中文词语相似度计算

周奇安

院 （系）：计算机科学与技术学院　专　　业：计算机科学与技术

学　　号：　1130310402　指导教师：　孙大烈

2017 年 **5** 月

哈爾濱工業大學

毕业设计（论文）

题　　目　中文词语
相似度计算

专　　业　计算机科学与技术

学　　号　1130310402

学　　生　周奇安

指 导 教 师　孙大烈

答 辩 日 期　2017 年 5 月

摘　要

摘要是论文内容的高度概括，应具有独立性和自含性，即不阅读论文的全文，就能获得必要的信息。摘要应包括本论文的目的、主要研究内容、研究方法、创造性成果及其理论与实际意义。摘要中不宜使用公式、化学结构式、图表和非公知公用的符号和术语，不标注引用文献编号。避免将摘要写成目录式的内容介绍。

关键词：关键词 1；关键词 2；关键词 3；……；关键词 6（关键词总共 3 — 6 个，最后一个关键词后面没有标点符号）

Abstract

Externally pressurized gas bearing has been widely used in the field of aviation, semiconductor, weave, and measurement apparatus because of its advantage of high accuracy, little friction, low heat distortion, long life-span, and no pollution. In this thesis, based on the domestic and overseas researching……

Keywords: keyword 1, keyword 2, keyword 3, ……, keyword 6 (no punctuation at the end) 英文摘要与中文摘要的内容应一致，在语法、用词上应准确无误。

目　录

第 1 章 绪论

1.1 问题背景

词语的相似度计算是自然语言处理中的一项基本任务。它的目的是使用自动化的方法给出一个量化的指标，来衡量一对词语之间的相似程度。词语相似度计算对于很多高层次的自然语言处理任务有着重要的意义，例如问答、信息检索、改写检测与文本蕴含。

为了评价一种词语相似度计算方法的好坏，可以使用内部任务评价或外部任务评价。内部任务评价直接使用一个标准的词语相似度标注数据集，并使用一些评价指标来衡量机器给出的结果与标准数据集之间的差异；而外部任务评价则利用词语相似度计算的结果解决其他的高层次机器学习问题，通过衡量不同计算结果在这些问题上的性能来简介衡量词语相似度计算方法的好坏。

什么样的词语可以被称为“相似”的，这本身并不是一个定义良好的问题。词语的相似度非常依赖于人的主观判断。（写写每个数据集都是怎么搞的。）

本课题取自NLPCC-ICCPOL 2016会议中的“中文词语相似度计算”开放任务。该开放任务使用内部任务评价的方式来衡量各参赛队提交的结果。其组织者发布了一个PKU-500数据集，其中包含500个词语对，以及对每个词语对之间的相似度标注。共有49个参赛队伍报名参加了此次开放任务，而最终由21个参赛队伍提交了总计24个系统[1]。

1.2 相关工作

在英文词语相似度任务中，已经有多个数据集可以作为基准测试。其中，使用最广泛的是WordSim-353数据集。这个数据集包含353个词语对，并由16名标注者对其相似度进行了标注。

在NLPCC-ICCPOL 2016会议举办之前，Semeval-2012会议上也举行了一次中文词语相似度的比赛，比赛所使用的词语对是经过翻译的WordSim-353数据集，并重新进行了人工标注。遗憾的是，这次比赛只有两个队伍参加。

本次开放任务中的参赛队伍使用的方法大致分为三类：基于辞典或知识库的方法、基于语料的方法与基于信息检索的方法。

基于词典或知识库的方法利用了一些现有的语言学资源，例如知网

（HowNet）和同义词词林。这些词典或知识库的作者往往同时给出了用于计算词语相似度的规则。有些参赛队伍直接使用了这些规则，还有一些对这些规则进行了修改。这些方法依赖于词典或知识库的组织结构，并且无法处理未登录词。此外，人工制定的规则是否合理也是影响模型性能的重要因素。

基于语料的方法主要是根据上下文信息进行词嵌入，这类方法假设“相似的词语应该出现在相似的上下文中”。典型的词嵌入方法，例如word2vec，具有成熟的工具包，因此在各参赛队伍中非常受到欢迎。除了词嵌入方法以外，来自大连理工大学的参赛队使用了深度学习的方法来识别描述词语类比的句子，例如“寂寞和孤单的区别是什么”。如果这类句子出现在语料中，很可能意味着这两个词语是相关的。基于语料的方法需要大规模的语料库和漫长的训练时间，由于这些模型普遍使用无需人工标注的生语料库，因此语料的获取来源非常广泛。而训练时间的缩短只能通过使用运算能力更强的硬件，因此实验可能受到硬件条件的制约。针对中文语料而言，绝大部分的模型是以词为单位进行处理，这样分词的准确度也会对模型最终性能产生一定影响。

基于信息检索的方法利用搜索引擎查询包含目标词语的页面，根据同时包含两个词语的页面数、仅包含一个词语的页面数利用公式计算目标词语的相似度。这种方法的实现非常的简单，只需爬取搜索引擎的结果页面，或调用搜索引擎提供的API。有的队伍将这种方法作为多种评价方法之一，也有的队伍用信息检索得到的结果作为弱监督学习的指标。这种方法的一大问题是词语相似度和词语共同出现的频率并不一致，例如一些事物的别名经常在科普类的文章中被介绍（例如“西红柿”和“番茄”），而因口语习惯而导致不同的同义词（例如“拖后腿”和“拉后腿”）就很难在一篇文章中同时出现。

