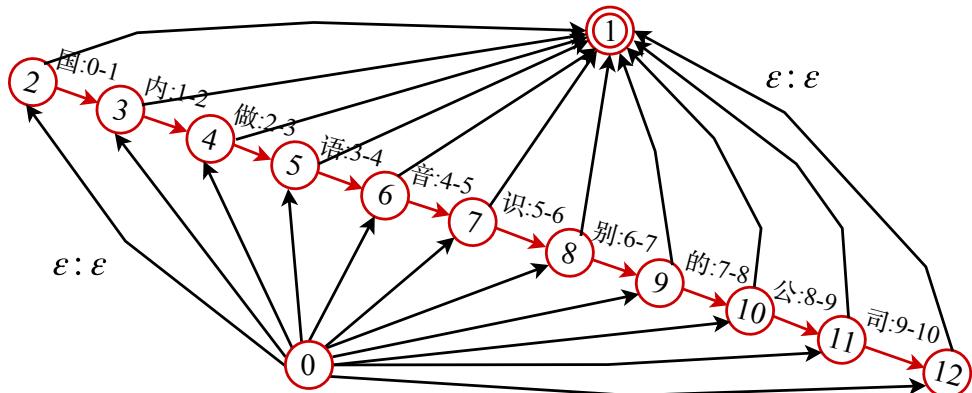
## 附录 A 基于 FST 的规则解析过程详解

输入问句：国内做语音识别的公司

1. 构建输入句子的有限状态机 FST\_in，  
边上转移条件的输入是句子中的字，输出是对应字的位置

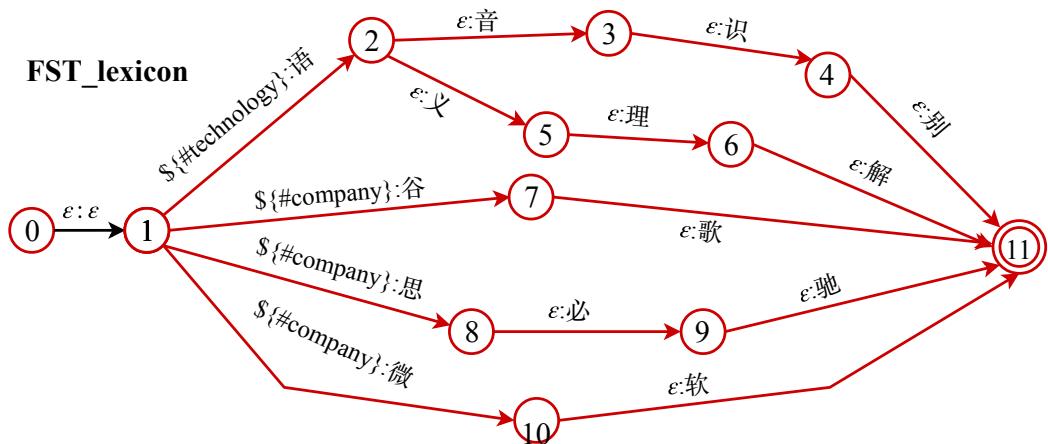


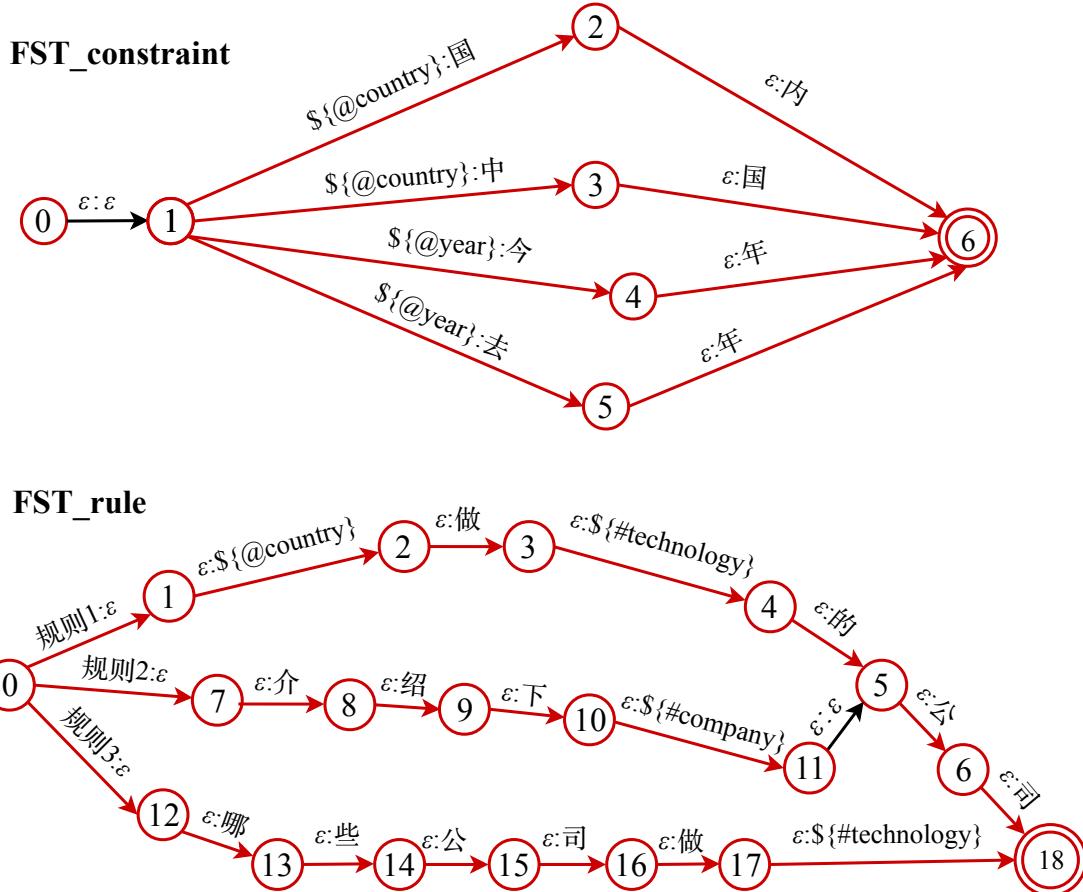
2. 将输入句子的有限状态机 FST\_in 转化为待匹配的有限状态机 FST\_to\_be\_replaced，  
黑色的边均是无条件转移边，输入和输出符号都是  $\epsilon$



### 步骤一 FST资源准备

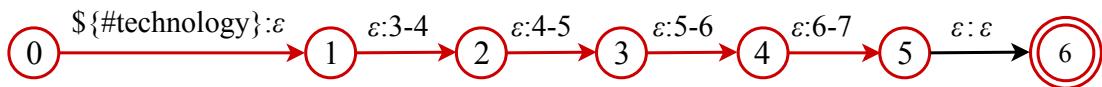
