

天津大学

本科生毕业论文



学	院	<u>计算机科学与技术学院</u>
专	业	<u>计算机科学与技术学院</u>
年	级	<u>2014 级</u>
姓	名	<u>陶舒畅</u>
指导教师		<u>张鹏</u>

2018 年 5 月 14 日

摘 要

中文摘要一般在 400 字以内，简要介绍毕业论文的研究目的、方法、结果和结论，语言力求精炼。中英文摘要均要有关键词，一般为 3 — 7 个。字体为小四号宋体，各关键词之间要有分号。英文摘要应与中文摘要相对应，字体为小四号 Times New Roman，详见模板。

关键词：关键词 1；关键词 2；关键词 3；……；关键词 7（关键词总共 3 — 7 个，最后一个关键词后面没有标点符号）

ABSTRACT

The upper bound of the number of Chinese characters is 400. The abstract aims at introducing the research purpose, research methods, research results, and research conclusion of graduation thesis, with refining words. Generally speaking, both the Chinese and English abstracts require the keywords, the number of which varies from 3 to 7, with a semicolon between adjacent words. The font of the English Abstract is Times New Roman, with the size of 12pt(small four).

Keywords: keyword 1, keyword 2, keyword 3, ·····, keyword 7 (no punctuation at the end)

目 录

第一章	绪论	1
1.1	课题背景	1
1.2	研究意义	2
1.2.1	短文本-短文本匹配	2
1.2.2	短文本-长文本匹配	2
1.2.3	长文本-长文本匹配	3
1.3	研究内容	3
1.4	本文组织结构	4
第二章	相关工作与国内外研究现状	5
2.1	文本匹配研究相关工作	5
2.2	强化学习研究现状	11
2.3	本章小结	14
第三章	基于强化学习的文本匹配算法建模	16
3.1	文本匹配描述	16
3.2	马尔可夫决策过程	16
3.2.1	马尔科夫相关概念	17
3.2.2	马尔可夫决策过程求解	18
3.3	基于强化学习的文本匹配建模	20
3.3.1	基于马尔可夫决策过程的匹配路径建模	21
3.3.2	匹配路径的判别模型	24

第四章	基于价值迭代的文本匹配算法实现	25
4.1	算法实现	25
4.1.1	训练过程	25
4.1.2	预测过程	26
4.2	实验数据及评价指标	26
4.2.1	实验数据	26
4.2.2	评价指标	27
4.3	实验结果及分析	28
4.4	本章小结	30
第五章	基于蒙特卡洛树搜索的文本匹配建模及实现	31
5.1	蒙特卡洛树搜索	31
5.1.1	蒙特卡洛树搜索介绍	31
5.1.2	围棋中的蒙特卡洛树搜索应用	31
5.2	蒙特卡洛树搜索增强的马尔可夫决策过程的文本匹配建模	33
5.2.1	蒙特卡洛树搜索增强的 MDP	34
5.2.2	匹配路径的判别	35
5.2.3	蒙特卡罗树搜索的训练和推断	36
结 论		36
参考文献		39
外文资料		
中文译文		

致 谢

第一章 绪论

1.1 课题背景

语言是人类交流沟通的重要方式，人们的绝大部分知识也是以语言文字的形式记载和流传下来的，它在人类的社会生活中起着至关重要的作用。随着时代的发展，计算机越来越广泛地渗透到人类社会的各个领域，语言文字信息也逐渐数字化，互联网技术的迅猛发展，人们摆脱了信息贫乏的桎梏，进入了一个信息极度丰富的社会。

在英剧《黑镜 (Black Mirror)》中有一集是刻画女主人公在男友艾仕车祸去世后，在朋友的推荐下，利用艾仕在社交网络上留下的信息塑造了一个可以“完美”模仿艾仕的人工智能系统，可以和女主角语音交流。这个人工智能系统就是通过阅读艾仕的邮件、聊天记录等海量信息来了解他。理解语言这项人类的基本技能，也成为了众多科幻作品中人工智能的入门标配。在计算机学科和人工智能学科中，研究如何让机器理解语言已经成为了一个专门的研究领域，即自然语言处理。人工智能取得了一系列进展，并在各个领域取得了广泛应用，不过毋庸置疑的是，自然语言处理始终是实现自然人机交互愿望的一块重要技术基石。

自然语言处理的主要研究内容，是如何让计算机处理语言文字信息，以及实现人与计算机之间用自然语言进行有效沟通。然而，这是十分困难的，造成困难的根本原因是自然语言文本的各个层次上广泛存在的各种各样的歧义性，包括词语中的一词多义与多词同义、短语中多个词性所导致的语法歧义、句子中的语序不同却表达同样意义等等。解决这个问题可以抽象为判断两个文本是否匹配，比如：一个有歧义的文本表达的是一种含义，而另一个没有歧义的文本表达的也是同样的含义，若文本匹配系统判断两个文本匹配，则说明系统可以有效识别文本信息。

文本匹配是自然语言处理领域中的核心问题之一，目标是对于两个文本，给出它们，求出相似度，进而判断其是否匹配。许多任务，如问题复述、阅读理解、信息检索、机器翻译、自动文摘系统都可以归结成文本匹配问题。问题复述，即对相同语义的不同表达，可以抽象为判断两个句子的语义是否匹配；阅读理解，可以抽象为文章中信息和所问题目之间的匹配；信息检索，可以抽象为检索词和文档的匹配，机器翻译可以抽象为两种不同语言之间的匹配；自动文摘可以抽象为文章和摘要之间的匹配。

其中，以问题复述、信息检索、以及机器翻译为代表等多个任务，已经广泛应用在人们平时生活中的。问题复述可以应用在知乎、Quora 等在线问答平台，

用户可能会提问很多相似问题，判断两个问题是否表达了同样的语义，即两段文本是否匹配，十分重要。信息检索在生活中的应用，早已不止于人们耳熟能详的搜索引擎百度、谷歌，在微博、推特等社交媒体中，搜索已经作为独立的应用上线，微博搜索已经成为用户关注时事热点的主要来源，微信也提供了微信公众号文章搜索，历史记录搜索等，这些技术的关键都是文本匹配任务。因而文本匹配的研究具有重要的理论和应用价值。^[9]

1.2 研究意义

对于文本匹配问题的研究，可以应用在自然语言处理绝大部分任务中。根据使用场景的不同，可分为三类：短文本-短文本匹配，短文本-长文本匹配，长文本-长文本匹配。

1.2.1 短文本-短文本匹配

很长一段时间以来自然语言处理都在研究句子级的文本，即短文本。短文本和短文本的匹配在工业界应用最为广泛，如：问题复述、机器翻译、自动问答系统等。

以问题复述在机器学习中的经典方式为例，对于给定的两句话，首先要获得他们的单词表达，通过函数映射得到句子表达，然后将两个文本进行交互，提取它们的交互特征信息，最后综合前面一个或多个信息得到他们匹配程度的打分，这也是文本匹配任务的一般过程。问答系统，当前自动问答的大量工作主要是基于知识库进行检索，需要将问题和知识库中的候选答案进行匹配，返回按匹配程度从高到低排序序列。

机器翻译也很类似，如将一句话由语言 A 翻译成语言 B，用在统计机器翻译中首先要从语言 B 的平行语料库中找到该句子的每个词语使用语言 B 的表示，这就需要用到单词级别的匹配技术。可以使用单词级别的词向量进行匹配。

1.2.2 短文本-长文本匹配

短文本和长文本的匹配在工业界也有许多应用，如信息检索、阅读理解、自动文摘等。

信息检索的主要过程是可以分为三部分：建立文档索引，查询召回候选文档，检索。首先要从预处理之后的文档集离线生成索引，用户输入查询语句，搜索引擎初步召回一批候选文档，将查询词或语句和候选文档进行文本匹配，将候选文档按匹配程度由高到低进行排序后返回。其中召回候选文档和检索是关键步骤，都需要用到文本匹配。召回候选文档时需要将生成的索引与用户查询进行匹配；检索则是通过计算查询语句和各个文档的语义相关度，得到他们之间的匹配程度，并进行排序后返回在网页上。召回候选文档和检索时由于候选文档相对

较长，需要用到短文本与长文本的匹配。短文本与长文本的匹配方式与 1.2.1 中的整体方法类似，区别在于短文本得到句子表达即可，长文本表达的信息更多，还需要得到段落表达，而由于段落是以层次化的形式组织起来的，文本匹配时还需要考虑不同层次的匹配信息。

阅读理解的时候，要将阅读中的问题和原文章进行短文本-长文本匹配，而理解长文本语义与句子间的连贯、上下文、句子的歧义性有很大关联，因而采取什么样的形式表达长文本也是一个很大的难点。同时短文本和长文本的匹配存在着匹配的非对称性问题。

1.2.3 长文本-长文本匹配

长文本和长文本的匹配的应用，相对较少。原因是长文本组织层次丰富，同时存在上下文关联情况，因而长文本之间匹配的应用，如新闻推荐，往往可以先提取摘要，然后将摘要进行匹配。新闻推荐还可以使用主题模型，先得到两个文本的主题分布，再通过计算两个多项分布的距离，反映出他们之间的匹配程度。

1.3 研究内容

文本匹配的研究方式主要可分为传统方法，和机器学习方法。传统方法主要基于人工提取特征、设计规则，一个好的模型依赖于提取出的优质特征、规则，因而耗费人力较大，规则难以维护，系统也较为复杂。

近几年来，随着深度学习在语音识别、计算机视觉取得了突出进展，文本匹配也成为了当前深度学习研究的热点问题。基于深度学习的文本匹配模型，是利用深度神经网络，从文本中直接提取模式，并计算匹配程度得分。由于它可以直接从大量的数据中，端到端的学习特征，节省了人力物力。文本匹配的研究人员提出了许多深度学习模型，分别在不同的具体任务上有效。其中一个模型 MatchSRNN 在匹配任务结束后，通过回溯找到了两个句子的匹配路径，这符合人类在判断两个句子是否匹配的自然过程。但是使用传统方法，或者深度学习方法来找到匹配路径，需要大量特征、以及标注数据，来定义正确的匹配路径，成本和复杂度较高。因而本文希望找到一种可以生成匹配路径的方法。

最近，基于，在围棋领域战胜了人类高手李世石，引起了学术界的广泛关注。强化学习是机器学习的一个重要分支，一个完整的强化学习过程，可以让计算机由完全不了解任务，通过不断的尝试，从和环境的交互学习，最后找到规律，达到任务目的。强化学习算法本身就会和环境进行交互，从获得的奖励或惩罚中学习。这些奖或惩，其实就可以作为生成匹配路径的需要的大量数据，同时由于这种学习过程，强化学习在序列生成问题上表现突出，生成匹配路径本质是一种序列生成的过程，于是本文使用强化学习算法来模拟匹配路径的生成过程，从而计算匹配程度。

基于上述现状，本文使用强化学习方法来建模文本匹配问题，采用了马尔可夫决策过程（Markov Decision Process, MDP）方法本文的主要贡献如下：

针对文本匹配问题，设计了马尔可夫决策过程中的状态、动作、价值函数以及奖励函数，实现了基于价值迭算法（value iteration）的文本匹配模型，在在线问答平台 Quora 数据集训练模型，并进行优化，与经典的深度学习的文本匹配模型 MatchPyramid 和 MatchSRNN 的结果进行比较，实验结果表明基于价值迭代算法的文本匹配模型，在各个评价准则下均优于其他经典文本匹配算法的效果。

1.4 本文组织结构

本文共分为六章，每章节的内容组织如下：

第一章为绪论部分，主要介绍了文本匹配任务的研究背景及意义，并说明了本文的主要贡献。首先由自然语言处理引出文本匹配这一重要任务，根据使用场景的不同分为三类，介绍了各自的经典任务以及实现方式，总结了当前文本匹配的研究现状，提出了可以利用强化学习的建模文本匹配过程。

第二章从两个方面介绍了国内外研究现状。第一方面是关于文本匹配的相关工作，主要介绍了模，从传统方法到深度到强化学习，另一方面是强化学习的研究现状，主要介绍了强化学习的基本概念、符号定义以及相关算法，为本文之后提出的模型打下基础。

第三章介绍了文本匹配问题的特点以及详细描述，引入了强化学习中的马尔可夫决策过程，设计了基于强化学习的文本匹配模型，包括面向文本匹配问题的马尔可夫决策过程状态，以及匹配路径的判别模型。

第四章实现了基于价值迭代方法的文本匹配模型，并在真实数据集上进行了实验，利用通用评价指标，将本算法和其它 xx 文本匹配模型的实验结果进行了比较，发现在大数据量的情况下表现优异。

第五章针对价值迭代基于贪心的选择方法引入的问题，利用蒙特卡洛树搜索算法对马尔可夫决策过程进行优化求解，根据蒙特卡洛树搜索算法的特点对马尔可夫决策过程的状态进行了调整，并对实验结果进行分析。