1.3 数据生成与评价指标

PKU-500数据集的生成过程分为三个步骤，分别为词语选择、词语对生成和相似度标注。

PKU-500数据集的词语选择过程遵循以下的条件：

领域 同时涉及传统书面语言与近期的网络语言。

频率 应同时包含高频、中频与低频的词语。

词性 应同时包含名词、动词与形容词。除上述实词外还应包含虚词（例如副词和连词）。

字数 应包含1–4个字的词语。

词义 应包含部分多义词。

极性 应同时包含正面词语与负面词语。

根据上述条件，词语的选择过程如下：

1. 数据来源于两个领域：三个月的《人民日报》新闻与一大批新浪微博数据。

2. 数据经过开源的ansj工具进行分词和词性标注。

3. 从两个领域中分别根据其频率、词性与字数抽取词语。

4. 根据词义与极性手动挑选自动抽取的词语，最终从《人民日报》新闻与微博数据中分别得到了514和202个词语。

上述步骤生成了716个单独的词语，接下来的步骤用于生成词语对：

1. 对于每一个词语，从哈工大同义词词林（扩展板）中自动提取三个候选词语：第一个属于同一个原子词群，第二个属于某一个父节点，第三个是从其它词语中随机选择的。

2. 每个目标词语的候选词语被进一步人工挑选，并根据上文中提到的条件去除与增加了一些词语，最终得到470个词语对。

3. 从WordSim-353数据集中选择30个翻译的词语对。

4. 最终，得到了500个用于中文词语相似度计算的词语对。

相似度标注过程由20个研究生完成，他们都以中文为母语，来自汉语言学专业。相似度的取值范围为[1, 10]，其中1表示词语完全不同，而10表示词语意义相同。最终的相似度得分是20个人类标注的平均值。标注者没有受到任何有关标注的指示，这意味着他们会根据自己对语言的理解来判断。同时，对相关性与相似度也不加区分。

上述过程得到的500个词语对全部作为测试数据，而不提供训练数据。在开放任务开始前，组织者先行公布了40个词语对作为样例。参赛系统可以使用任何技术，例如传统的基于语料分布的相似度、基于词典的相似度，与近期发展出来的词嵌入与深度学习方法。

为了防止对小数据集的过拟合，该开放任务的组织者将500个词语对混入了10000个从大词典中随机生成的词语对，并作为参赛系统的输入。

系统性能的评价指标是系统输出与人工标注之间的Spearman等级相关系数：

$$\rho = 1 - \frac{6\sum_{i=1}^{n}(R_{X_i} - R_{Y_i})^2}{n(n^2-1)} \tag{1-1}$$

其中，n是词语对的数量，而R_{X_i}与R_{Y_i}分别是系统输出与人工标注相似度的等级变量（排名）的标准差。

1.4 本课题研究方法

本课题的主要研究三种方法。一是深度学习方法，二是集成学习方法，三是基于词典或知识库的方法的改进。上述方法将直接用于词语相似度计算，或用于改进与结合现有的方法。

1.4.1 深度学习方法

本课题中使用的深度学习方法是一种基于语料的方法。本课题利用循环神经网络及其变种构造分类器。语料库中的所有句子经过一定的处理之后作为输入，其中将句子不加变化地输入循环神经网络作为正例，而将每个句子中挑选一个词语替换成其它的词语作为负例。分类器的训练目标是尽可能准确地识别被修改的句子。最终评价词语相似度时，将目标词语所在的句子中的目标词语替换成词语对中的另一个词语，并利用上文中的分类器进行识别，使用分类器输出的结果和置信度作为词语相似度的评价标准。

此外，课题还将使用深度学习方法辅助词嵌入的训练。这种方法训练一个带有嵌入矩阵的双向循环神经网络，用来解决词预测问题。令神经网络通过词语两侧的上下文信息补全每一个空缺词。为了解决需要在整个单词表的范围内做规范化的问题，可以使用噪声对比估计的方式。

1.4.2 集成学习方法

词语相似度的计算具有多种方法，为了结合这些方法的优势，提高模型的准确率与稳定性，需要利用集成学习的方法。集成学习的方法是通过一些结合策略，将多个个体学习器的输出结合成最终的输出结果。集成学习中最常见的方法是简单平均法和加权平均法。除此之外，在其他参赛选手的系统中还出现了很多人为制定的基于规则的结合策略。这些方法虽然简单朴素，甚至有些看起来缺乏理论根据，但是在词语相似度计算的任务中取得了相当优秀的成绩。因此作为对比，本课题也会对这些方法进行尝试。另外一种比较高级的集成学习方法是学习法，这种方法使用另一个学习器来进行结合。这个学习器以每个初级学习期的输出作为输入，而输出一个单个的结果。这样就可以使用初级学习器在测试集上的输出和测试集的标注训练次级学习器。由于官方给出的测试集规模较小，因此不能使用基于深度学习的方法构造次级学习器。本课题将会选用较为简单的模型用于集成学习。由于可供测试的数据只有宝贵的500个词语对，本课题将使用交叉验证甚

至留一法的方式训练次级学习器。这样所有的数据既能参与训练，又能用于评测。而且，进行交叉验证所训练出的学习器还可以再次使用简单平均法集成，形成一个最终的模型。

1.4.3 基于词典或知识库的方法的改进

基于词典或知识库的方法往往利用词典和知识库作者提供的相似度计算规则。本课题将会利用词典或知识库的结构提取出特征，然后用机器学习的方式计算相似度。由于这里也需要使用测试集来训练，因此同样需要选择更为简单的模型。由于基于词典或知识库的学习器和集成学习的次级学习器相近，因此也可以用一个统一的学习器，输入深度学习模型的输出与词典或知识库中的特征，输出最终的结果。

第 2 章 深度学习的二分类方法

中文词语相似度计算本身是一个无监督的回归问题，这增加了解决问题的难度。如果能够将无监督学习转化为有监督学习，并且将回归问题转化为分类问题，就可以利用大量已知的模型，问题解决的难度也将会大大降低。

本章将介绍一种方法，这种方法利用词语在句子中的可替换性计算词语相似度。如果一个“通顺”的句子中的一个词语被替换之后，形成的句子仍然较为“通顺”，则本方法认为原词语和替换的词语较为接近；否则，认为两词语差距较大。本方法从未经处理的生语料库中抽取句子生成数据，并训练一个辅助的二分类器以判定句子的“通顺”与否。二分类器使用深度学习模型构建，以提高模型对整个句子的理解能力。