3. 对词法规则、短语规则和句法规则分别构建有限状态机  
分别记作 FST\_lexicon, FST\_constraint, FST\_rule



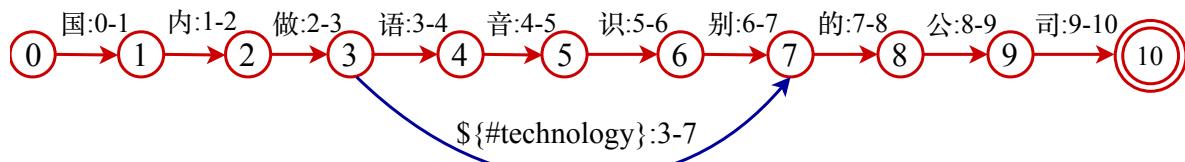


## 步骤二 匹配词法规则

4. FST\_lexicon和FST\_to\_be\_replaced作compose(组合)操作，得到的FST是所有匹配的实体在输入语句中的位置，记作Matched\_lexicon。这里的例子中匹配上实体：语音识别



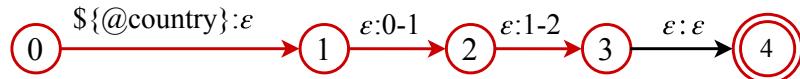
5. 根据Matched\_lexicon，对FST\_in添加新边，得到新的FST\_in