第六章对全文进行总结，本文的主要贡献和不足。

第二章 相关工作与国内外研究现状

2.1 文本匹配研究相关工作

文本匹配是自然语言处理中的经典任务，早期文本匹配的研究大多是基向量空间模型，如将文本从稀疏的高维空间映射到一个低维的向量空间的潜在语义分析模型^[1]（Latent Semantic Analysis, LSA），主题模型^[2]（Latent Dirichlet Allocation, LDA），然而这种利用统计推断发现数据潜在模式的方式在短文本上表现不好，也不能精准建模文本匹配中的语义相关程度。研究学者将机器翻译的思想引入文本匹配，Berger 和 Lafferty 使用统计机器翻译模型计算文本间的匹配程度，将两个句子的匹配视为不同语言的翻译问题，实现了近义词之间的相互匹配映射。之后又有研究者从语义紧密度等度量出发来规避结构转义问题，从对网页打关键词标签来解决非对称匹配问题等，但这样人工提取特征的代价很大，无法挖掘隐含在大量数据中含义不明显的特征，同时还会导致逻辑非常复杂。

近年来，随着计算机计算能力的不断提升，互联网数据呈爆炸式增长，深度学习依靠利用大规模数据与高性能计算的优势在语音识别、计算机视觉均取得了突出进展，自然语言处理也成为了当前深度学习研究的热点领域。有不少研究学者们使用深度学习方法来解决文本匹配问题。

基于深度学习算法的文本匹配模型的过程大致如图所示 2-1，首先将每个单词映射到词向量，利用函数得到短语或句子的中间表达，然后进行文本的交互形成匹配空间，再进一步提取其中的模式信息，最后综合前面所有信息得到匹配度得分。

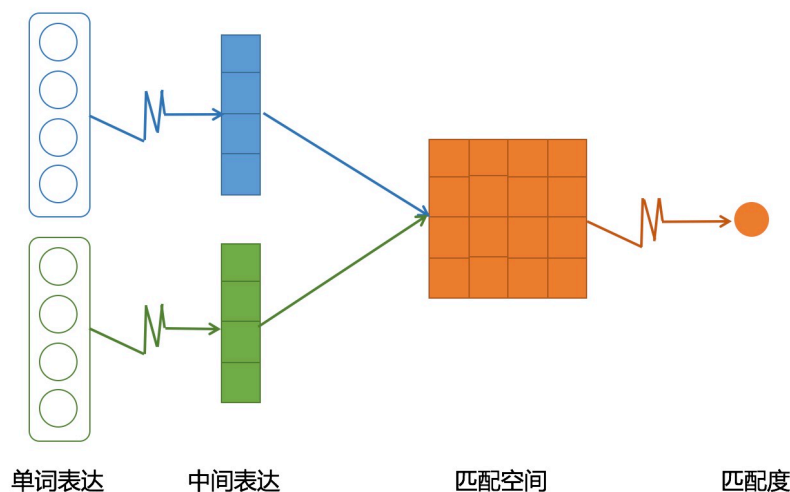


图 2-1 基于深度学习的文本匹配过程

HUANG P S, et al. 率先提出了深度语义结构模型^[3] (Deep Structured Semantic Model, DSSM), 这也是最早利用深度学习进行文本匹配的工作。主要针对查询和文档的匹配任务。它的数据是搜索引擎里查询与对应文档的点击日志, 利用深度神经网络在一个连续的语义空间学习查询和文档的低维向量表示, 通过余弦相似度计算两个文本的语义相似度。

深度语义结构模型如图2-2主要分为3层: 输入层、表示层、匹配层。输入层使用词哈希 (Word Hashing) 处理文本, 对于英文文本, 将单词切分成三个字母为一组的单词。这样可以压缩空间, 提高泛化能力。对于中文文本, 以单字的一位有效编码 (one-hot) 进行输入。在表示层, 采用词袋 (Bag of words, BOW) 的方式, 输入一个4层深度神经网络^[4] (Deep Neural Network, DNN), 输出128维的向量表示, 匹配层利用 cosine 距离衡量两个语义向量的相似度, 并使用 softmax 将查询与正样本的相似度转化为概率分布。

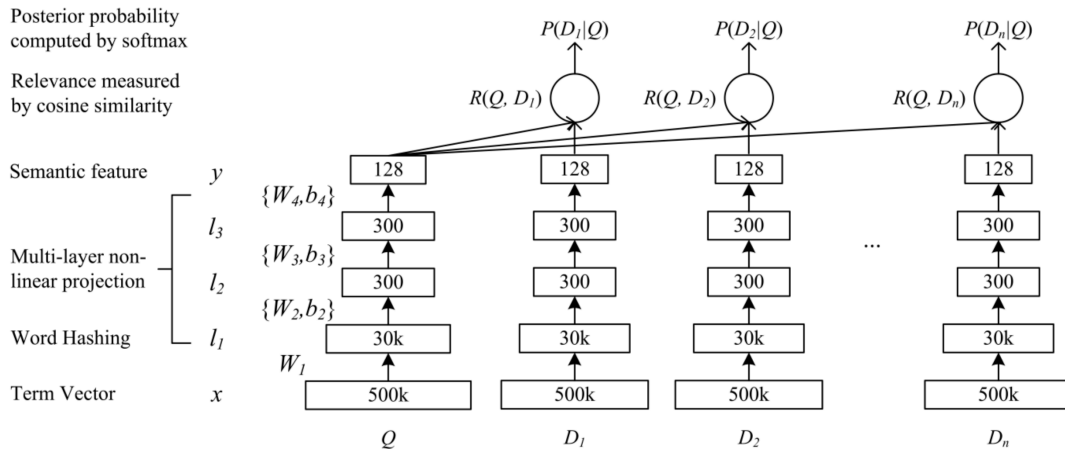


图 2-2 深度语义结构模型

这种方法可以减少对切词依赖, 并提高对模型的泛化能力, 因为语义可以服用, 同时不同于 Word2Vec 和 LDA 的无监督学习, DSSM 采用有监督学习方式, 精准度较高。不过由于 DSSM 使用的全连接神经网络参数过多, 难以训练, 同时它采用词袋模型, 缺失了语序信息和上下文信息。

针对 DSSM 的缺点, 微软提出了使用卷积神经网络^[5] (Convolutional Neural Network, CNN) 提取上下文信息, 即基于单词序列的卷积潜在语义模型^[6] (Convolutional Latent Semantic Model, CLSM), 与 DSSM 相比, CLSM 主要对输入和表示层进行了改进, 在输入层增加了单词滑动窗口 (Word-n-gram), 来提取序列信息。在表示层使用了卷积和池化结合的方式提取上下文信息。

图2-3, 单词滑动窗口包含了上下文信息, 对于滑动窗口内的词, 类似 DSSM,

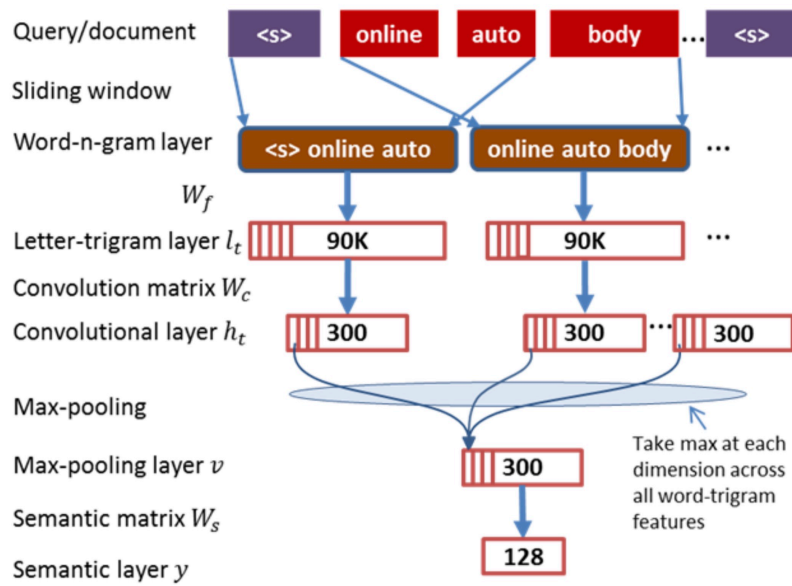


图 2-3 CLSM 模型结构

它将每个单词的 3 字母组合的词哈希向量连接得到最终的向量表示。在表示层包括了卷积层和池化层，卷积层中卷积核提取上下文特征。池化层使用最大化池化 (Max-pooling) 的方式为句子找到全局的上下文特征，最后利用全连接层将结果映射到一个 128 维的向量空间。匹配层和 DSSM 的相同，不做过多描述。

相比于 DSSM，CLSM 通过卷积和池化使得上下文信息得到较为有效的保留，但是由于卷积核大小的限制，对于间隔较远的上下文信息仍然难以捕捉。

针对上述问题，有人提出使用循环神经网络 (Recurrent Neural Network, RNN) 建模间隔较远的上下文信息。RNN 可以被看做是同一神经单元的多次复制，各个神经单元共享参数，会把消息传递给下一个，这使得它可以很好的处理序列数据。图2-4描述了一个简单的 RNN 网络结构。

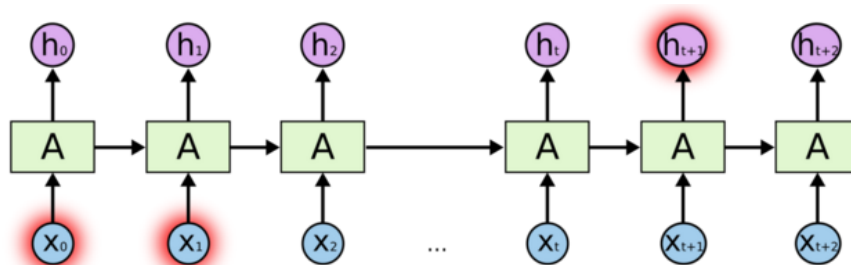


图 2-4 RNN 网络结构

不过由于 RNN 的网络和序列长度有关，因此当 RNN 的序列长度过长时，梯度消失的情况会十分明显，导致 RNN 难以对长序列建模。为解决这个问题，有研究学者提出了长短时记忆网络 LSTM^[7]，如图2-5：

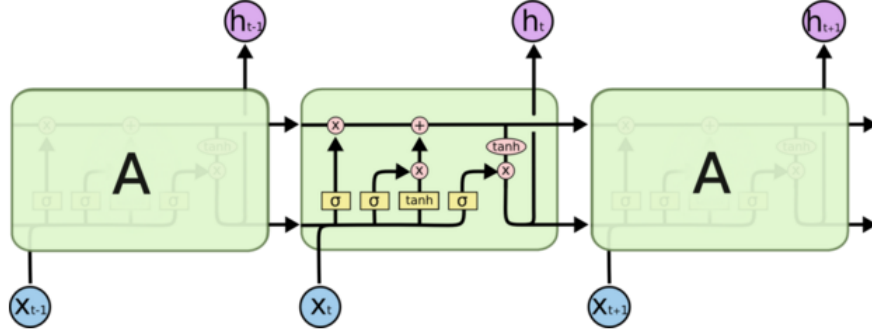


图 2-5 LSTM 网络结构

LSTM 可保留误差，用于沿时间和层进行反向传递。LSTM 将误差保持在恒定的水平，让循环网络能够进行许多个时间步的学习（超过 1000 个时间步），从而打开了建立远距离因果联系的通道。由于本文提出的算法也用到了 LSTM，接下来会重点介绍一下 LSTM 的工作原理。LSTM 的关键就是细胞状态，它类似于传送带。直接在整个链上运行，只有一些少量的线性交互。信息在上面流传保持不变会很容易。LSTM 有通过“门”的结构来去除或者增加信息到细胞状态的能力。门是一种让信息选择式通过的方法，包含一个 sigmoid 神经网络层和一个 pointwise 乘法操作。

LSTM 利用忘记门 (FORget gate) 决定会从神经单元状态中丢弃的信息，利用输入门 (input gate) 确定被存放在细胞状态中的新信息，利用输出门 (FORget gate) 确定输出的信息：

$$\begin{aligned}
 f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
 i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
 \tilde{C}_t &= \tanh(W_C[h_{t-1}, x_t] + b_C) \\
 o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
 h_t &= o_t \tanh(\tilde{C}_t)
 \end{aligned} \tag{2-1}$$