2.1 原理介绍

为了更清晰的介绍深度学习的二分类方法，本文将先给出一些形式化的定义。

在深度学习的二分类方法中使用的语料库C是一个句子的多重集合，其中每一个句子是一个词语的序列，即：

$$C \subset \mathcal{V}^{+} \tag{2-1}$$

其中$\mathcal{V}$是所有词语的集合。

替换函数f_{r}可以将一个句子$s \in C$中的某个词语s_i替换为一个不同的词语$v \neq s_i$，即：

$$f_{\mathrm{r}}(s, i, v) = (s_1, \ldots, s_{i-1}, v, s_{i+1}, \ldots, s_{|s|}) \tag{2-2}$$

对语料库中的句子应用替换函数f_{r}得到的句子称为替换句：

$$C_{\mathrm{r}} = \{ f_{\mathrm{r}}(s, i, v) \mid s \in C, v \in \mathcal{V} \} \tag{2-3}$$

一般来说，$C_{\mathrm{r}} \cap C \approx \emptyset$。本课题希望得到一个二分类器$M_C$，满足：

$$M_C(s) = \begin{cases} 1 & s \in C, \\ 0 & s \in C_{\mathrm{r}} \end{cases} \tag{2-4}$$

使得M可以区分原句与替换句。

利用这样的二分类器，我们就可以定义词语对$(v_{\mathrm{a}}, v_{\mathrm{b}})$的相似度：

$$\mathrm{sim}(w_\mathrm{a}, w_\mathrm{b}) = E\Big(M_C(f_\mathrm{r}(s, i, v_\mathrm{b})) \,\Big|\, s \in C, s_i = v_\mathrm{a}\Big) \tag{2-5}$$

其中E为数学期望。其中的思想是，利用包含词语v_a的句子以v_b替换得到的替换句测试二分类器M_C，若二分类器容易将这样的替换句误判为原句，则说明词语对$(v_\mathrm{a}, v_\mathrm{b})$相似。为了提高准确性，应同时考虑从$v_\mathrm{a}$到$v_\mathrm{b}$和从$v_\mathrm{b}$到$v_\mathrm{a}$的替换。

2.1.1 训练数据生成

为了训练二分类器M_C，我们需要一些带有标注的句子$(x, y) \in \mathcal{V}^+ \times \{0, 1\}$作为训练数据。其中：

$$y = \begin{cases} 1 & x \in C, \\ 0 & x \in C_\mathrm{r} \end{cases} \tag{2-6}$$

原句的生成是非常容易的，只需将M_C中全部的句子标注为1即可。而替换句的生成需要一些额外的步骤。

利用随机采样的方法，我们可以生成一些采样句。采样句定义如下：

$$C_\mathrm{s} = \Big\{ f_\mathrm{r}(s, i, v) \,\Big|\, s \in C, i \sim \mathcal{U}\{1, |s|\}, v \sim \mathcal{D}_\mathcal{V} \Big\} \tag{2-7}$$

其中$\mathcal{U}$为均匀分布，$\mathcal{D}_\mathcal{V}$为词语在自然语言中出现的概率分布。也就是说，采样句中的被替换词语位置i是随机选择的，而替换词语v亦是从词语分布中采样得到的。

显然，$C_\mathrm{s} \subset C_\mathrm{r}$，而随机采样对于计算机来说是轻而易举的任务。这样，我们就可以通过生成采样句的方式获得用于训练的替换句。

2.1.2 循环神经网络

上文提到，本课题需要训练一个二分类器M_C用于区分原句和替换句。二分类器的输入是一个句子$x \in \mathcal{V}^+$，而输出为$y' = M_C(x) \in \{0, 1\}$。

循环神经网络（RNN）是一种具有循环连接的神经网络。这类神经网络可以用于进行时间序列的处理，因此常用于自然语言处理领域。

由于各类神经网络均以向量作为输入，因此需要将句子中的词语转化为向量，也就是词嵌入。词嵌入的形式非常简单，设有词嵌入矩阵$E^{|\mathcal{V}| \times n_\mathrm{embed}}$，则词嵌入的结果为：

$$e_i = E_{x_i} \tag{2-8}$$

其中n_embed是词嵌入维数。由于目前缺少成熟的预训练的中文词嵌入，所以本课题将词嵌入矩阵E作为可训练参数，与网络参数同时接受优化算法的优化。

循环神经网络的结构如图2-1所示。其中，e_i如上文所述，作为循环神经网络的输入，而h_i是循环神经网络的输出。

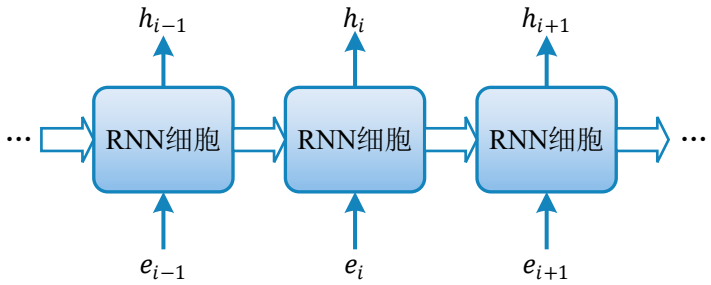

图 2-1 RNN结构

RNN细胞可以看作是一个带有可训练参数的函数，每个RNN细胞的可训练参数共享。相邻的RNN细胞之间会传递一些信息，信息的内容取决于RNN细胞的种类。

2.1.3 LSTM

现代的循环神经网络具有多种变种，其中非常著名的一种即是[2]中提出的LSTM（Long short-term memory）。其经过了[3]改进，形成了现在常见的LSTM形式。本课题中即采用上述LSTM模型构建二分类器。

LSTM与其它RNN的不同之处在于其独特的细胞结构，如图2-2所示。LSTM的最大特点是其优秀的“记忆能力”，即能处理时间序列中间隔很长的序列。这得益于LSTM中加入的遗忘门、输入门和逻辑门的设计。