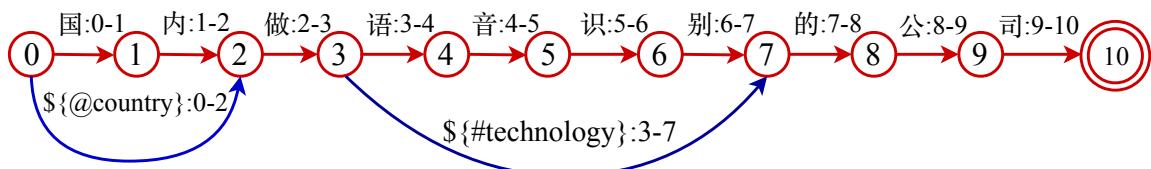
同理，`FST_to_be_replaced`也添加新边，得到新的`FST_to_be_replaced`（图略）

### 步骤三 匹配短语规则

6. FST\_constraint和步骤二后得到的FST\_to\_be\_replaced作compose(组合)操作，得到的FST是所有匹配的限制条件在输入语句中的位置，记作Matched\_constraint。这里的例子中匹配上属性限制：{@country=国内}



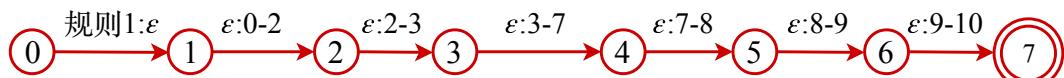
7. 根据Matched\_constraint，对步骤二后得到的FST\_in添加新边，得到新的FST\_in



同理，对步骤二得到的FST\_to\_be\_replaced也添加新的边，得到新的FST\_to\_be\_replaced

### 步骤四 匹配句法规则

7. FST\_rule和步骤三后得到的FST\_to\_be\_replaced作compose(组合)操作，得到的FST是所有能够匹配的规则在输入语句中的起始位置，记作Matched\_rule。例子中只匹配上规则1: \${@country} 做 \${#technology}" 的公司，匹配上原语句的位置是0-10，即全局匹配



8. 由FST Matched\_rule输出所有的路径，就是所有匹配的规则，从中筛选最优规则作为我们中间语义表示的来源(这里例子中只匹配了规则1，所以只有1条输出路径，实际筛选时方法很多，最简单的做法是挑选匹配原输入语句最长的规则)

至此，基于FST的一轮规则解析完成，若是原输入语句中还有内容没有匹配上(局部匹配)，可以将本轮解析匹配上的部分用对应的返回实体替换，继续递归解析。

## 附录 B SpeechLab 领域本体结构设计

	概念	7种	红色标记
	属性	33种	蓝色标记
	关系	14种	绿色标记
机构 (实验室)			
	"institute": { "type": "string", "name": "string", "description": "string", "detail": "string", "ambition": "string", "inception": "string", "location": "string", "country": "string", "employees": "string", "award": "string", "culture": "string", "official_website": "string", "image": "string", "direction": "concept", "member": "person", "books": "book", "papers": "paper", "patents": "patent", "demos": "demo" },		
人物	"person": { "type": "string", "name": "string", "gender": "string", "country_of_citizen": "string", "year_of_join": "int", "identity": "string", "email": "string", "award": "string", "educated_at": "string", "academic_degree": "string", "occupation": "string", "description": "string", "detail": "string", "personal_website": "string", "image": "string", "date_of_birth": "string", "place_of_birth": "string", "field_of_work": "concept", "publication": "paper" },		
研究方向	"concept": { "type": "string", "name": "string", "description": "string", "detail": "string", "inception": "string", "link": "string", "image": "string", "persons": "person", "publication": "paper" },		
	"demo": { "type": "string", "name": "string", "description": "string", "detail": "string", "production_type": "string", "inception": "string", "link": "string", "technology": "concept" },		产品成果
	"paper": { "type": "string", "name": "string", "title": "string", "abstract": "string", "publication_date": "string", "published_in": "string", "link": "string", "award": "string", "institute_place": "institute", "author": "person", "first_author": "person" },		论文
	"book": { "type": "string", "contribution": "string", "name": "string", "title": "string", "publication_date": "string", "publisher": "string", "link": "string", "author": "person", "translator": "person" },		出版书籍
	"patent": { "type": "string", "name": "string", "description": "string", "detail": "string", "patent_id": "string", "technology": "concept", "author": "person" }		专利