LSTM 通过这种方式，有效地避免了 RNN 网络的梯度消失问题。

LSTM-DSSM 与 CLSM 的主要区别就是将 CNN 换成了 LSTM。其整体网络结构如图2-9：

DSSM 及其衍生模型都是端到端的模型，虽然省去了算法工程师特征工程部

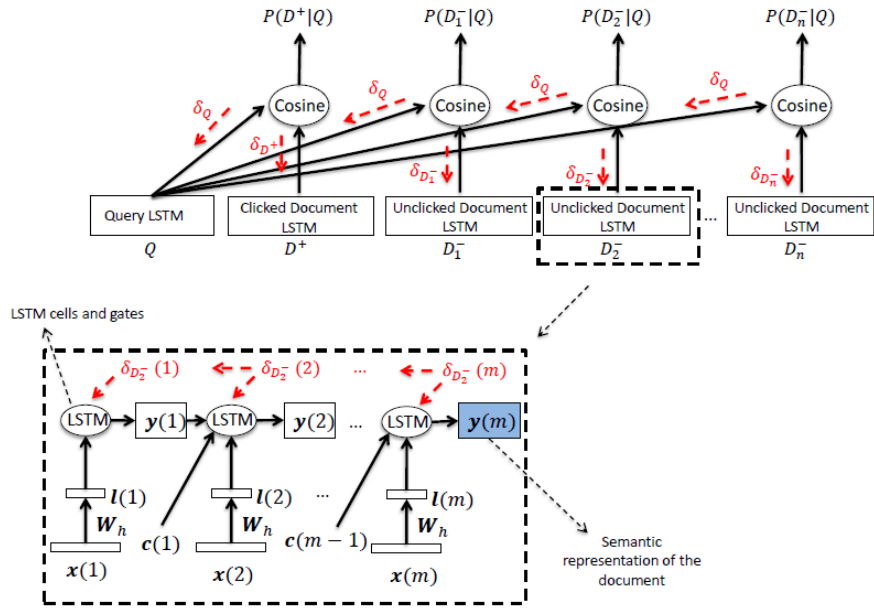


图 2-6 LSTM-DSSM 模型结构

分操作，但其效果不可控；而且 DSSM 都是若监督模型，需要大量的训练样本进行训练。DSSM 模型论文中作者提到实际训练所使用的样本量超过一亿，而且论文中所使用的样本都是曝光置信度比较高的样本。这些限制因素极大地限制了 DSSM 的使用场景，往往只有少量大公司才可以使用。

因而，有研究学者关注直接利用深度学习建模匹配模型的方式，提取句子之间的各个级别的交互特征，这种方式更加直观，也更符合人类判断两个句子是否相似的过程。

MatchPyramid^[8] 利用匹配矩阵建模两个句子的交互过程，如图 2-7。匹配矩阵中每个值都是 2 个句子中词两两计算得到的相似度。相似度可以使用词向量的余弦相似度或点积等进行刻画，而 2 个词在各自句子中的位置自然组成了一个二维坐标，开创性的构造出了单词级别相似性矩阵，即匹配矩阵。之后将匹配的问题建模为在这个匹配矩阵上进行图像识别的过程，这也是 MatchPyramid 的一大创新点。利用卷积和池化操作捕捉单词、短语、句子级别的交互信息，类似金字塔的层次结构，最终经过全连接得到句子之间的匹配程度。

但是，和 CDSSM 类似，利用分层结构的 CNN 建模文本匹配的过程，可能会失去上下文信息，使用序列结构的 RNN 优势更加显著。

有学者提出了 MatchSRNN^[9]，MatchSRNN 首先是用单词交互张量即匹配矩阵表示单词级别的交互信息，然后利用二维 RNN^[10] 递归的整合局部交互信息，最后的匹配分数是由全局交互信息计算的。单词交互张量是表达一对文本

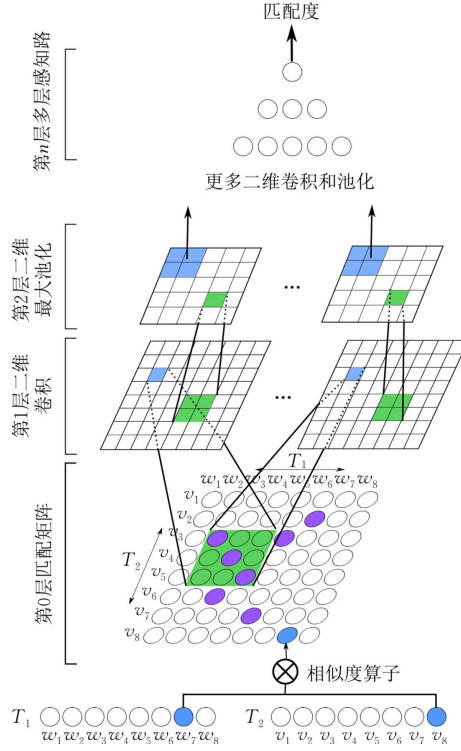


图 2-7 MatchPyramid 模型

中各个词向量之间交互信息的神经张量网络。普通一维 RNN 的当前状态 \vec{h}_t 是由前一时刻状态 \vec{h}_{t-1} 和当前输入 \vec{x}_t 决定的，而在张量网络上运行的 RNN 则将其扩展到了二维空间，它当前位置状态由左方、上方、对角线方向上的前一位置 $\vec{h}_{i-1,j}, \vec{h}_{i,j-1}, \vec{h}_{i-1,j-1}$ 与当前位置上的输入 \vec{s}_{ij} 的共同决定，RNN 输入如 (2-2)：

$$\vec{h}_{ij} = f(\vec{h}_{i-1,j}, \vec{h}_{i,j-1}, \vec{h}_{i-1,j-1}, \vec{s}_{ij}) \quad (2-2)$$

MatchSRNN 使用了 RNN 的变种门控循环单元^[11] (Gated Recurrent Unit, GRU) GRU 是 LSTM 的简化，只有更新门和重置门，重置门决定了如何将新的输入信息与前面的记忆相结合，更新门定义了前面记忆保存到当前时间步的量。GRU 的计算方式：

$$\begin{aligned} r_t &= \sigma(W_r[h_{t-1}, x_t]) \\ z_t &= \sigma(W_z[h_{t-1}, x_t]) \\ \tilde{h}_t &= \tanh(W_h[r_t \times h_{t-1}, x_t]) \\ h_t &= (1 - z_t) \times h_{t-1} + z_t \times \tilde{h}_t \end{aligned} \quad (2-3)$$

MatchSRNN 用 GRU 遍历匹配矩阵，计算了每个门在三个方向上的值，如图 2-8，使用 SoftmaxByRow 对每一个维度进行线性变换，最后对二维 GRU 进行全连接层，非线性变换得到了最后的匹配程度。

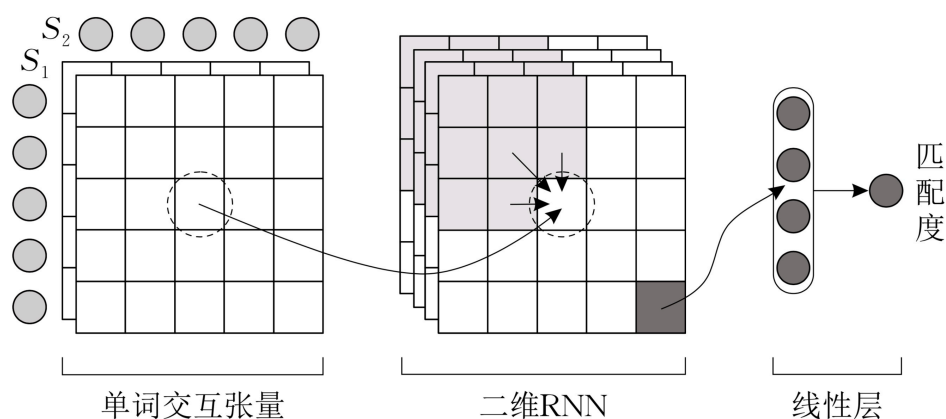


图 2-8 MatchSRNN 模型

2.2 强化学习研究现状

强化学习是机器学习中与监督学习、非监督学习并列的一个重要研究领域，它的本质是解决序列决策的问题，即自动进行决策，并且可以做连续决策^[12]。和监督学习给定标注不同，强化学习给定的是一个奖励函数，主体（agent）以试错（trial-and-error）的机制与环境进行交互，最终目标是使主体在与环境交互中得到最大的累计奖励。这类算法有三个特征：闭环性，学习系统产生的行为会影响到后续的输出；无监督，学习对象只能通过学习去得这这些信息；延时性，行动（action）产生的结果，包括奖励（reward），需要很多个时间周期才能显现出来。强化学习中最基本的要素是主体和环境，除了这二者之外，还有四个要素策略、奖励信号、价值函数、对环境的建模（model）。其中：

- 策略（Policy, π ）定义了主体在给定时间内的行为方式，是从环境到这些状态下采取行动的映射，是强化学习主体的核心，可能是简单函数或查找表，也可能是随机的；
- 奖励信号（Reward signal, R ）定义了强化学习问题的目标。在每个时间步骤，环境都会向主体发送一个奖励信号，主体唯一的目标是最大限度的提高长期得到的总奖励，因而奖励信号是改变策略的主要依据，它也是一个函数；
- 价值函数（Value function, V ）定义了长远的回报。一个状态的价值是从该状态开始，这个主体在未来可以预期积累的总价值。奖励决定了状态的直接内在可取性，价值表明各个状态考虑到未来状态的长期可取性；
- 模型（Model, M ）推断环境会如何表现，即给定一个状态和行为，模型会预测下一个状态和下一个奖励，模型用于规划，通过历史信息，考虑可能

的未来情况来确定行为的方式。不过不是每个强化学习方法都需要模型，有的问题可以使用无模型方法解决。

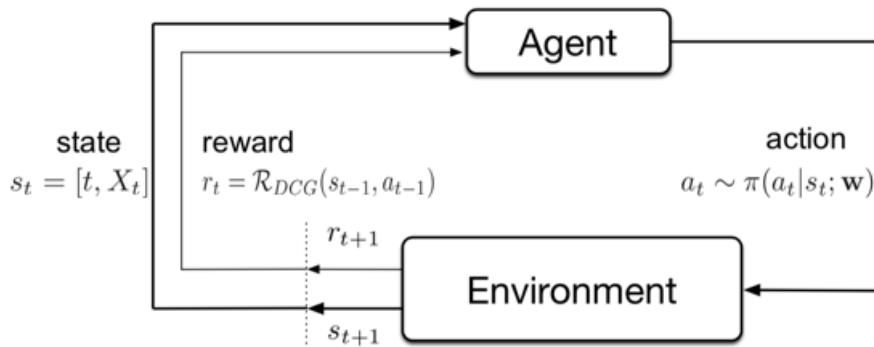


图 2-9 主体与环境的交互

主体和环境的交互可以用图??表示。主体在当前状态 s_t 下根据策略 选择动作 a_t ，环境接收到该动作并转移到下一状态 s_{t+1} ，主体接收环境反馈回来的奖励选择下一步动作。强化学习不需要监督信号，主要算法包括 Q 学习（Q-learning），策略梯度（policy gradient）等等。

深度强化学习的主要思路是将神经网络用于抽取复杂高维数据中的信息，并将其映射到一个低维向量空间便于强化学习处理。由于卷积神经网络在计算机视觉领域的统治地位，DeepMind 团队在 2013 年尝试将卷积神经网络和强化学习结合，提出深度 Q 网络（DeepQ Network, DQN）^[13]，并成功的将该方法用在了 Atari 视频游戏。这是深度强化学习首次在高维度的状态空间下起作用。

2015 年，DeepMind 团队进一步完善了 DQN 算法^[14]。DQN 将深度卷积神经网络和 Q 学习结合到一起，并集成了经验回放技术 (memory replay) 和目标 Q 网络。经验回放通过随机采样系统在探索环境时得到的状态数据对神经网络参数进行更新，打破了 Q-学习算法采样数据之间的相关性。DQN 在没有任何人类先验知识的情况下在 Atari 视频游戏表现出了等同人类玩家的水平纪念，是深度强化学习领域的重要工作。

2016 年初，DeepMind 团队发表了围棋 AI: AlphaGo^[15]。AlphaGo 利用强化学习指导蒙特卡洛树搜索的过程，将深度强化学习的研究推向了新的高度。它通过策略网络学习不同位置的落子概率，利用价值网络学习棋局的胜率评估，通过策略和价值网络的结合减小了蒙特卡洛树搜索的搜索次数，提高了搜索效率。在在线对弈时，利用蒙特卡洛树搜索以及策略和价值网确定当前的落子位置。

2017 年初，AlphaGo Zero^[16] 对 AlphaGo 进行了改进和升级。AlphaGo Zero 抛弃了 AlphaGo 复杂的特征输入，只需要将棋局图片作为数据即可；将策略网