图2-2中的c_i是LSTM细胞间的隐藏状态，f_i为遗忘门，i_i为输入门，o_i为输出门。公式2-9 – 2-13给出了一个LSTM细胞的完整定义，其中W、U、b都是可训练参数，而σ是sigmoid函数，$\circ$是逐项乘积。

$$f_i = \sigma(W_\mathrm{f} e_i + U_\mathrm{f} h_{i-1} + b_\mathrm{f}) \tag{2-9}$$

$$i_i = \sigma(W_\mathrm{i} e_i + U_\mathrm{i} h_{i-1} + b_\mathrm{i}) \tag{2-10}$$

$$o_i = \sigma(W_\mathrm{o} e_i + U_\mathrm{o} h_{i-1} + b_\mathrm{o}) \tag{2-11}$$

$$c_i = f_i \circ c_{i-1} + i_i \circ \tanh(W_\mathrm{c} e_i + U_\mathrm{c} h_{i-1} + b_\mathrm{c}) \tag{2-12}$$

$$h_i = o_i \circ \tanh(c_i) \tag{2-13}$$

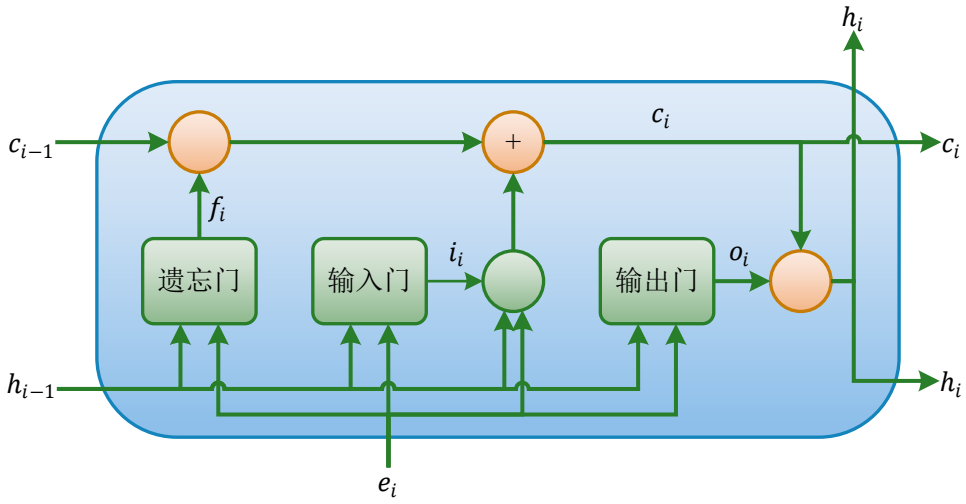

图 2-2 LSTM细胞结构

2.1.4 深层LSTM

与其它深度学习模型类似，LSTM网络也可以增加模型的层数以加强模型的学习能力。文献[4]中给出了多种LSTM与RNN层叠加的方案。本课题使用两个LSTM层叠加的方式，使一个LSTM层的输出向量作为另一个LSTM层的输入向量，如图2-3所示。

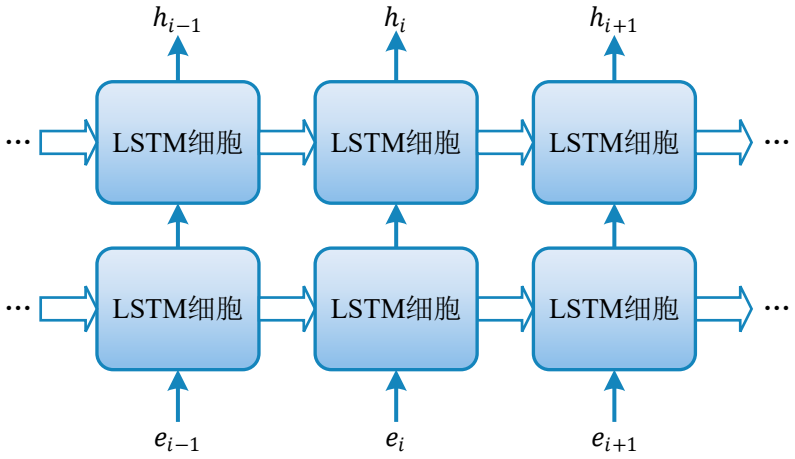

图 2-3 深层LSTM

2.1.5 分类结果生成

LSTM网络的输出向量最终需要转换为一个二元的输出$y' \in \{0,1\}$，考虑到使用连续变量能够更容易进行运算，本课题使$y' \in [0,1]$。

本课题中使用逻辑回归来完成上面的任务。逻辑回归函数如公式2-14所示：

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{2-14}$$

其中β_0与β_1都是可训练参数。本课题取LSTM网络的最后一个输出为整个网络的输出向量，并应用逻辑回归函数得到分类结果。即：

$$y' = F(h_{|s|}) \tag{2-15}$$

为了提高模型性能，本课题还尝试在LSTM输出后增加一个全连接层。此时输出的二分类结果如下：

$$y' = F(\sigma(W_{\text{full}} h_{|s|} + b_{\text{full}})) \tag{2-16}$$

2.1.6 模型训练

为了训练一个深度学习，需要定义一个损失函数，并且使用一种优化方法来最小化损失函数的值。

损失函数是一种用于衡量预测值y'与真实值y之间的偏差的函数。当$y' = y$时，$L(y', y) = 0$；而当$y' \neq y$时，$L(y', y) > 0$。一般而言，预测值与真实值之间的偏差越大，损失函数的值也应越大。这样使用数值优化的方法来最小化损失函数的值，就可以得到预测更为真实的模型。

在损失函数的选择方面，本课题选择了二分类问题常用的对数损失函数。对数损失函数的形式如下：

$$L(y', y) = -(y \log(y') + (1 - y) \log(1 - y')) \tag{2-17}$$

由于采样句的数量可能与原句不相等，因此会形成不平衡数据集。为了将数据集平衡化，在汇总训练数据的损失函数值时应为每个训练数据赋予一个权重$y(k-1)+1$。其中，k是训练集中采样居与原句的比例。这样，原句和采样句在训练中占有的权重就是相等的了。最终汇总的损失函数值如下：

$$\sum (y(k-1)+1) L(y', y) \tag{2-18}$$

深度学习模型的训练大多使用基于梯度下降的优化算法。传统的随机梯度下降方法需要人为指定一个学习速率，并且在训练中途还可能需要进行学习速率的调整，这会给实验带来更多的麻烦。而一些自适应学习速率的优化算法则可以在