## 附录 C SpeechLab 解析规则示例

宏定义
whois=(谁   哪一? 位   哪(一 几)? 个人?   什么人   哪(些 群)人?   哪一人)
whetherhas=(有   有没有   有木有   啊有   是不是有   有吗   有无)
haswhat=(都?(是 有)(哪(几)? 个   哪(一)? 些   些? 什么   些? 啥   多少))
whattime=(哪一? 年   哪一? 天   (什么   啥)时候   (什么   啥)时间   何时   何年)
规则模板
(\${#institute}) 的? (官方的?)? (网址   网络   页   站)(地址   链接)   链接   网页) (是什么   是啥   在? 哪儿   里) 有?)? =>((#1,official_website,?),[ ])
(研究   搞   从事   做) (\${#concept}) (课题   方向   主题   研究   领域)? 的 (实验室   lab) (\${haswhat})? => ((institute),[(eq,direction,#1)])
(\${#person}) 的? (研究方向   科研方向   方向   科研兴趣   研究兴趣   研究领域   研究课题   课题) (\${haswhat})? => ((#1,field_of_work,?),[ ])
(\${#person}) (\${whetherhas} \${haswhat})  都?(有几篇   都?(有   是) 哪(些   几   一)? 篇)  都? 有多少篇) (刊登   发表?   投中   撰写   写   投) (在   于) (\${@published_in}) (会议   上?   期刊上?)? 的? (论文   文章   文献   paper) =>((#1,publication,?),[(eq,published_in,@1)])
(\${#paper} ((这一?(篇   个))?(文献   文章   论文   学术成果   paper))? (主要   具体   大概   大致)? (讲   谈   提出   讨论   探讨   研究   分析   解决   针对   调研) (了   的是)? 什么 ((新颖   全新) 的?)? (内容   观点   想法   方案   方法   问题   思路   难题   困难))? => ((#1,abstract,?),[ ])
(撰写   编   写   完成)? (\${#book}) (这一?(本   部)(书   著作   作品   经典(作品)?))? 的? (原创作者   原创   作者   原作) 是? (\${whois})? => ((#1,author,?),[ ])
(\${#demo}) ((这(一   一? 个   一? 项   一? 款))?(demo   产品))? (具体   大概   大致)? 是? (\${whattime}) (发布   发行   公开   推出   推行   推广   发售) 的? =>((#1,inception,?),[ ])
(\${#patent}) ((这(一   一? 个   一? 项))?(专利   发明)?   成果   发明专利)? (具体   主要)? 是? 都? (应用   使用   运用   涉及   关于) 了? (哪(一   些)?(方面   方向   领域) 的?   什么   啥子?   哪一? 些   哪一? 项) (技术   知识)? => ((#1,technology,?),[ ])
(\${#patent}) (究竟   到底   具体   主要)? (能够   能   做   干) 些?(什么   啥)   有?(什么   啥子?) 用   (怎么   如何   怎么   样) (操作   使   用   实现) 的? ) =>((#1,description,?),[ ])
(\${#institute} \${#person} \${#paper} \${#book} \${#demo} \${#patent} \${#concept}) => ((nil),[ ])

表 C-1 SpeechLab 领域规则示例

Table C-1 Examples of Semantic Parsing Rules in SpeechLab Domain

## 附录 D SpeechLab 问卷及测试数据收集样例

问卷样例，图D-1和D-2：

测试数据回收样例，表D-1：

简单地介绍一下实验室的情况
你们出国比例
隔壁 e-learning 实验室都有啥成员是 2017 年加入的
和语音识别技术相关专利有哪几个
13 年俞凯论文的 list
你们实验室 2017 年发布了哪些 demo
你们实验室的 big picture 是什么
语音实验室的实验室文化是什么
2018 年新加入到实验室的科研助理是谁
都有哪几个实验室做语音识别方向
论文 what does the speaker embedding encode 是哪个单位发表的
请详细描述一下语义方面的知识
有关丰富音频处理的文章有哪些
leaning deep architectures for ai 是谁翻译的
数字串纠正装置专利主要使用了哪些技术
我在哪个网站可以下载到 speech recognition 这本书
2015 年出版的你们的书有哪些
你们实验室毕业于西安交通大学的人有吗
你们实验室是什么时候建立的
你们总共几个人
智能语音实验室里都有哪些老师
你们实验室在 2018 年 icassp 上发表的论文有哪些