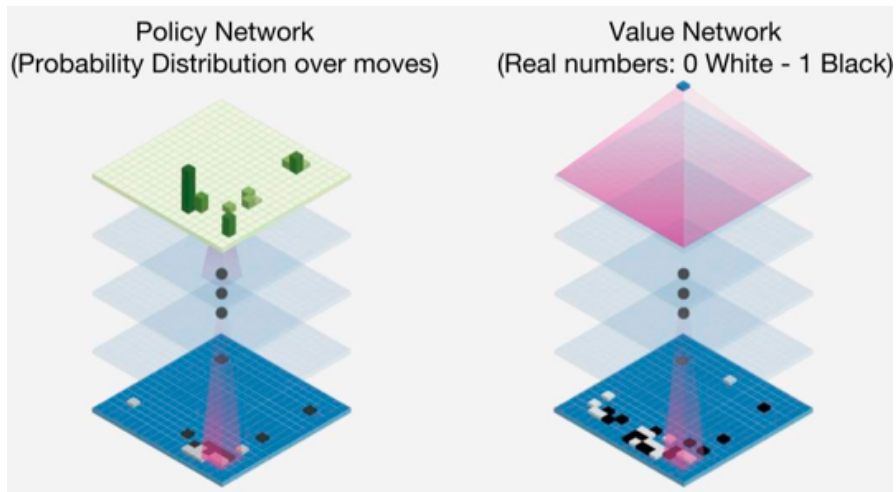


图 2-10 AlphaGo 的策略网络和价值网络

络和价值网络整合在一起，直接利用深度强化学习方法进行端到端的自我对弈学习。相比于 AlphaGo，AlphaGo Zero 去除了棋手的落子网络，因此不需要任何先验知识；策略网络和价值网络的整合使得神经网络的复杂度降低，泛化性进一步增强，降低了硬件的资源需求，减少了训练时间。AlphaGo Zero 的成功证明

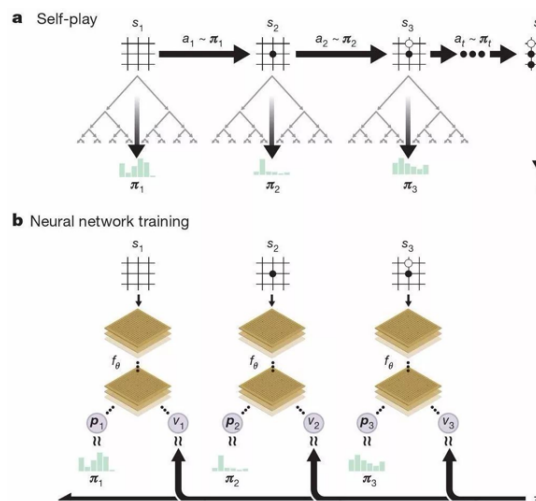


图 2-11 AlphaGo Zero 自我对弈训练过程

了在任何没有先验经验的情况下，深度强化学习在围棋领域仍然能取得巨大的成功；而在围棋下法上，AlphaGo Zero 创造了更多的下棋方式，大大开拓了人类对围棋的认知。

2.3 本章小结

本章主要介绍了文本匹配和强化学习的相关工作与研究现状。

本章的第一小节主要介绍了基于深度学习的文本匹配算法。目前基于深度学习的文本匹配算法大都集中于利用神经网络理解输入句子的语义信息。最早利用深度学习解决文本匹配问题是 **DSSM**。**DSSM** 利用词哈希以及词袋的方式得到句子的向量表示,并利用一个三层的全连接网络将句子的向量表示映射为一个低维向量表示。最后利用余弦相似度计算两个语义向量的距离,并利用 **softmax** 对得到的结果进行非线性变换,得到最后的概率分布。**DSSM** 通过词哈希降低了对且此算法的依赖,但是由于词袋模式的使用使其难以捕捉句子的上下文信息,而且全连接网络参数量极大,很难训练。为了解决 **DSSM** 模型问题,有人提出了 **CDSSM**。相比于 **DSSM**,**CDSSM** 在词哈希的基础上引入了滑动窗口,通过滑动窗口解决了 **DSSM** 中上下文信息建模苦难的问题。同时 **CDSSM** 在表达网络中引入了卷积和池化操作,通过卷积建模上下文特征,利用池化发现全局特征,最后利用一个全连接层将结果映射到一个低维向量空间。但是由于卷积神经网络不是为序列化设置,因此对于长句子的上下文信息,**CDSSN** 仍然难以捕捉。针对这个问题,有人提出将 **LSTM** 用于表达网络。**DSSM** 及其后续的改进工作虽然有效的提高了文本匹配的效果,但是 **DSSM** 端到端的学习方式以及弱监督特征决定了它的利用场景十分有限。这种情况下有人提出了直接建模匹配模型的方法。**MatchPyramid** 利用两个句子之间词的相似度构造了一个匹配矩阵,将文本匹配过程视为对这个匹配矩阵的分类过程。**MatchPyramid** 使用 **CNN** 建模了分类过程。和 **MatchPyramid** 类似,**MatchSRNN** 在匹配矩阵上使用了一个二维的 **GRU** 建模文本匹配的过程。

本章的第二小节主要介绍强化学习。强化学习是机器学习中的一个重要领域,最早用于机械控制领域,强调如何基于环境行动已获得最大收益。早期的增强学习方法包括策略梯度,值迭代等等,但是由于数据量过小,模型表达能力不强等等原因,强化学习的建模能力一直不强。近年来强化学习和深度学习的结合大大增加了强化学习的表达能力,使得强化学习近年来获得了长足的发展。**DeepMind** 提出了 **DQN** 并将其用于 **Atari** 游戏中,强化学习首次拥有了在高维状态空间下解决问题的能力。

16 年初,**DeepMind** 团队提出了 **AlphaGo** 算法,该算法将强化学习和蒙特卡洛树搜索结合,通过策略网络和价值网络指导蒙特卡洛树搜索过程,大大提升了蒙特卡洛树搜索的速度。去年 **DeepMind** 进一步改进了 **AlphaGo** 算法,提出了 **AlphaGo Zero**。**AlphaGo Zero** 算法在没有任何人类先验知识的情况下在围棋领域取得了巨大的成功。

目前深度学习和强化学习的结合使得强化学习的表达能力大幅度上升，为学习传统文本匹配中复杂的规则和模式带来了可能。由于目前计算机理解人类语义十分困难，因此基于模式匹配的方式在短文本匹配场景中仍然拥有巨大的价值。本文会结合强化学习和文本匹配的特点，设计使用与文本匹配的强化学习算法。

第三章 基于强化学习的文本匹配算法建模

文本匹配的应用本章主要介绍了文本匹配问题的数学描述，强化学习的主要算法——马尔可夫决策过程以及面向文本匹配的算法设计。

3.1 文本匹配描述

为了更好地解决文本匹配问题，本节会利用数学符号形式化地定义文本匹配这个任务。

文本匹配任务的输入为两个句子集合和一个标签： $S_1 = \{q_1, q_2, \dots, q_n\}$, $S_2 = \{d_1, d_2, \dots, d_n\}$, $L = \{y_1, y_2, \dots, y_n\}$ ，给定训练数据对，两个文本和一个标签： $(q_1, d_1, l_1), (q_2, d_2, l_2), \dots, (q_N, d_N, l_N)$ ，其中 $q_i \in Q$, $d_i \in D$ 为输入的一对文本， $l_i \in L$ 为标签，是 0 或 1。在问答系统中，三者分别为问题、答案、答案能否回答问题；在问题复述中，三者分别为要判定是否同义的两个句子和是否同意义；在搜索引擎中，三者分别为查询、文档和是否相关。

文本匹配的过程是自动学习训练数据上的匹配模型 f ，以便对于测试数据的每次输入，可以预测 q_i 和 d_i 的匹配程度 r_i ，可设定一个阈值，若高于该阈值就认为匹配，低于该阈值就认为不匹配。

在匹配的过程中，有一些关键步骤可以形式化的表述出来，以便之后求解。

- 单词表达：将每个单词映射到词向量
- 中间表达：
- 匹配空间：
- 匹配度得分：

3.2 马尔可夫决策过程

强化学习研究的是如何根据环境的反馈，作出行动，从而最大化总收益。详细过程如下：在某一时刻 t ，主体会处于状态 s_t ，并从环境中得到观测 o_t ，根据观测和行动利用策略 $\pi_\theta(a_t|o_t)$ 或者 $\pi_\theta(a_t|s_t)$ 作出行动 a_t ，环境根据主体状态 s_t 以及行动 a_t 通过转移概率分布函数 $p(s_{t+1}|s_t, a_t)$ 得到状态 s_{t+1} ，通过奖励函数 $r(s, a)$ 得到奖励 r_t ，周而复始。而当环境是完全可观测的时候，它通常被形式化的描述为马尔可夫决策过程（Markov Decision Process, MDP）几乎所有。强化学习中马尔可夫决策过程可以解决大部分问题^[12]，本小节会简要介绍马尔可夫决策过程及相关概念。

3.2.1 马尔科夫相关概念

在描述马尔可夫决策过程前，首先需要定义马尔可夫性质，一个状态有马尔可夫性质，当且仅当它满足式 (3-1)，即在给定现在状态时，它与过去状态（即该过程的历史路径）是无关的。当前的状态已经包含了历史所有相关信息的状态，只要已知当前状态，可以丢弃所有历史信息，因为该状态已经足够预测未来。

$$P[s_{t+1}|s_t] = P[s_{t+1}|s_1, \dots, s_t] \quad (3-1)$$

马尔科夫性描述的是每个状态的性质，马尔可夫链（或马尔可夫过程） $M = \langle \mathcal{S}, \mathcal{P} \rangle$ 描述了一个状态序列，其中 \mathcal{S} 是状态空间， $s_t \in \mathcal{S}$ 是满足马尔可夫性质的随机状态， \mathcal{P} 是一个状态转移概率矩阵，转移概率 $p(s_{t+1}|s_t)$ 确定了在当前状态下转移到下一个状态的概率。令转移概率 $\mathcal{P}_{s,s'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$ ，则这些概率组成了概率转移矩阵 \mathcal{P} 。马尔科夫链具有马尔科夫性，即转移概率只与当前的状态有关。

马尔可夫决策过程是马尔可夫链在决策环境中的扩展 $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ 。其中：状态空间 \mathcal{S} 与马尔科夫链类似；行动空间 \mathcal{A} 即行动的集合；转移概率 \mathcal{P} 不仅受到当前状态的影响，还受到行动 $a \in \mathcal{A}$ 的影响，定义为 $\mathcal{P}_{s,s'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$ ； \mathcal{R} 为奖励函数，定义为 $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$ ，是一个 $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ 的映射； $\gamma \in [0, 1]$ 是衰减因子。

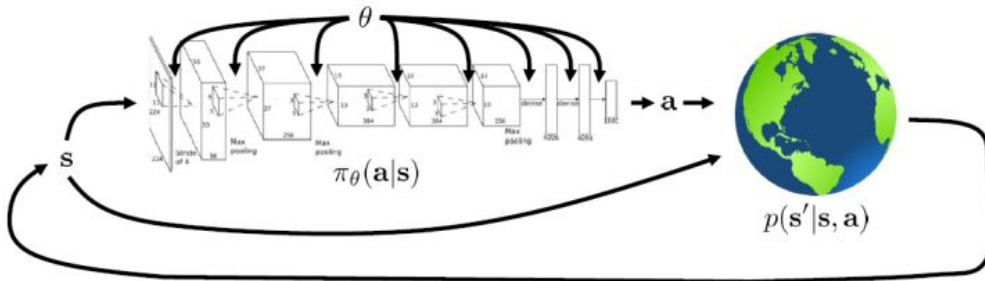


图 3-1 马尔可夫决策过程

马尔可夫决策过程，按如下进行：主体初始状态 s_0 ，然后按策略执行动作 a_0 ，根据转移概率 $\mathcal{P}_{s_0,s_1}^{a_0}$ ，跳转到下一个状态 s_1 ，再执行动作 a_1 ，转移到 s_2 ，如图

考虑一个有限长度的轨迹 $\tau = \{s_1, a_1, \dots, s_t, a_t\}$ ，它产生的概率如式 (3-2)

$$p_\theta(\tau) = p(s_1) \prod_{t=1}^T \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t) \quad (3-2)$$

初始状态 s_1 往往是确定的。根据马尔科夫性，后面每个时刻的行动和状态

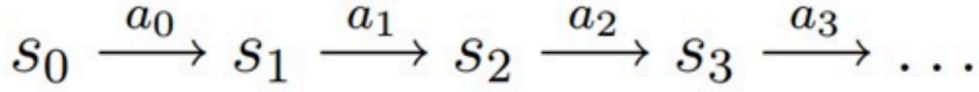


图 3-2 MDP 状态转移

都是由当前的行动确定的。我们希望优化式 (3-3) 以最大化总收益函数关于轨迹的期望。

$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_{\theta}(\tau)} [\sum_t r(s_t, a_t)] \quad (3-3)$$

对于有限长度的轨迹，我们在求解 θ^* 时只需要关注马尔科夫链在一个时间点上的边际分布；对于无限长度的轨迹，根据马尔科夫性，有式 (3-4)：

$$\begin{bmatrix} \mathbf{s}_{t+k} \\ \mathbf{a}_{t+k} \end{bmatrix} = \mathcal{P} \begin{bmatrix} \mathbf{s}_{t+k-1} \\ \mathbf{a}_{t+k-1} \end{bmatrix} = \dots = \mathcal{P}^k \begin{bmatrix} \mathbf{s}_t \\ \mathbf{a}_t \end{bmatrix} \quad (3-4)$$