学习过程中自行调节学习速率，例如Adagrad[5]、Adadelta[6]、RMSprop[1]与Adam[7]。为了降低实验的复杂程度，本课题使用了[8]中推荐的Adam算法进行模型训练。

2.2 程序实现

为了完成深度学习的二分类方法，本项目实现了一个可以处理原始语料、构建模型并进行训练的系统，其结构如图2-4所示。

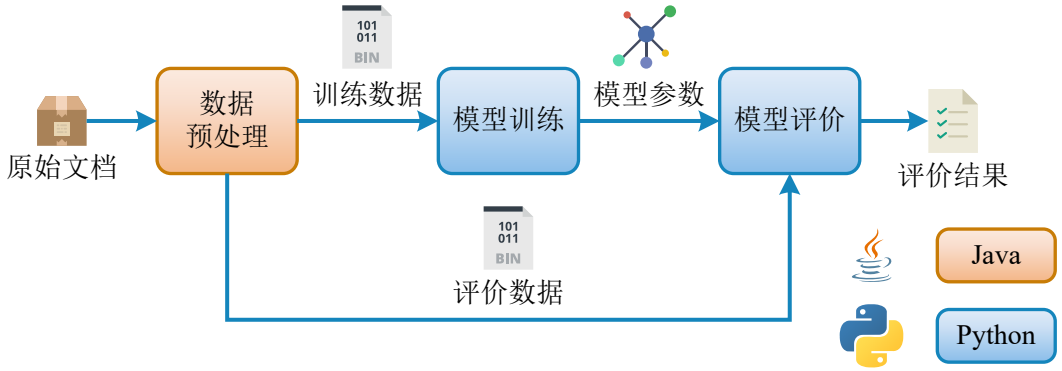

图 2-4 整体结构

程序实现分为数据预处理、模型训练和模型评价三个模块，根据模块特点的不同，分别使用了Java和Python两种编程语言实现。

2.2.1 数据预处理

由于Python的文本处理性能较差，会花费过长的时间，数据预处理的程序使用Java语言编写。预处理程序的结构如图2-5所示。

本课题使用了清华大学THUCTC工具包中提供的THUCNews数据集作为语料库。该语料库中的内容为从新浪新闻中得到的新闻文档，属于无标注的生语料库，以文本形式存储在文件中。

为了提高模型的准确度，本课题的深度学习模型没有采用一些早期深度学习工作中常用的直接将文档切分为固定长度序列的方式，而是以句为单位进行训练。句子的识别只需要根据简单的标点符号规则，本课题最初实现的句子分割器与后来使用的LTP平台均是如此实现。

接下来对句子进行分词和词性标注。本课题尝试了多种现有的分词工具，包括Java语言的Ansj，ltp4j与Python语言的结巴分词。经过对比分词工具对测试集中

[1]RMSprop并不是一个在论文中发表的算法，它是在多伦多大学的一个课堂讲义（http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf）中被提出的。

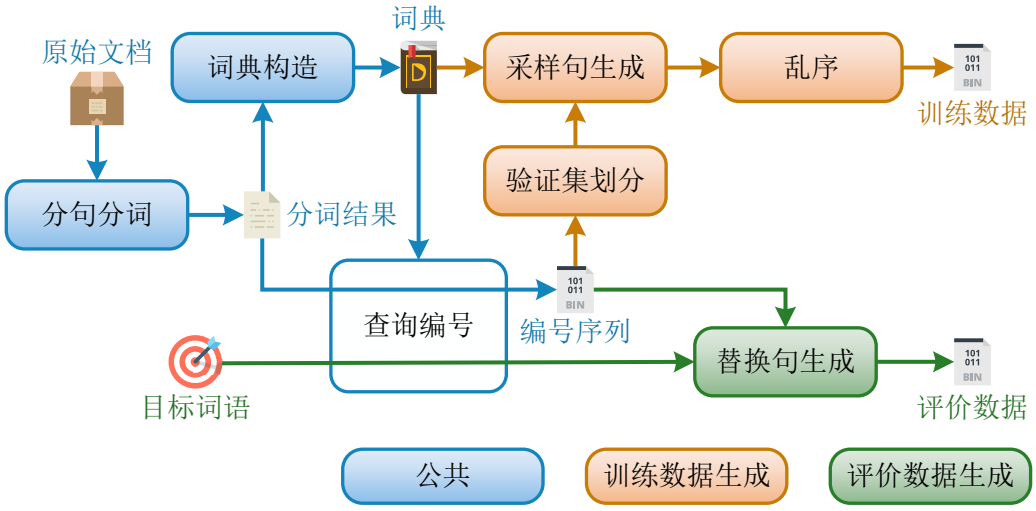

图 2-5 数据预处理

词语的切分准确度，最终确定使用Ansj的NLP分词模式。由于部分分词工具的运行速度十分缓慢，分词和词性标注的结果以序列化的方式存储在磁盘上，以备后续步骤使用。

分词和词性标注的结果会被程序读取两次。第一次读取的目的是构造词典，由于词汇量过大，会对词嵌入步骤带来严重的麻烦，本课题将词典中词语的数量限制在500000个，超出词典大小的词语以它们的词性来代替。为了方便词嵌入，词典中的每个词语均赋予一个整数编号。第二次读取数据则是为了将词语序列转换为编号序列，以备后续步骤使用。

生成的序号序列会按照比例随机划分为训练集和验证集。训练集和验证集的后续处理步骤相同。

采样过程如上文中采样句的定义所述，会随机挑选被替换词和替换用词进行替换。在实际实验中，每个原句产生了5个采样句。

为了使学习的结果更加稳定，上述步骤得到的结果最终会被乱序处理。由于数据量过大，乱序处理无法在内存中进行。因此本课题使用了外存储器辅助完成乱序，如算法2-1所述。

为了方便基于TensorFlow框架的Python程序读取数据，数据预处理阶段使用TFRecords格式保存数据。这样就可以避免书写Python程序读取数据，而通过TensorFlow中定义的reader来完成，避免数据的读取成为速度瓶颈。