表 D-1 SpeechLab 领域测试数据收集样例

Table D-1 Examples of Test Set in SpeechLab Domain



## 知识图谱问答中的语义理解测试数据收集

本问卷用来采集针对“智能语音实验室”提问的测试数据及标注，请根据引导语的询问意图，改写上下文的问法（换一个说法）并填写你使用的语义槽(slot)值

备注：slot可以理解为句子中承载语义信息的key word，比如“有没有北京到上海的火车票”，关键信息为出发城市="北京"、到达城市="上海"、交通工具="火车"

### 注意事项：

- 1.引导语中由 => 引出的是slot信息，问句中非slot部分是上下文
- 2.请确保slot填空框中填写的内容和你改写的语句中使用的值完全一致(不区分大小写)，比如你在问句中使用的是“俞老板”，slot填写时请填写“俞老板”而非“俞凯”(如果完全没有改变引导语中的slot值可以不填)
- 3.改写时任何一个slot的信息不能丢失，比如询问“实验室的本科生”，错误改写为“你们实验室有哪些人”缺少身份(identity=本科生)信息
- 4.所有提问针对的是我们智能语音实验室，你们、你们那儿等都是指的我们实验室

**特别提醒：**腾讯问卷没有缓存功能，退出后不能保存，建议网页版填写

### 样例1：

引导语：俞凯去年发表了什么论文 => \${#person}=俞凯, publication\_date=去年

改写为：2016年钱教授发了哪些论文

slot: \${#person}=钱教授

slot: publication\_date=2016年

样例2：引导语：实验室本科生有哪些 => \${#institute}=实验室, identity=本科生

改写为：智能语音实验室有哪些博士

slot: \${#institute}=智能语音实验室

slot: identity=博士

样例3：引导语：Speaker Verification with Deep Features什么时候发表的 => \${paper}=Speaker Verification with Deep Features

改写为：Speaker Verification with Deep Features的发表时间是哪年

slot: \${#paper}=\_\_\_\_\_ (没有改动引导语的slot值可以不填)

1.引导语：语音实验室里面都是什么2012年才加入本科生 => \${#institute}=语音实验室, year\_of\_join=2012年, identity=本科生 [多选题] \*