对于无限长度的问题，当到达平稳分布时，我们可以对目标进行平均，如式 (3-5)：

$$\theta^* = \arg \max_{\theta} \frac{1}{T} \sum_{t=1}^T \mathbf{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim p_{\theta}(\mathbf{s}_t, \mathbf{a}_t)} r(\mathbf{s}_t, \mathbf{a}_t) \rightarrow \mathbf{E}_{(\mathbf{s}, \mathbf{a}) \sim p_{\theta}(\mathbf{s}, \mathbf{a})} r(\mathbf{s}, \mathbf{a}) \quad (3-5)$$

3.2.2 马尔可夫决策过程求解

求解马尔可夫决策过程，除上述五元组外，还需要定义一些概念：

- 策略 π : $\pi(a|s) = [A_t = a|S_t = s]$ ，根据给定状态的行动分布。策略能够完全决定主体的行为。
- 回报 G_t : $G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ 从时间步 t 开始累积的经过衰减的总奖励，即奖励期望。
- 价值函数 $v(s)$: $v(s) = \mathbb{E}_{\pi} [G_t | S_t = s]$ 从状态 s 开始，遵循策略 π 的预期回报，价值函数是一个状态在未来的价值，即回报的期望。

值函数可被分为两部分，立即奖励 R_{t+1} 和经过衰减的后继状态价值：

$$\begin{aligned} v(s) &= \mathbb{E}[G_t | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \dots) | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \end{aligned} \quad (3-6)$$

(3-6) 被称为 Bellman 方程^[17]，它表明价值函数可以通过迭代计算得到，马

尔可夫决策过程的目的就是通过迭代最大化 $v(s)$ 。

通过 Bellman 方程，可以得到 (3-7)，通过迭代计算价值函数来优化策略，收敛到最优。

$$\begin{aligned} v_{k+1}(s) &= \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) [r + \gamma v_k(s')] \end{aligned} \quad (3-7)$$

策略迭代一般分为策略评估和策略改进。策略评估 (Policy Evaluation)：利用当前策略产生新样本；策略改进 (Policy Improvement)：使用新样本更新当前策略，最终收敛到最优。在策略评估时需要知道环境的状态转移概率，因此策略迭代需要依赖模型。

Algorithm 1 policy iteration

Step 1. Initialization

$V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily FOR all $s \in \mathcal{S}$

Step 2. Policy Evaluation

repeat

$\Delta \leftarrow 0$

for each $s \in \mathcal{S}$ **do**

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s', r} p(s', r|s, \pi(s)) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

end for

until $\Delta < \theta$

Step 3. Policy Improvement

policy-stable $\leftarrow true$

for each $s \in \mathcal{S}$ **do**

$a \leftarrow \pi(s)$

$\pi(s) \leftarrow \arg \max_a \sum_{s', r} p(s', r|s, \pi(s)) [r + \gamma V(s')]$

If $a \neq \pi(s)$ then policy-stable $\leftarrow false$

end for

If policy-stable, then stop and return V and π ; else go to 2

策略迭代算法每次都要进行策略评估和改进，大大增加了算法的训练时间，由于对文本匹配问题对策略的改进和对值函数的改进是一致的，所以可以将策

略改进视为对值函数的改进。

通过 Bellman 方程，我们可以得到：

$$\begin{aligned} v_*(s) &= \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \\ &= \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_k(s')] \end{aligned} \quad (3-8)$$

从(3-8)可以发现，我们可以通过 Bellman 最优模型来更新值函数收敛，最后收敛可以得到当前策略的最优值 v_* 。

Algorithm 2 value iteration

Initialization array V arbitrarily

repeat

$\Delta \leftarrow 0$

for each $s \in \mathcal{S}$ **do**

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

end for

until $\Delta < \theta$

Output a deterministic policy, π such that

$\pi(s) = \arg \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma V(s')]$

相比于算法 1，值迭代算法 2 避免了策略评估部分的计算量，降低了算法的运行时间，有效地提高了算法的运行效率。

我们发现无论是算法 1 还是 2，每次都是选择概率或者是值最高的行为，这种策略被称为利用（exploitaion）。与之对应的，如果我们每次都根据概率或者值的分布进行采样（sampling），根据采样的结果采取行动，这种策略被称为探索（exploration）。探索的优势在于当信息匮乏时，可以取得较好的效果；利用的优势在于效率很高，但只有在信息足够充足的情况下才会有效。

探索和利用各有优势，将二者结合的算法被称为 ϵ -贪心算法。我们在每次选择行动的时候，都以 ϵ 的概率进行探索，以 $1 - \epsilon$ 的概率进行利用。可以通过 ϵ 的调整在算法的不同阶段达到不同的效果。

3.3 基于强化学习的文本匹配建模

本节将强化学习建模为一个强化学习过程。

3.3.1 基于马尔可夫决策过程的匹配路径建模

在2.1节, 中我们介绍了 MatchSRNN^[9] 算法。MatchSRNN 使用匹配矩阵中提取句子间的交互信息, 利用二维 GRU 提取单个句子和句子之间的序列信息。在二维 GRU 计算完成后, 通过二维 GRU 的状态, 我们可以回溯出一条路径。图3-3表示了对 $Q=\text{How to get rid of memory stick error of my sony cyber shot?}$, $D=\text{You might want to try to FORmat the memory stick but what is the error message you are receiving.}$ 得到的匹配矩阵进行回溯得到的路径, 如图3-3:

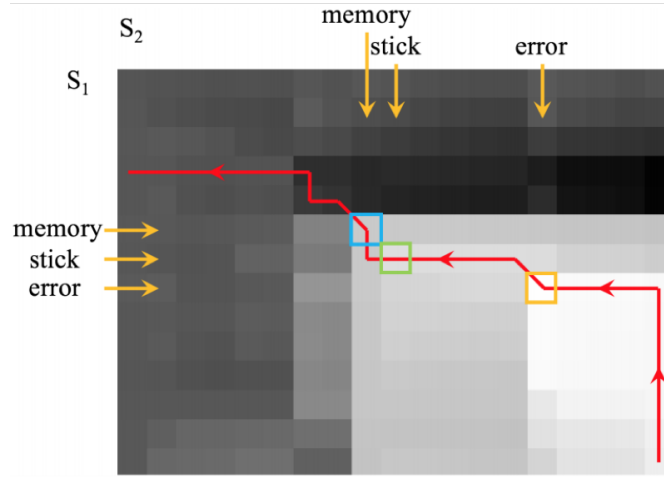


图 3-3 MatchSRNN 的回溯路径

既然 MatchSRNN 通过回溯可以得到一条路径, 那么如果不通过二维 GRU 的方式, 直接获得一条路径, 利用这条路径判断这两个句子是否匹配, 如图3-4:

和 MatchPyramid 以及 MatchSRNN 类似, 这里会使用匹配矩阵的概念。从匹配矩阵左上角出发, 根据当前位置, 确定行动方向, 直到走到右下角为止, 会得到一条匹配路径。然后需要建立一个判别模型, 根据匹配路径判断是否匹配。

首先描述文本匹配算法中获得匹配路径的过程, 该过程为一个马尔可夫决策过程。其输入为一个句子对 $Q = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n\}$, $D = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m\}$

1. 状态 \mathcal{S} : 将在 t 时刻的状态 s_t 定义为从当前位置开始向前看的 m 个单词:

$$s_t = \{[\mathbf{u}_{q_t}, \mathbf{v}_{d_t}], [\mathbf{u}_{1+q_t}, \mathbf{v}_{1+d_t}] \dots, [\mathbf{u}_{m+q_t}, \mathbf{v}_{m+d_t}]\} \quad (3-9)$$

式(3-11)中, q_t 和 d_t 表示当前位置。

2. 动作 \mathcal{A} : 在每个时间 t : 向包括右、下、右下三个方向中的某个移动一步。

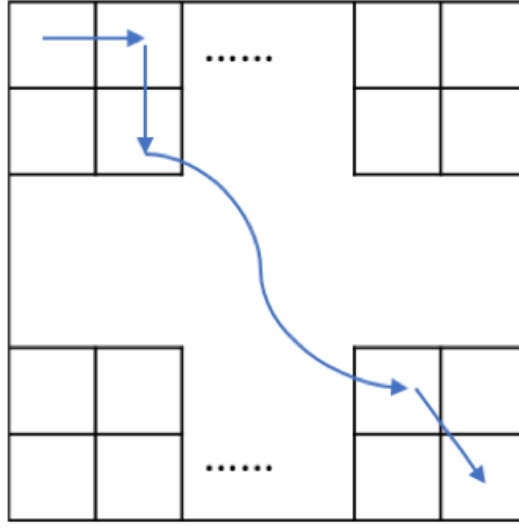


图 3-4 文本匹配路径计算过程

3. 状态转移函数 $\mathcal{T}(S, \mathcal{A})$: 状态转移函数 $\mathcal{T} : S \times \mathcal{A} \rightarrow S$ 被定义为:

$$(q_{t+1}, d_{t+1}) = \begin{cases} (q_t + 1, d_t) & \text{if goes down} \\ (q_t, d_t + 1) & \text{if goes right} \\ (q_t + 1, d_t + 1) & \text{if goes obliquely} \end{cases} \quad (3-10)$$

$$s_{t+1} = \mathcal{T}(s_t, a_t) = [\mathbf{u}_{q_{t+1}}, \mathbf{v}_{d_{t+1}}], [\mathbf{u}_{1+q_{t+1}}, \mathbf{v}_{1+d_{t+1}}], \dots, [\mathbf{u}_{m+q_{t+1}}, \mathbf{v}_{m+d_{t+1}}] \quad (3-11)$$

4. 在每个时间 t : 系统根据当前状态选择一个行动（移动方向），并根据方向移动到下一个位置，状态随之转移：根据当前的位置 (q_{t+1}, d_{t+1}) 更新未来状态，完成转移后，再根据当前状态继续移动。

5. 值函数 V : 值函数 $V : S \rightarrow \mathbb{R}$ 是对当前模型是否能够得到正确结果的预测。值函数需要不断更新以更加准确的得到预测结果。本节所提出的值函数的目标是预测模型是否能够预测正确，即 $\mathbf{1}_{t=y}$ ，其中 t 为预测标签， y 为实际标签。值函数以两个句子当前位置的前 m 个单词作为输入，确定当前橘子是否可以正确判别。

我们利用一个 LSTM 网络处理输入的序列，通过对 LSTM 网络输出的加权和进行线性变换，得到值函数：

$$V(s) = \sigma(\langle w, g(s) \rangle, b_v) \quad (3-12)$$

其中， $g(s)$ 是 LSTM 的输出：

$$g(s) = LSTM(Q[q_t : m + q_t], D[d_t : m + d_t]) \quad (3-13)$$

LSTM 网络接受两个长度为 m 的单词序列，计算得到隐状态，计算过程如式：

$$\begin{aligned}
 f_k &= \sigma(W_f[Q_{k+q_t}, D_{k+d_t}] + U_f h_{k-1} + b_f) \\
 i_k &= \sigma(W_i[Q_{k+q_t}, D_{k+d_t}] + U_i h_{k-1} + b_i) \\
 o_k &= \sigma(W_o[Q_{k+q_t}, D_{k+d_t}] + U_o h_{k-1} + b_o) \\
 c_k &= f_k \circ c_{k-1} + i_k \circ (W_c[Q_{k+q_t}, D_{k+d_t}] + U_c h_{k-1} + b_c) \\
 h_k &= o_k \circ \tanh(c_k)
 \end{aligned} \tag{3-14}$$

图 3-5 以 $Q = [\text{The, cat, sat, on, the, mat}], D = [\text{The, dog, played, balls, on, the, floor}]$, $m=3$ 为例展示了值函数。当前的状态 $s = [2, 2]$ ，我们将 $Q[2:5] = [\text{sat, on, the}]$, $D[2:5] = [\text{played, balls, on}]$ 的词向量作为 LSTM 的输入，利用 LSTM 捕捉未来词汇的序列信息，计算每个行为的值。

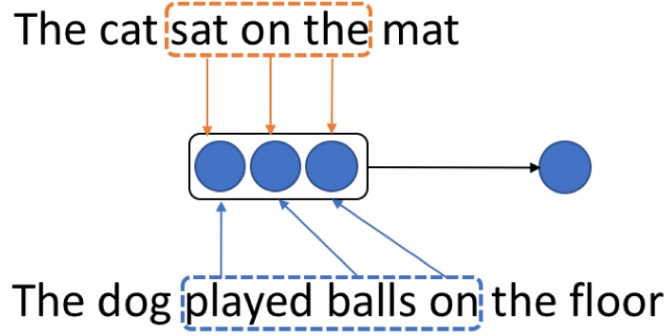


图 3-5 文本匹配值函数输入

获得匹配路径的算法可以被表示为：

Algorithm 3 MDP of Text Match

Input: Labeled data $D = \{(Q, D, Y)\}$

Output: Path

Initialize Path $\leftarrow [0, 0]$

set $q_1 = 0, d_2 = 0$

while $q_t < \text{len}(s_1)$ and $d_t < \text{len}(s_2)$ **do**

$a_t = \arg \max_a v(Q[q_t : m + q_t], D[d_t : m + d_t]a)$

update status according to Eq ??