Data: 待乱序序列X
Result: 乱序后序列Y
for i *in* $[1, n_{split}]$ **do**
    以写入模式打开临时文件F_i;
end
for x *in* X **do**
    设i为$1 - n_{\text{split}}$之间的随机数;
    将x写入F_i;
end
for i *in* $[1, n_{split}]$ **do**
    以读取模式重新打开F_i;
    从F_i中读取全部内容Y_{split};
    在Y_{split}上执行内存乱序算法;
    for y *in* Y_{split} **do**
        输出y;
    end
end

算法 2-1 外存储器乱序算法

2.2.2 模型训练

模型训练使用了Python语言的TensorFlow框架。该框架既可以通过简洁的API直接生成神经网络层，也可以使用底层的矩阵操作对数据进行细致的调整。在运行时，TensorFlow框架运用了CUDA技术，可以充分利用计算机的GPU资源。

为了平衡训练的速度和稳定性，各类基于梯度下降的学习算法往往需要将数据划分为mini-batch，而每一个mini-batch的数据在TensorFlow平台中由一个多维数组来表述。这意味着在生成mini-batch时，需要将变长序列填补成固定长度的序列。利用TensorFlow框架提供的队列机制可以完成这一任务，它可以在取出内容时动态填补序列。每个序列需要一个单独的线程进行驱动，这将在后文中讲解。

在利用TensorFlow框架进行开发时，首先应进行数据的读取。由于数据预处理程序已经将数据输出为TFRecords格式，这里可以使用reader轻松地读取数据。

接下来应构建TensorFlow图。TensorFlow图由一系列的操作组成。每个操作接受一定数量（也可能没有）的Tensor作为输入，并且产生一个Tensor作为输出。

TensorFlow框架提供了包括Tensor变形、数学运算、神经网络层等API来应用操作。根据第2.1节中所列出的表达式，就可以构造出完整的TensorFlow图。在构造过程中，应该注意设置设备布局，将适合CPU计算的操作布局在CPU上布局，而将适合GPU计算的操作在GPU上布局。

为了简化程序的编写，TensorFlow框架提供了Supervisor机制以管理训练过程。Supervisor负责产生驱动队列的线程，保存模型参数，并TensorBoard数据。创建Supervisor后，程序根据当前训练步数决定运行训练数据或验证数据。

2.2.3 模型评价

为了评价模型，在数据的预处理阶段就需要根据目标词对生成用于评价的替换句。为了编程方便，评价数据的生成与训练数据的生成同时进行，这样很多用于训练数据生成的代码和中间数据都可以复用。程序只需要过滤所有的原句，查找目标词语。如果发现目标词语，则用词语对中对应的词语生成替换句。评价数据的数据格式与训练数据大致相同，只需要将原本的正负例标注换成用于生成替换句的词语对ID。由于评价数据不影响模型训练，因此无需进行乱序操作。

将评价数据输入第2.2.2小节中构建的深度学习模型中，就可以得到每个句子的预测值y'，这样，用于TensorFlow图构建的大部分程序可以被复用。根据词语对ID将预测值分组，并且求预测值的平均值，就可以得到每个词语对的相似度评价。对于词典外词汇，由所有句子的预测值平均值代替本词语对的预测值平均值。由于模型性能的评价指标是Spearman等级相关系数，与预测值的绝对值无关。因此无需对输出做任何缩放处理。Spearman等级相关系数的计算由Excel软件完成。

2.3 实验结果

第 3 章 深度学习辅助词嵌入方法

词嵌入是一种将词语映射为实数向量的技术，这种技术是词语相似度计算中的常用技术。因为相比于词语，向量的相似度比较是非常容易的，通过余弦相似度和平方欧氏距离等方法都可以轻而易举的计算向量的相似度。本章将会介绍一种词嵌入的训练方法，并利用这种方法得到的词嵌入结果计算词语相似度。

词嵌入的训练方法分为直接训练和间接训练。直接训练利用语料库中的统计规律进行训练，最终以得到词嵌入结果为目的；而间接训练则将词嵌入的结果作为其他任务的输入，并将词嵌入矩阵作为模型参数与其他任务的模型共同训练。对于直接训练而言，现在已经有多种成熟的技术，例如NNLM[9]、LBL[10]、C&W[11]、CBOW[12]、Skip-gram[12]与GloVe[13]。而间接训练则多用于深度学习这类基于梯度下降的学习方法中，在第2.1.2小节中，本课题就使用了一种间接训练的方式来训练词嵌入。

上述的直接训练方法都是通过词语与其上下文间的关系来进行训练的，这些方法分为两个类型。其中一种（NNLM、LBL、CBOW、Skip-gram）认为，一个词语应该可以被其上下文所预测出来。而一种该方法（C&W）认为，应该存在一个打分函数，使得一个词语与其上下文之间的得分尽可能高。除此之外，这些方法还使用不同的方式来组合上下文词语[14]。

本章介绍的词嵌入方法是一种直接训练的方法。它将会利用深度学习的技术，对LBL的一种改进方法ivLBL[15]进行了改进。

3.1 原理介绍

3.1.1 ivLBL

由于本章介绍的方法是基于ivLBL方法的改进，所以在这里首先介绍ivLBL方法[1]。

沿用公式2-1中对于C的定义，设有一个词语$w \in \mathcal{V}$，我们很容易从语料库中找到包含这个词语的一个句子s，该句子满足$s \in C, s_i = w$。目标词语附近一定范围内

[1]实际上，为了更好的介绍本章的模型，本小节的内容与原文[15]介绍的ivLBL模型有一定偏差。原文使用了上下文词语间的独立假设，对本小节介绍的部分又进行了更加深入的改进。这一部分将不在本文的讨论范围之内。

的词语（除自身外）称为称为这个词语的上下文。即：

$$c = (s_l, \ldots, s_{i-1}, s_{i+1}, \ldots, s_r), 1 \le l \le i \le r \le |s|, l \ne r \tag{3-1}$$