改写为：\_\_\_\_\_

slot: \${#institute}=\_\_\_\_\_

slot: year\_of\_join=\_\_\_\_\_

slot: identity=\_\_\_\_\_

图 D-1 SpeechLab 领域测试数据收集问卷样例 1

Figure D-1 Examples of Questionnaire Part 1

2.引导语：跟语义技术相关发明专利都是什么 => technology=语义 [多选题] \*

改写为：\_\_\_\_\_

slot: technology=\_\_\_\_\_

3.引导语：智能语音实验室都有哪些paper => \${#institute}=智能语音实验室 [多选题] \*

改写为：\_\_\_\_\_

slot: \${#institute}=\_\_\_\_\_

4.引导语：关于思必驰纠错哥这款产品各方面阐述 => \${#demo}=思必驰纠错哥 [多选题] \*

改写为：\_\_\_\_\_

slot: \${#demo}=\_\_\_\_\_

5.引导语：谭天有些什么cnccl发表了的paper => \${#person}=谭天, published\_in=cnccl [多选题] \*

改写为：\_\_\_\_\_

slot: \${#person}=\_\_\_\_\_

slot: published\_in=\_\_\_\_\_

6.引导语：你们那块的内部照片 => \${#institute}=你们那块 [多选题] \*

改写为：\_\_\_\_\_

slot: \${#institute}=\_\_\_\_\_

7.引导语：发明了数字串纠正方法这个专利的人是谁 => \${#patent}=数字串纠正方法 [多选题] \*

改写为：\_\_\_\_\_

slot: \${#patent}=\_\_\_\_\_

图 D-2 SpeechLab 领域测试数据收集问卷样例 2

Figure D-2 Examples of Questionnaire Part 2

## 参考文献

- [1] WANG Y Y, DENG L, ACERO A. Spoken language understanding[J]. IEEE Signal Processing Magazine, 2005, 22(5): 16-31.
- [2] TRAUM D R. Speech acts for dialogue agents[G]// Foundations of rational agency. [S.l.]: Springer, 1999: 169-201.
- [3] YOUNG S. CUED standard dialogue acts[J]. Report, Cambridge University Engineering Department, 14th October, 2007, 2007.
- [4] THOMSON B. Statistical methods for spoken dialogue management[M]. [S.l.]: Springer Science & Business Media, 2013.
- [5] LIANG P. Lambda dependency-based compositional semantics[J]. ArXiv preprint arXiv:1309.4408, 2013.
- [6] 周志华. 机器学习[M]. [出版地不详]: Qing hua da xue chu ban she, 2016.
- [7] BERANT J, CHOU A, FROSTIG R, et al. Semantic parsing on freebase from question-answer pairs[C]// Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. [S.l.]: [s.n.], 2013: 1533-1544.
- [8] CHEN B, SUN L, HAN X, et al. Sentence rewriting for semantic parsing[C]// Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vol. 1. [S.l.]: [s.n.], 2016: 766-777.
- [9] YAO X, VAN DURME B. Information extraction over structured data: Question answering with freebase[C]// Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vol. 1. [S.l.]: [s.n.], 2014: 956-966.
- [10] BORDES A, CHOPRA S, WESTON J. Question answering with subgraph embeddings[J]. ArXiv preprint arXiv:1406.3676, 2014.
- [11] GOODFELLOW I, BENGIO Y, COURVILLE A, et al. Deep learning[M]. Vol. 1. [S.l.]: MIT press Cambridge, 2016.
- [12] CHO K, VAN MERRIËNBOER B, GULCEHRE C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. ArXiv preprint arXiv:1406.1078, 2014.
- [13] SUTSKEVER I, VINYALS O, LE Q V. Sequence to sequence learning with neural networks[C]// Advances in neural information processing systems. [S.l.]: [s.n.], 2014: 3104-3112.
- [14] BAHDANAU D, CHO K, BENGIO Y. Neural machine translation by jointly learning to align and translate[J]. ArXiv preprint arXiv:1409.0473, 2014.

- [15] WANG Y, BERANT J, LIANG P. Building a semantic parser overnight[C]// Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Vol. 1. [S.l.]: [s.n.], 2015: 1332-1342.
- [16] MESNIL G, DAUPHIN Y, YAO K, et al. Using recurrent neural networks for slot filling in spoken language understanding[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2015, 23(3): 530-539.
- [17] YAO K, PENG B, ZHANG Y, et al. Spoken language understanding using long short-term memory neural networks[C]// Spoken Language Technology Workshop (SLT), 2014 IEEE. IEEE. [S.l.]: [s.n.], 2014: 189-194.
- [18] HUANG Z, XU W, YU K. Bidirectional LSTM-CRF models for sequence tagging[J]. ArXiv preprint arXiv:1508.01991, 2015.
- [19] LUONG M T, PHAM H, MANNING C D. Effective approaches to attention-based neural machine translation[J]. ArXiv preprint arXiv:1508.04025, 2015.

## 致 谢

感谢上海交通大学和致远工科荣誉计划项目四年以来对我的教育和资助，为我的学业提供了一个科研资源丰富的环境。尤其感谢俞凯教授指引我来到智能语音实验室，这里严谨的学术氛围、一丝不苟的科研作风和优越的设备条件，让我总能够在徘徊无助的时候得到第一时间的排忧解惑。俞教授独特的人格魅力更是令我如沐春风，很早便确定了自己未来的研究兴趣及方向——自然语言处理，深刻认识到自己的不足，每一天都如雨中春笋般迫不及待地汲取知识的甘霖，即便是枯燥的论文阅读和繁重的科研实验也一样乐此不疲。其次我要感谢实验室的朱苏、陈露、张慧峰等学长在我思路受阻、无计可施的情况下，一直耐心地用他们丰富的经验知识指引我开辟新的道路，作出新的尝试。然后我还要感谢实验室的诸位同学，在我需要收集测试数据时无偿贡献出自己宝贵的时间，认真进行的标注工作，是你们精细的工作让我顺利完成了实验。特别感谢的那一群每日一同起早贪黑地学习生活、即将一起在交大攻读硕士和博士学位的同学，是你们的陪伴让我的生活更添一份色彩。此外，我还要感谢为我的毕设论文提供模板的交大物理系同学，为我省去了很多繁琐的排版工作。