Path $\leftarrow \text{Path} \oplus [q_t, d_t]$ { \oplus means appends to end}

end while

在计算路径时，算法从初始状态 $s_1 = [Q[1:m], D[1:m]]$ 出发，每次将当前位置的前 m 个词作为输入，得到每个行动的值，选择值最大的行动作为当前的行动，并根据当前的行为移动到下一个位置，直到到达矩阵的右下角。一直选择值最大的行动可能会导致算法陷入局部最优解，因此可以采用 ϵ -贪心的方法控制探索和利用的程度。

3.3.2 匹配路径的判别模型

在计算匹配路径之后，我们需要根据路径判断两个句子是否匹配。判别算法以原句子 Q, D 以及对应的路径映射 $m_1 = [q_1, q_2, \dots, q_t], m_2 = [d_1, d_2, \dots, d_t]$ 作为输入，判断两个句子是否匹配。

我们利用 3 个 LSTM 网络作为判别模型，其中 $LSTM_1$ 和 $LSTM_2$ 将 S_1 和 S_2 映射为 2 个矩阵， $LSTM_c$ 将这两个矩阵映射为一个一维向量 $g_c(s)$ ：

$$f(Q) = LSTM_1(Q)$$

$$h(D) = LSTM_2(D)$$

$$g_c(s) = LSTM_c([f(Q)_{q_1}, h(D)_{d_1}], [f(Q)_{q_2}, h(D)_{d_2}], \dots, [f(Q)_{q_t}, h(D)_{d_t}])$$

通过对 $g_c(s)$ 的加权和进行归一化得到最终的匹配概率

$$p = \sigma(< w, g_c(s) > + b) \quad (3-15)$$

其中 w 和 b 都是需要学习的参数， $\sigma(x) = \frac{1}{1+e^{-x}}$ LSTM 网络和 ?? 类似， $LSTM_1$ 和 $LSTM_2$ 共享参数。

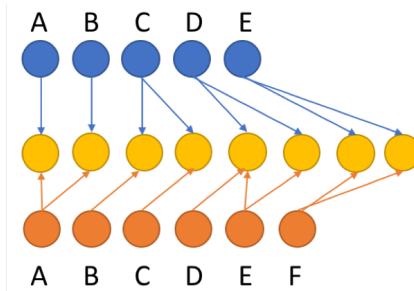


图 3-6 匹配路径判定

第四章 基于价值迭代的文本匹配算法实现

上一章介绍了基于强化学习的文本匹配建模，本章主要介绍该模型的训练与预测过程，以及试验情况。我们利用价值迭代算法实现了进行文本匹配的训练与判别，将我们的方法称为基于价值迭代的文本匹配方法 (value iteration match, VIM)。

4.1 算法实现

4.1.1 训练过程

训练时，使用算法2进行优化，模型的参数包括马尔可夫决策过程的参数 Θ_{MDP} 以及判别模型的参数 Θ_c ，在训练时，模型的输入为： $\{Q^{(n)}, D^{(n)}, Y^{(n)}\}_{n=1}^N$ ，输出参数为 Θ_{MDP} 以及 Θ_c 。

Algorithm 4 Training Process of VIM

Input: Labeled data $\{Q^{(n)}, D^{(n)}, Y^{(n)}\}_{n=1}^N$, learning rate η

Output: $\Theta_{MDP}, \Theta_c \leftarrow$ random values in $[-1, 1]$

while not convergency **do**

 generate path according to Alg 3

while not convergency **do**

$\Theta_c = \Theta_c - \eta \frac{\partial \ell_c}{\partial \Theta_c}$ { ℓ_c is defined in Eq 4-2 }

end while

$\Theta_{MDP} = \Theta_{MDP} - \eta \frac{\partial \ell}{\partial \Theta_{MDP}}$ { ℓ is defined in Eq 4-4 }

end while

在训练阶段，每次迭代时，利用算法3为每个样本 Q, D 生成一条路径：

$$\{[q_1, d_1], [q_2, d_2], \dots, [q_t, d_t]\} \quad (4-1)$$

并根据生成的路径以及真实结果训练路径判别模型。训练路径匹配模型时的损失函数为交叉熵：

$$\ell_c(Y, p) = -(Y \log(p) + (1 - Y) \log(1 - p)) \quad (4-2)$$

在路径判别模型收敛后，利用判别模型输出该路径未匹配的概率 p ，并利用 p 计算奖励：

$$r = \log\left(\frac{1}{|p - Y + \Delta|}\right) \quad (4-3)$$

其中 Δ 表示一个极小值以保证 $|p - Y + \Delta| > 0$ 。

之后我们利用 r 更新 Θ_{MDP} 。MDP 过程的损失函数为：

$$\ell(V(s), r) = \sum_{i=1}^t ((y - V(r_i))^2) \quad (4-4)$$

4.1.2 预测过程

推断时，模型接收句子对 Q, D 以及算法 4 中得到的值函数 V 作为输入，输出预测结果 Y 。

在推断时，利用算法 3 为样本 Q, D 生成一条路径：

$$\{[q_1, d_1], [q_2, d_2], \dots, [q_t, d_t]\} \quad (4-5)$$

并将生成的路径输入到路径判别模型中，利用公式 3-15 计算得到最终的匹配概率。

Algorithm 5 Inference Process of VIM

Input: sentence pair $D = \{(Q^{(n)}, D^{(n)})\}_{n=1}^N$, value function V

Output: label Y

$s \leftarrow [1, 1]$

generate path according to Alg 3

compute Y by Eq. 3-15

return Y

4.2 实验数据及评价指标

4.2.1 实验数据

为了对 VIM 的有效性进行评估，我们利用 Quora 的问题匹配数据集进行了测试。Quora 是美国的在线知识问答平台，每天会产生大量的相似问题。该数据集即为 Quora 从其问答平台上收集到并进行了人工标注的匹配问题数据。

该数据集共包含约 40 万数据集，句子长度从 3 到 100 不等，能够较好的对模型的有效性进行评估。由于该数据集的样本量过大，我们从样本中挑选出了长度在 8 到 10 的约 6 万个句子对作为数据集，选择其中的 45056 个句子对作为训练集，22528 个句子对作为验证集，4441 个句子对作为测试集对我们的算法进行了测试。

4.2.2 评价指标

本节介绍用于文本匹配算法的评价指标。在本文中，文本匹配可被看作一个分类问题，判断一个二分类预测模型优劣的评价指标主要有：精确率（Precision），召回率（Recall），F 值（F-score），准确率（Accuracy, ACC），受试者工作特征曲线（Receiver Operating Characteristic, ROC），AUC（Area Under Curve of ROC）。本文使用的是准确率，F1 和 AUC 这三个指标。

	预测为正样本	预测为负样本
实际为正样本	被模型预测为正的样本 True Positive TP	被模型预测为负的正样本 False Negative FN
实际为负样本	被模型预测为正的负样本 False Positive FP	被模型预测为负的负样本 True Negative TN

图 4-1 混淆矩阵

图4-1是混淆矩阵（Confusion Matrix）。本节出现的所有指标都是以该矩阵为依据。

准确率 ACC 是计算样本被正确预测的次数占总样本的比例，如式 (4-6)。准确率越高，算法在该指标上表现越好，当准确率为 1 时，该评价指标达到最优值。

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \quad (4-6)$$

精确率是被模型预测为正的样本中，实际为正样本的比例，召回率是实际为正的样本中，正确预测为正样本的比例。F₁ 是精确率和召回率的调和平均数，相当于精确率和召回率的综合评价指标，如式 (4-7)。F₁ 越大，算法在该指标上表现越好，当 F₁ 为 1 时，该评价指标达到最优。

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} \quad (4-7)$$

AUC 就是 ROC 曲线和坐标轴相交部分覆盖的面积大小，ROC 曲线由两个变量：真正样本率 TPR（True Positive Rate）和假正样本率 FPR(False Positive Rate) 绘制，其中 TPR 和 FPR 如式 (4-8)，分别反映了正样本和负样本的分类准确率，AUC 越大，算法在该指标上表现越好，当 AUC 为 1 时，该评价指标达到最优。

$$\begin{aligned} TPR &= \frac{TP}{TP + FN} \\ FPR &= \frac{FP}{FP + TN} \end{aligned} \quad (4-8)$$

4.3 实验结果及分析

对于文本匹配问题，目前已经有多种算法可以很好地解决该问题。本文选取了其中两个经典的文本匹配算法 MatchPyramid^[8] 以及 MatchSRNN^[9] 来与本文提出的算法进行比较。

表 4-1 文本匹配算法在 Quora 数据集上的测试结果

Algorithm	Evaluation Criterion		
	ACC	F1	AUC
VIM	0.7222(*)	0.7217	0.7979(*)
MatchPyramid	0.7130	0.7220(*)	0.7853
MatchSRNN	0.7105	0.7201	0.7848

在 Quora 数据集上，本文进行了算法的五折交叉验证。即将原始数据集分割成五份，每份大小相同。每一折实验选择其中的四份数据作为训练集，一份数据作为验证集。上述过程重复五次，即每份数据都被用做验证集。每一折实验都选取验证集上评价准则最高的参数进行测试，将测试结果取平均值作为最后的评估结果。从表4-2中可以发现 VIM 在 ACC 和 AUC 上均显著优于 MatchPyramid 和 MatchSRNN，在 F1 上三者相差并不大。

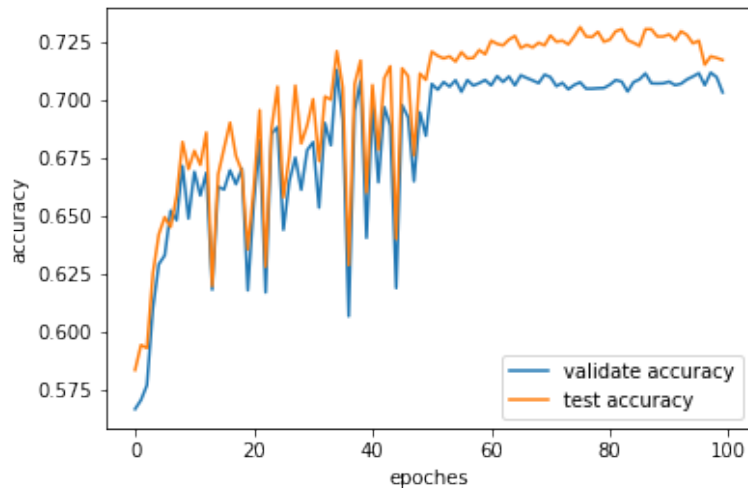


图 4-2 文本匹配算法的准确率变化曲线

根据在 QuoraQP 上的测试结果来看，使用价值迭代的方式训练马尔科夫决策过程的收敛速度很快，图4-2为我们的算法在上面试验中某一折数据的准确率

变化曲线，蓝色曲线为测试集的准确率，黄色曲线为验证集的准确率。可以发现我们的算法收敛速度很快，在第 50 轮左右就可以做到收敛。

为了测试向前看的单词个数 m 对于算法性能的影响，我们对向前看 2, 3, 4, 5 个单词分别进行了实验。实验结果如表4-2。

表 4-2 文本匹配算法在 Quora 数据集上的测试结果

向前看单词个数	Evaluation Criterion		
	acc	F1	auc
$k = 2$	0.7280	0.7425	0.8072
$k = 3$	0.7345(*)	0.7417(*)	0.8149(*)
$k = 4$	0.7311	0.7338	0.8112
$k = 5$	0.7280	0.7366	0.8067

可以发现 $m = 3$ 时算法的表现最优。这是因为一般情况来说，语言的组合结构问题构成的词语个数都是有限的，一般都在 3, 4 个左右。在 $m = 2$ 时，无法覆盖大部分的次序颠倒的场景；在 m 大于 3 时，算法的各项指标都开始逐渐下滑。这可能是因为在 m 过大后，截取到的句子长度远大于次序颠倒部分的长度，使得神经网络捕捉到的信息中更多的包含了序列信息，次序颠倒信息的比重下降。

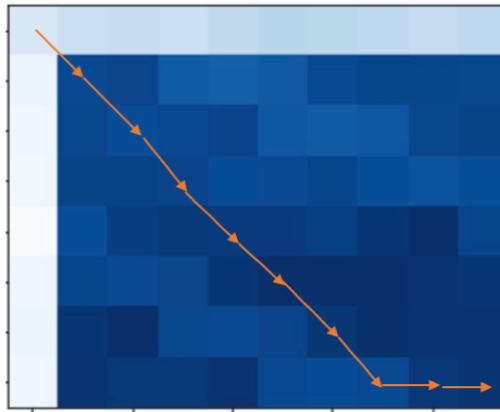


图 4-3 值迭代生成的路径

图4-3 展示了在输入样本为 $Q=\text{What are some examples of acidic substances?}$, $D=\text{What are some examples of substances soluble in water?}$ 值迭代的生成路径。由于每个方向的值都可以由其左边、上边、和左上的方块得到，因此我们在画图时