与其他基于上下文预测的词嵌入方法类似，ivLBL方法使用两个词嵌入矩阵E与E'，分别用于目标词与上下文中的词语的嵌入。这样，目标词w的嵌入表示为E_w，而上下文中的一个词语h_i的嵌入表示为E'_{h_i}。

接下来需要根据整个上下文得到一个预测表示。预测表示是一个与E_w维度相同的向量。为了完成这个任务，需要一个表示函数$\hat{q}$。在ivLBL中，其定义如下：

$$\hat{q}(c) = \frac{1}{n} \sum_{i=1}^{|c|} E'_{h_i} \tag{3-2}$$

得到向量表示以后，我们需要一个打分函数s_θ以评价预测向量与实际向量的相似程度。这个函数可以实现为一个简单的线性变换：

$$s_\theta(w, c) = \hat{q}(c)^\intercal E_w + b_w \tag{3-3}$$

利用这样的打分函数，就可以进行目标词语的预测。

3.1.2 上下文表示

上文中介绍了与上下文c有关的概念，以及上下文表示函数$\hat{q}$。而本课题使用的方法与ivLBL方法的不同之处就在于这两者。现有的ivLBL方法选择固定并且较短（论文中使用的是10，目标词语前后各5个词语）的上下文长度，这样的长度并不足以挖掘句子中的全部上下文信息。并且ivLBL使用的简单平均方式还会丢失词语的顺序信息。

为了完整照顾整个句子中的所有上下文信息，我们设上下文的范围为整个句子。这样，对于句子s中的每一个词语s_i，我们都可以得到一个与之对应的上下文$c = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_{|s|})$。

令$e'_i = E'_{s_i}$，假设我们将e'_i组成的序列输入LSTM模型中，这样就可以得到一个输出序列h_i。由第2.1.3小节的介绍可以得知，输出项h_{i-1}可以携带输入项$e'_1, \ldots, e'_{i-1}$的全部信息，这覆盖了上下文c的前半部分。

但是这还不够，因为我们无法覆盖剩余的部分。为了解决这个问题，需要使用双向LSTM。

3.1.3 双向LSTM

双向LSTM[16]是LSTM模型的一个变种，它分为前向连接和后向连接，其具有如图3-1所示的结构。其前向连接与第2.1.3小节中介绍的普通LSTM完全一致，而后向连接则将中间状态的传递方向改为了从|*s*|到1。

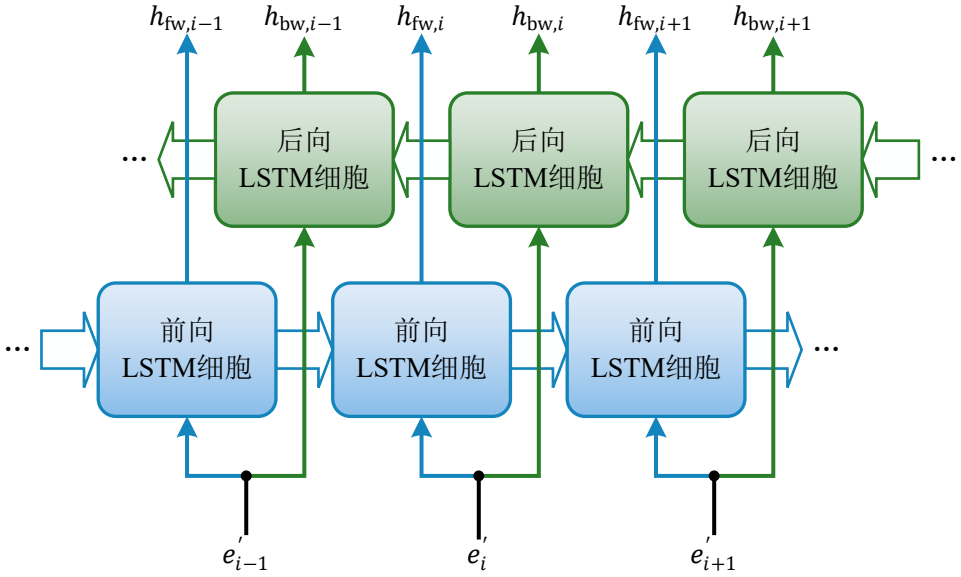

图 3-1 双向LSTM结构

双向LSTM中的前向LSTM细胞和后向LSTM细胞是相同形状的LSTM细胞，但它们之间的参数值互相独立，不能共享。h_{fw}是前向连接的输出，而h_{bw}是后向连接的输出。双向LSTM的完整定义如公式3-4–3-13所述。

$$f_{\mathrm{fw},i} = \sigma(W_{\mathrm{f,fw}}e_{\mathrm{fw},i} + U_{\mathrm{f,fw}}h_{\mathrm{fw},i-1} + b_{\mathrm{f,fw}}) \tag{3-4}$$

$$f_{\mathrm{bw},i} = \sigma(W_{\mathrm{f,bw}}e_{\mathrm{bw},i} + U_{\mathrm{f,bw}}h_{\mathrm{bw},i+1} + b_{\mathrm{f,bw}}) \tag{3-5}$$

$$i_{\mathrm{fw},i} = \sigma(W_{\mathrm{i,fw}}e_{\mathrm{fw},i} + U_{\mathrm{i,fw}}h_{\mathrm{fw},i-1} + b_{\mathrm{i,fw}}) \tag{3-6}$$

$$i_{\mathrm{bw},i} = \sigma(W_{\mathrm{i,bw}}e_{\mathrm{bw},i} + U_{\mathrm{i,bw}}h_{\mathrm{bw},i+1} + b_{\mathrm{i,bw}}) \tag{3-7}$$

$$o_{\mathrm{fw},i} = \sigma(W_{\mathrm{o,fw}}e_{\mathrm{fw},i} + U_{\mathrm{o,fw}}h_{\mathrm{fw},i-1} + b_{\mathrm{o,fw}}) \tag{3-8}$$

$$o_{\mathrm{bw},i} = \sigma(W_{\mathrm{o,bw}}e_{\mathrm{bw},i} + U_{\mathrm{o,bw}}h_{\mathrm{bw},i+1} + b_{\mathrm{o,bw}}) \tag{3-9}$$