最感谢的还是父母这四年来自我经济和精神上的支持，在我面临诸多人生重要的抉择时始终无条件地站在我这一边，积极鼓励我勇敢抉择。

## NATURAL LANGUAGE UNDERSTANDING IN QUESTION ANSWERING SYSTEM BASED ON KNOWLEDGE GRAPH

With the development of artificial intelligence and the prevalence of Deep Learning, services provided in automobile, robotics, home, and customer service sectors have become increasingly intelligent. The smart customer service system is based on a large-scale knowledge base and uses Automatic Question Answering System as back-end support. It combines knowledge fusion, natural language understanding, dialogue management, and text generation technologies together to provide customers with specialized service and answers. Intelligent customer service helps companies greatly reduce the investment on human labor in call centers. Sundar Pichai, CEO of the Google, presented a scene in which Google Assistant made a phone call appointment for haircut on Google I/O Developer Conference in 2018. The personalized services and specialized knowledge of smart customer service have begun to gradually replace the company's repetitive question and answering work.

In May 2012, Google spent a lot of money to acquire company Metaweb, and for the first time it put forward the concept of Knowledge Graph. Since then, large-scale Knowledge Graph of various open domains, such as NELL, FreeBase, and DBpedia, has appeared. Knowledge Graph is essentially a semantic network. Nodes in the network represent entities or concepts. Edges represent semantic relationships between entities or concepts. These relationships can be attributes or relations. Knowledge is managed through the relationship between entities or concepts. This network-structured knowledge management approach fully embodies the semantic relationships between entities and is ideally suited to automatic Question Answering applications which have a strong dependence on semantic relevance. Question Answering System based on Knowledge Graph is used as a new type of Question Answering System. The use of information in Knowledge Graphs to respond to people's natural language questions has become increasingly prominent.

In a complete Automatic Question Answering System, the input voice signals generated by the user can be converted from audio input to textual information via the speech recognition module, but the original natural language text can only be recorded and saved by the machine, and the computer has no idea how to "understand" the meaning conveyed in the raw text. The common Natural Language Understanding task (NLU) is to transform plain text information into a structured information that the computer can parse, called a semantic expression. Semantic expression is the core idea in natural language understanding tasks.

The Natural Language Understanding module defines a complete set of semantic expression form for the target domain. Different tasks and different domains adopt different semantic expressions. Semantic information is passed to the dialog management module to perform corresponding infomation retrieval requests or dialogue actions. Natural Language Understanding is an extremely complicated task, since natural language is inherently ambiguous (for example, apples can refer to a kind of fruit, it can also refer to

a company founded by Steve Jobs) and diversity (such as Shanghai Jiaotong University can be abbreviated as SJTU). Human beings not only need to understand the literal meaning of each word, but also rely on the context and need extra background knowledge or common sense as additional support. Natural Language Understanding based on Knowledge Graph provides us with this kind of rich background knowledge, as well as the organizational structure to facilitate the query and retrieval. With the successful application of deep learning in natural language processing tasks, the intelligent customer service Question Answering System based on Natural Language Understanding Supported by Knowledge Graph will provide more professional and user-friendly services.

In a word, the Automatic Question Answering System aims to provide direct answers to questions asked by users, and the ability of Knowledge Graph Database to organize, manage, and understand massive structured information, provides a profound background knowledge for the computer to identify the intent of users' inquiries. Knowledge Graph brings new opportunities for Natural Language Understanding, and it also shows great vigor in intelligent Question Answering System. It has become the infrastructure of intelligent services nowadays.