将 3 个值取平均作为这个方块的值。从图中我们可以明显发现，右下角为颜色最浓的部分，右上和左下的颜色相对来说都要淡很多。

但是由于值迭代基于贪心的策略导致算法很容易陷入过拟合，为了测试 VIM 算法的小数据集上的效果，我们在 Quora 数据集上选取了 10240 个句子对进行测试。测试方法同样采用五折交叉验证：

表 4-3 文本匹配算法在 Quora 数据集上的测试结果

Algorithm	Evaluation Criterion		
	acc	F1	auc
VIM with $\epsilon = 0$	0.6465	0.6644	0.7056
VIM with $\epsilon = 0.1$	0.6519	0.6772(*)	0.7080
VIM with changed ϵ	0.6532	0.6612	0.7147
MatchPyramid	0.6671(*)	0.6702	0.7237
MatchSRNN	0.6617	0.6736	0.7285(*)

从表格 4-3 可以发现，在小数据集下 VIM 算法出现了明显的过拟合，人工调节的 ϵ 可以小幅度提高算法的效果，但是仍然无法超过 MatchPyramid 和 MatchSRNN。

4.4 本章小结

本章主要介绍了文本匹配的马尔科夫决策过程。首先介绍了马尔科夫决策过程的背景，包括马尔科夫链以及马尔科夫决策过程的训练和推断，并且根据文本匹配的场景对文本匹配的 MDP 进行了形式化描述，并利用值迭代的方法进行训练和推断。该模型在大数据量的情况下表现优异，但是由于值迭代算法的缺陷，在小数据场景下表现不佳。在后续章节中，本文将针对该问题进行改进。

第五章 基于蒙特卡洛树搜索的文本匹配建模及实现

利用 VIM 可以解决大部分文本匹配的问题，但是 VIM 基于贪心的路径搜索难以解决语言的组合结构问题。对于词序的微小变化导致的语义差别，VIM 往往难以识别。而且 VIM 在小数据集下容易陷入局部最优而导致算法严重过拟合。而 AlphaGo 以及 AlphaGo Zero 通过引入蒙特卡洛搜索树很好地解决了路径搜索中的贪心导致的局部最优解问题。本章介绍了蒙特卡洛树搜索相关算法并提出了基于蒙特卡洛树搜索的文本匹配算法。

5.1 蒙特卡洛树搜索

5.1.1 蒙特卡洛树搜索介绍

蒙特卡洛树搜索^[18] (Monte Carlo Tree Search, MCTS) 是一种人工智能问题中做出最优决策的方法，一般是在组合博弈中的行动的规划形式，它结合了随机模拟的一般性和树搜索的准确性。通过随机的对游戏进行推演来逐渐建立一棵不对称的搜索树的过程。

蒙特卡罗树搜索每次循环包含四步：

1. 选择 (Selection)：从根节点 R 开始，一次选择最佳的子节点，直到达到叶节点。选择最佳子节点的难点主要在于探索和利用的平衡：探索可以保证充分探索解空间，以便找到未访问的点中受益较大的点；利用则能充分利用现有模拟结果以加快搜索速度。

解决该平衡问题的方法被称为用于树搜索的上限置信区间算法^[19] (Upper Confidence Bounds for trees, UCT) 该方法基于上限置信区间算法^[20] (Upper Confidence Bounds 1, UCB1)，建议在选择子节点时，最大化 $\frac{w_i}{n_i} + c \sqrt{\frac{\ln N_i}{n_i}}$ 为目标。式中 w_i 表示第 i 次移动后的取胜次数（利用）； n_i 表示第 i 次移动后的仿真次数（探索）； N_i 表示总仿真次数， c 表示探索参数。

2. 扩展 (Expansion)：如果游戏没有结束，那么扩展叶节点，将一个或多个可用节点添加为该叶节点的子节点，并选择其中的一个子节点作为 C。
3. 仿真 (Simulation)：从节点 C 开始，用随机策略选择节点进行游戏。
4. 反向传播 (Back propagation)：根据仿真的结果更新从 C 到 R 的节点。

5.1.2 围棋中的蒙特卡洛树搜索应用

AlphaGo 算法中存在 3 个网络：使用监督学习训练的策略网络 p_σ 以及该网络的简化版 p_π ；使用强化学习训练的策略网络 p_ρ 以及价值网络 v_θ 。在树搜索的过程中， p_σ 被用作模拟人类棋手落子； p_ρ 决定落子位置， v_θ 判断当前局势。

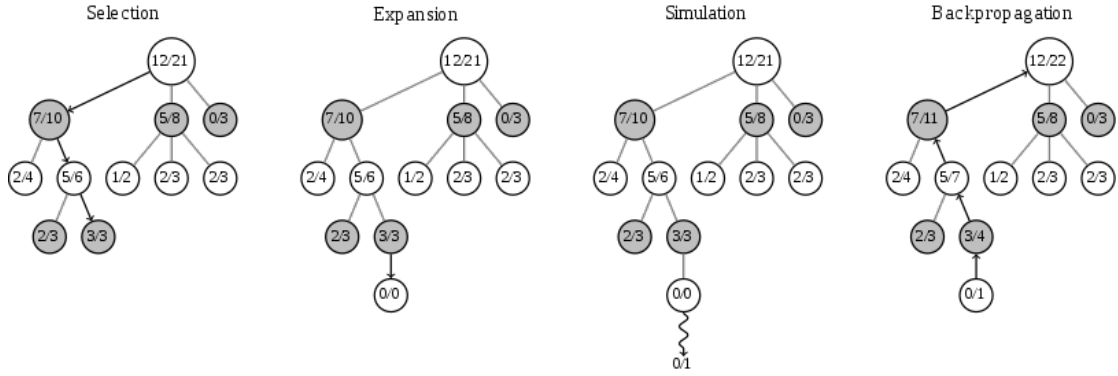


图 5-1 蒙特卡洛树搜索步骤

p_σ 的训练数据来自于 KGS 的三千万棋局，利用每个棋局及其对应的落子的数据，训练得到概率分布 $p_\sigma(a|s)$ ，即在当前状态（棋局）下采取行动（落子）的概率。由于 p_p 网络结构比较复杂，推导速度较慢，因此在 AlphaGo 算法训练时采用简化版的 p_π 以降低训练速度；推导时采用 p_σ 以提高胜率。

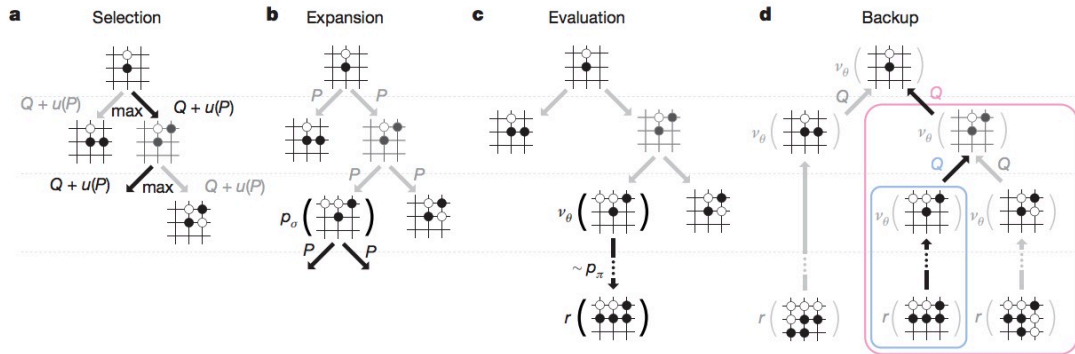


图 5-2 AlphaGo 的树搜索过程

蒙特卡洛树搜索使用随机进行仿真，Alpha Go 利用价值网络辅助策略网络作为落子参考进行仿真。图5-2中每一条边 (s, a) 都包含了行动价值 $Q(s, a)$ ，该节点的访问次数 $N(s, a)$ 以及先验概率 $P(s, a) = p_\sigma(a|s)$ 。每次仿真从根节点开始根据当前状态 s_t 选择行动 a_t 直至到达叶节点。

$$a_t = \arg \max_a (Q(s_t, a) + u(s_t, a)) \quad (5-1)$$

$$u(s_t, a) \propto \frac{P(s, a)}{1 + N(s, a)}$$

叶节点的重要程度由价值网络 v_θ 和策略网络 p_π 输出 z_L 确定。

$$V(s_L) = (1 - \lambda)v_\theta(s_L) + \lambda z_L \quad (5-2)$$

模拟结束时，所有边的行动价值以及参观次数都会被更新。

$$\begin{aligned} N(s, a) &= \sum_{i=1}^N \mathbf{1}(s, a, i) \\ Q(s, a) &= \frac{1}{N(s, a)} \sum_{i=1}^N \mathbf{1}(s, a, i) V(s_L^i) \end{aligned} \quad (5-3)$$

式中 s_L^i 表示第 i 次仿真的叶节点， $\mathbf{1}(s, a, i)$ 表示边 (s, a) 是否被访问过。

AlphaGo Zero 是对 AlphaGo 的改进。相对于 AlphaGo，AlphaGo Zero 主要改进了以下几点：

1. AlphaGo Zero 是一个自训练的强化学习过程，直接用棋盘状态作为训练样本，不需要任何人类的先验知识，
2. AlphaGo Zero 采用了新的网络架构。将 Alpha Go 中的策略网络和价值网络融合成为一个网络，因为策略网络和价值网络的输入层和网络隐层的结构完全一样，只不过前者输出落子概率向量，后者输出当前局面的赢棋概率。
3. AlphaGo Zero 采用了深度残差网络（Deep Residual Network）缓解了卷积神经网络中的梯度消失问题，网络深度加深，扩展了网络的解释能力。

AlphaGo Zero 使用了一个参数为 θ 的深度网络 f_θ ，该网络将棋盘作为输入，输出对应的概率和值函数 $(\mathbf{p}, v) = f_\theta(s)$ ，式中 \mathbf{p} 表示行动（落子）的概率 $p_a = \text{Pr}(a|s)$ ， v 是一个标量，表示当前棋局的胜率。

深度网络的训练仍然采用蒙特卡洛树搜索的方法。在时刻 t 使用 f_θ 的输出作参考进行下一轮蒙特卡洛树搜索，每一轮的树搜索都会输出当前状态（棋局）下每个行动（落子）的概率 π 蒙特卡洛树搜索得到的落子概率比直接 f_θ 得到的落子概率更强，因此蒙特卡洛树搜索可以被认为是一个策略提升（Policy Improvement）的过程。利用 π 进行落子后，不断循环至到期具结书。在棋局结束后，将对弈者的胜负情况 z 作为价值更新参数。最后更新 f_θ 的网络参数以让 f_θ 的输出的落子概率和胜率 (\mathbf{p}, v) 尽可能接近 (π, z) ，并在后面的棋局中使用新的参数来进行自对弈。

f_θ 的损失函数为：

$$\ell = (z - v)^2 + \pi^T \log \mathbf{p} + c \|\theta\|^2 \quad (5-4)$$

其中 c 是正则化的系数。

蒙特卡洛树搜索以及参数更新的相关部分参考5.1.1节。

5.2 蒙特卡洛树搜索增强的马尔可夫决策过程的文本匹配建模

再上一章，本文对文本匹配算法的 MDP 进行了介绍。受到 AlphaGo 和 Alphaha 的启发，本章尝试将蒙特卡洛树搜索方法应用于文本匹配中。我们将本章

提出的方法称为 MM-Match (MCTS enhanced MDP for Matching)。

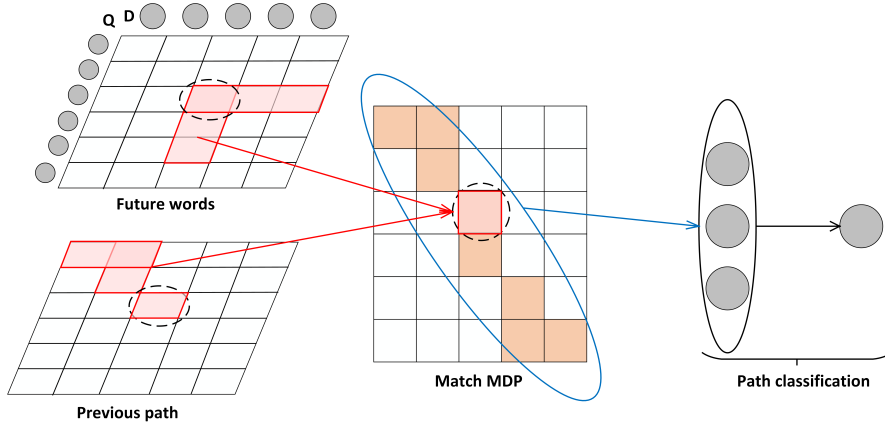


图 5-3 MM-Match 算法结构

5.2.1 蒙特卡洛树搜索增强的 MDP

本节主要介绍对 3.3 节所提到的 MDP 过程的改进。算法的整体流程和 3.3 节相比没有太大改进，仍然是先生成匹配路径之后基于生成的路径判断是否匹配。

1. 状态 \mathcal{S} : 我们将在 t 时刻的状态 s_t 定义为当前的路径以及从当前位置开始向前看的 k 个单词:

$$\begin{aligned} \mathbf{Q}_t &= \{\mathbf{u}_{q_1}, \mathbf{u}_{q_2}, \mathbf{u}_{q_3}, \dots, \mathbf{u}_{q_t}\}, \\ \mathbf{D}_t &= \{\mathbf{v}_{d_1}, \mathbf{v}_{d_2}, \mathbf{v}_{d_3}, \dots, \mathbf{v}_{d_t}\}, \\ \mathbf{F}_t &= \{[\mathbf{u}_{q_t}, \mathbf{v}_{d_t}], [\mathbf{u}_{1+q_t}, \mathbf{v}_{1+d_t}] \dots, [\mathbf{u}_{k+q_t}, \mathbf{v}_{k+d_t}]\}, \\ s_t &= [\mathbf{Q}_t, \mathbf{D}_t, \mathbf{F}_t] \end{aligned} \quad (5-5)$$