In this thesis, we first investigated a variety of semantic representations, including Semantic Frame, Dialogue Act, and the Logical Forms like  $\lambda$  calculus. After comparing three classic methods to the task of Question Answering over Knowledge Graph, namely method based on semantic parsing, method based on information extraction and method based on vector space modeling, we then delved into the prevalent semantic slot filling and intent classification tasks and did some survey on the current popular neural network models such as Sequence-to-Sequence Model, Attention Model. After that we summarized the common types of queries based on Knowledge Graph and list the main problems we confront with, thus formulated our research ideas and plans.

Starting from the basic point of commercial application, this paper first innovatively proposes and establishes a set of logical forms suitable for Question Answering System based on Knowledge Graph. Our proposed semantic expression is more easily understandable than the Lambda Dependency-Based Compositional Semantics logical form while as powerful as it. With respect to this semantic expression, we tries a rule-based finite state transducer method and statistics-based neural network models. Aiming at two different domains, one is AISpeech domain about enterprise information queries with perfect rules system and continuous maintenance during continuous on-line testing, and the other is a preliminarily established SpeechLab domain about intelligent speech laboratory information queries without any prior knowledge or raw data. We analyze the differences among various neural network models, including Bi-RNN, Bi-RNN plus CRF and one variant local attention model called Focus, and compare the performances of rule-based method with statistics-based method step by step. We point out both the advantages and disadvantages of these two methods. In the experiments chapter of this paper, we introduce and analyze the results on two domains, AISpeech and SpeechLab, in detail. The AISpeech domain is a relatively well-organized domain, and the SpeechLab domain aims at a new domain about short-term construction of the Shanghai Jiaotong University's Intelligent Speech Laboratory. In this totally new domain, rule system has not yet been tested. In the rule-based method, the AISpeech domain with relatively complete rules meets the requirements for

on-line use. On the cold start issue of SpeechLab, the performance is very poor, the system still need improvement of the rules. In statistical methods, the experimental results of the three models accord with the complexity of the models. No matter which domain, the Focus model is superior to the Bi-RNN+CRF model, and the Bi-RNN+CRF is superior to the Bi-RNN model. In the domain of AISpeech, the statistical model shows more powerful learning ability and generalized reasoning ability compared with that of rule parser. The generalization ability is further demonstrated in the domain of SpeechLab. Even the test queries do not meet with any rule template or contain entities or attribute constraints that donot appear in training set, there is still high possibility that the statistical parser can get the desired result. The accuracy of the statistical parsing result is much higher than that of the rule model. Although the statistical model is superior to the rule parser in both the learning ability and the generalization ability in our experiment, it does not mean that the statistical parser can completely replace the rule parser. The recursive parsing problem can be solved using rule-based system very easily. However, with respect to the statistics-based parser, it is difficult to implement this function due to the flattened output structure of the neural network. The structured semantic output representation problem of neural networks has been a major difficulty in the application of Deep Learning to Natural Language Understanding.

In the remaining part of our paper, we first briefly discuss the definition of ontology about the Knowledge Graph and semantic expression, and introduce the commonly used evaluation criteria in the first chapter. In the second chapter, this paper introduces related background knowledge, briefly explains the definition and common operations of finite state machines, the widely use of conditional random fields in sequence labeling problem, the basic knowledge of neural networks, and the popular Sequence-to-Sequence, Attention model. The third part of this article introduces the definition of our rules and the principle of the rule-based FST Parser. This paper sums up the commonly used query types, and points out several major difficulties in Natural Language Understanding tasks based on Knowledge Graphs. In light of these difficulties, we have discussed in detail the semantic logic forms that we define and use in experiments. In the appendix, a detailed example of using FST to parse sentence is given. In the fourth part of the paper, we first introduce the sources of the dataset in detail, including the training set generation procedure and the test set crowdsource collection. Then we elaborate on the neural network models we used in statistical experiments step by step, including the Bi-RNN, Bi-RNN+CRF, and a variant of Attention model. In order to realize the transformation from the output of the network to the logical form, we also introduced in detail the flat representation of the logical form and many algorithms of the decoding process.

All in all, based on the previous research experience, this paper tries the framework of rule-based parsing and constructs a new neural network statistical Natural Language Understanding model under our defined logical form. The experimental results of the statistical model greatly exceed the baseline of rule system. Moreover, the statistics model also has rapid adaptability and flexible scalability, and presents a great potential for improvement, waiting for further study in the future.