式中 q_t 和 d_t 仍然表示当前位置， \mathbf{Q}_t 和 \mathbf{D}_t 表示输入的句子对 Q, D 的前 q_t 和 d_t 按照路径拉伸得到的序列， \mathbf{F}_t 表示从当前位置 q_t, d_t 向前看 k 个单词得到的序列。

2. 动作 \mathcal{A} : 在每个时间 t : 向包括右、下、右下三个方向中的某个移动一步。
3. 状态转移函数 $\mathcal{T}(\mathcal{S}, \mathcal{A})$: 状态转移函数 $\mathcal{T}: \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ 被定义为:

$$\begin{aligned} \mathbf{Q}_t &= \{\mathbf{u}_{q_1}, \mathbf{u}_{q_2}, \mathbf{u}_{q_3}, \dots, \mathbf{u}_{q_t}\}, \\ \mathbf{D}_t &= \{\mathbf{v}_{d_1}, \mathbf{v}_{d_2}, \mathbf{v}_{d_3}, \dots, \mathbf{v}_{d_t}\}, \\ \mathbf{F}_t &= \{[\mathbf{u}_{q_t}, \mathbf{v}_{d_t}], [\mathbf{u}_{1+q_t}, \mathbf{v}_{1+d_t}] \dots, [\mathbf{u}_{k+q_t}, \mathbf{v}_{k+d_t}]\}, \\ s_t &= [\mathbf{Q}_t, \mathbf{D}_t, \mathbf{F}_t] \end{aligned} \quad (5-6)$$

在每个时间 t : 系统根据当前的状态选择一个行动 (移动方向)，并根据方向移动到下一个位置，状态随之转移：将当前位置两个句子的词向量分别

添加到对应路径的末尾，更新未来单词序列。完成转移后，再根据当前状态继续移动。

4. 值函数 V : 值函数 $V: S \rightarrow \mathbb{R}$ 是对当前模型是否可以得到正确预测结果的预测。值函数需要不断更新以更加准确的得到预测结果。本节提出的值函数需要用到 2 个 LSTM 将 Q_t 和 D_t 映射为 2 个一维向量，并将结果定义为对这两个向量加权和非线性转换：

$$V(s) = \sigma(\langle \mathbf{w}, \mathbf{g}_v(s) \rangle + \mathbf{b}_v) \quad (5-7)$$

式中 \mathbf{w}, \mathbf{b}_v 是需要学习的参数， $\sigma(x) = \frac{1}{1+e^{-x}}$ 是 sigmoid 函数以输出一个 0 ~ 1 之间的概率， $\mathbf{g}_v(s)$ 是对将 LSTM $_Q$ 以及 LSTM $_D$ 的最后一层隐向量连接得到的：

$$\mathbf{g}_v(s) = [\text{LSTM}_Q(\mathbf{Q}_t)^T, \text{LSTM}_D(\mathbf{D}_t)^T]^T \quad (5-8)$$

LSTM 网络的定义和公式 ?? 一致，LSTM $_Q$ 和 LSTM $_D$ 共享参数。

5. 策略函数 π : $\pi(s)$ 将环境状态作为输入，输出每个行动的概率。策略函数将 $\mathbf{g}_v(s)$, F_t 作为输入，使用一个 LSTM 将 F_t 映射为一个一维向量，并将其和 $\mathbf{g}_v(s)$ 连接，将连接后的向量利用 softmax 函数进行归一化得到每个行动的概率分布：

$$\pi(a|s) = \frac{\exp\{\Phi(a)^T \mathbf{U}_\pi \mathbf{g}_\pi(s)\}}{\sum_{a' \in \mathcal{A}(s)} \exp\{\Phi(a')^T \mathbf{U}_\pi \mathbf{g}_\pi(s)\}} \quad (5-9)$$

$$\mathbf{g}_\pi(s) = [\mathbf{g}_v(s), \text{LSTM}_F(\mathbf{F}_t)^T]^T$$

5.2.2 匹配路径的判别

在 3.3.2 中，我们提出了一种针对于路径的判别模型判断路径是否匹配。但是由于该方法在每次运行时的计算图都不尽相同，因此效率很低。本节对该方法进行了改进以提高其训练和推断速度。

判别模型将 \mathbf{Q}_t 和 \mathbf{D}_t 作为输入，并使用两个 LSTM 将其映射为 2 个矩阵，最后使用一个 LSTM 接受两个矩阵作为输入，输出一个一维向量：

$$\begin{aligned} \mathbf{g}_q &= \text{LSTM}_Q(Q_t) \\ \mathbf{g}_d &= \text{LSTM}_D(D_t) \\ \mathbf{g}_c^{\text{in}} &= [[\mathbf{g}_{q,1}, \mathbf{g}_{d,1}], [\mathbf{g}_{q,2}, \mathbf{g}_{d,2}], \dots, [\mathbf{g}_{q,t}, \mathbf{g}_{d,t}]] \\ \mathbf{g}_c^{\text{out}} &= \text{LSTM}_C(\mathbf{g}_c^{\text{in}}) \end{aligned} \quad (5-10)$$

LSTM $_Q$ 和 LSTM $_D$ 共享参数。最后输出概率会通过 sigmoid 函数变换概率输出：

$$p_m = \frac{1}{1 + e^{-\mathbf{g}_c^{\text{out}}}} \quad (5-11)$$

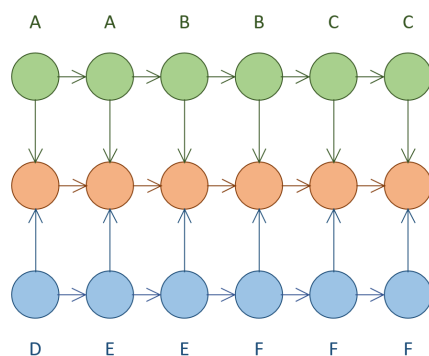


图 5-4 路径判别函数

5.2.3 蒙特卡罗树搜索的训练和推断

在4.3节，本文通过在大数据集和小数据集上的训练结果发现单纯将策略函数的输出作为行动概率很容易陷入局部最优解，即使使用 ϵ -贪心的方式也很难摆脱。受 AlphaGo 和 AlphaGo Zero 的启发，本节使用蒙特卡罗树搜索的方式进行前向式搜索。在 t 时刻，算法在策略和价值网络的辅助下执行一次蒙特卡罗树搜索，得到搜索策略 π ，并根据 π 确定下一个行动的位置。

Algorithm 6 TreeSearch

```

1: Input: root  $s_R$ , value  $V$ , policy  $\pi$ , search times  $K$ 
2: Output: Search policy  $\pi$ 
3: for  $k = 0$  to  $K - 1$  do
4:    $s_L \leftarrow s_R$ 
5:   {Selection}
6:   while  $s_L$  is not a leaf node do
7:      $a \leftarrow \arg \max_{a \in \mathcal{A}(s_L)} Q(s_L, a) + \lambda \cdot U(s_L, a)$  Eq. (??)
8:      $s_L \leftarrow$  child node pointed by edge  $(s_L, a)$ 
9:   end while
10:  {Evaluation and expansion}
11:   $v \leftarrow V(s_L)$  simulate  $v$  with value function  $V$ 
12:  for all  $a \in \mathcal{A}(s_L)$  do
13:    Expand an edge  $e$  to node  $s = [s_L.\mathbf{Q}_t, s_L.\mathbf{D}_t, s_L.\mathbf{F}_t]$ 
14:     $e.P \leftarrow p(a|s_L); e.Q \leftarrow 0; e.N \leftarrow 0$  init edge properties
15:  end for
16:  {Back-propagation}
17:  while  $s_L \neq s_R$  do
18:     $s \leftarrow$  parent of  $s_L; e \leftarrow$  edge from  $s$  to  $s_L$ 
19:     $e.Q \leftarrow \frac{e.Q \times e.N + v}{e.N + 1}$  Equation (??)
20:     $e.N \leftarrow e.N + 1; s_L \leftarrow s$ 
21:  end while
22: end for
23: {Calculate tree search policy. Eq. (??)}
24: for all  $a \in \mathcal{A}(s_R)$  do
25:    $\pi(a|s_R) \leftarrow \frac{e(s_R, a).N}{\sum_{a' \in \mathcal{A}(s_R)} e(s_R, a').N}$ 
26: end for
27: return  $\pi$ 

```

结 论

参考文献

- [1] Landauer T K, Foltz P W, Laham D. An Introduction to Latent Semantic Analysis [C]. 1998.
- [2] Blei D M, Ng A Y, Jordan M I. Latent Dirichlet Allocation [J]. Journal of Machine Learning Research, 2003, 3: 993–1022.
- [3] Huang P-S, He X, Gao J, *et al.* Learning deep structured semantic models for web search using clickthrough data [C]. In CIKM, 2013.
- [4] Hinton G E, Osindero S, Teh Y-W. A fast learning algorithm for deep belief nets [J]. Neural computation, 2006, 18 (7): 1527–1554.
- [5] LeCun Y, Bottou L, Bengio Y, *et al.* Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86 (11): 2278–2324.
- [6] Shen Y, He X, Gao J, *et al.* A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval [C]. In CIKM, 2014.
- [7] Hochreiter S, Schmidhuber J. Long Short-Term Memory [J]. Neural computation, 1997, 9 8: 1735–80.
- [8] Pang L, Lan Y, Guo J, *et al.* Text Matching as Image Recognition [C]. In AAAI, 2016.
- [9] Wan S, Lan Y, Xu J, *et al.* Match-SRNN: Modeling the Recursive Matching Structure with Spatial RNN [C]. In IJCAI, 2016.
- [10] Graves A, Fernández S, Schmidhuber J. Multi-dimensional Recurrent Neural Networks [C]. In ICANN, 2007.
- [11] Cho K, van Merriënboer B, Çaglar Gülçehre, *et al.* Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation [C]. In EMNLP, 2014.
- [12] Sutton R S, Barto A G. Reinforcement learning - an introduction [C]. In Adaptive computation and machine learning, 1998.
- [13] Mnih V, Kavukcuoglu K, Silver D, *et al.* Playing Atari with Deep Reinforcement Learning [J]. CoRR, 2013, abs/1312.5602.
- [14] Mnih V, Kavukcuoglu K, Silver D, *et al.* Human-level control through deep reinforcement learning [J]. Nature, 2015, 518 7540: 529–33.
- [15] Silver D, Huang A, Maddison C J, *et al.* Mastering the game of Go with deep neural networks and tree search [J]. Nature, 2016, 529 7587: 484–9.
- [16] Silver D, Schrittwieser J, Simonyan K, *et al.* Mastering the game of Go without human knowledge. [J]. Nature, 2017, 550 7676: 354–359.
- [17] Bellman R. A Markovian decision process [J]. Journal of Mathematics and Mechanics, 1957: 679–684.
- [18] Coulom R. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search [C]. In Computers and Games, 2006.

- [19] Kocsis L, Szepesvári C. Bandit Based Monte-Carlo Planning [C]. In ECML, 2006.
- [20] Auer P, Cesa-Bianchi N, Fischer P. Finite-time Analysis of the Multiarmed Bandit Problem [J]. Machine Learning, 2002, 47: 235–256.

外文资料

Here follows the English paper.

...

中文译文

这里就是外文资料的中文翻译。

...

致 谢

我就做了三件微小的事情，谢谢大家。