$$c_{\mathrm{fw},i} = f_{\mathrm{fw},i} \circ c_{\mathrm{fw},i-1} + i_{\mathrm{fw},i} \circ \tanh(W_{\mathrm{c,fw}}e_{\mathrm{fw},i} + U_{\mathrm{c,fw}}h_{\mathrm{fw},i-1} + b_{\mathrm{c,fw}}) \tag{3-10}$$

$$c_{\mathrm{bw},i} = f_{\mathrm{bw},i} \circ c_{\mathrm{bw},i+1} + i_{\mathrm{bw},i} \circ \tanh(W_{\mathrm{c,bw}}e_{\mathrm{bw},i} + U_{\mathrm{c,bw}}h_{\mathrm{bw},i+1} + b_{\mathrm{c,bw}}) \tag{3-11}$$

$$h_{\mathrm{fw},i} = o_{\mathrm{fw},i} \circ \tanh(c_{\mathrm{fw},i}) \tag{3-12}$$

$$h_{\mathrm{bw},i} = o_{\mathrm{bw},i} \circ \tanh(c_{\mathrm{bw},i}) \tag{3-13}$$

这样，对于词语s_i，$h_{\text{fw},i-1}$与$h_{\text{bw},i+1}$便分别包含了其上下文c前半部分和后半部分的信息。为了得到向量形式的上下文表示$\hat{q}(c)$，还需要将两个向量拼接在一起，并输入一个全连接层：

$$\hat{q}(c) = \tanh(W_{\text{full}}[h_{\text{fw},i-1}, h_{\text{bw},i+1}] + b_{\text{full}}) \tag{3-14}$$

3.1.4 噪声对比估计

第3.1.1小节提到，利用公式3-3中给出的打分函数，就可以进行目标词语的预测。设$p(w \mid c)$是在上下文c中出现词语w的实际概率，一种非常容易想到的想法是，使用柔性最大值函数对打分函数$s_\theta(w,c)$在整个词汇集上进行规范化，就可以得到对$p(w \mid c)$的一个估计，如公式3-15所述。

$$p(w \mid c, \theta) = \frac{e^{s_\theta(w,c)}}{\sum_{v \in \mathcal{V}} e^{s_\theta(v,c)}} \tag{3-15}$$

然而，这样的方法在实际训练中是不现实的。这种方法意味着对于每一个词语w，我们都必须对打分函数进行$|\mathcal{V}|$次求值[2]，这样的时间复杂度是完全无法接受的。

ivLBL使用了噪声对比估计[16]的方法来解决这个问题，这种方法建立了一个辅助的二分类问题。设w'是从某种分布中采样得到的随机词语，这种词语被称为“噪声”，则可以生成二分类问题的输入样本及其标记(v, y)：

$$y = \begin{cases} 1 & v = w, \\ 0 & v = w' \end{cases} \tag{3-16}$$

在数据生成的过程中，噪声词语的数量是目标词语的k倍。这样对于一个词语v，其属于目标词语的概率为：

$$p(D = 1 \mid v) = \frac{p(v \mid h)}{p(v \mid h) + k p_{\text{noise}}(v)} \tag{3-17}$$

其中，$p_{\text{noise}}(v)$是噪声分布中词语v的概率。

对公式3-3中给出的打分函数应用sigmoid函数，就可以得到对$p(v \mid h)$的一个估计：

$$p(v \mid c, \theta) = \sigma(s_\theta(v, c)) \tag{3-18}$$

使用$p(v \mid c, \theta)$代替$p(v \mid h)$，可以得到对$p(D = 1 \mid v)$的估计：

[2]在本课题的实际训练中，词汇量达到了5×10^5个。

$$p(D=1 \mid v,\theta)=\frac{p(v \mid c,\theta)}{p(v \mid c,\theta)+kp_{\text{noise}}(v)}=\sigma\Big(\log(p(v \mid c,\theta))-\log(kp_{\text{noise}}(v))\Big) \quad (3\text{-}19)$$

由于噪声对比估计模型工作在未规范化的数据集上，因此可以简单用$s_\theta(v,c)$代替$\log(p(v \mid c,\theta))$，于是得到[3]：

$$p(D=1 \mid v,\theta)=\sigma\Big(s_\theta(v,c)-\log(kp_{\text{noise}}(v))\Big) \quad (3\text{-}20)$$

应用交叉熵损失函数，可以得到：

$$L_{w,\theta}=\log p(D=1 \mid w,\theta)+\sum_{i=1}^{k}\log(1-p(D=1 \mid w_i',\theta)) \quad (3\text{-}21)$$

3.1.5 模型训练

训练模型使用的优化算法与第2.1.6小节中介绍的完全相同，为Adam算法，这里不再赘述。

3.2 程序实现

由于两种方法在技术上的相似性（均以深度学习为主），深度学习辅助词嵌入方法与深度学习的二分类方法使用了相同的程序整体结构。因此，本节的整体结构部分可以完全参照第2.2节的描述。

3.2.1 数据预处理

深度学习辅助词嵌入方法的数据预处理程序结构如图3-2所示。本次选用的语料库仍然是THUCNews数据集。实际上，“分局分词”、“词典构造”与“查询编号”的部分与第2.2.1小节是完全相同的，其代码也是完全共享。这大大减少了编程的复杂程度。

与之前不同的是，由于词嵌入训练直接使用语料库中的句子，因此本小节的数据预处理程序没有采样句生成的部分。词语的编号序列将直接划分为验证集和训练集，并进行乱序。划分和乱序的算法也与第2.2.1小节相同。

由于词语相似度评价需要计算目标词语的词嵌入间的相似度，因此也需要查询目标词语的编号。目标词语编号查询的结果将会保存在JSON文件中。

[3]由于项$\log(kp_{\text{noise}}(v))$是与$v$有关的常量，其和$s_\theta(v,c)$中可训练的偏置项$b_v$是直接相加关系。因此本课题简化模型，将二者合并为一个偏置项。

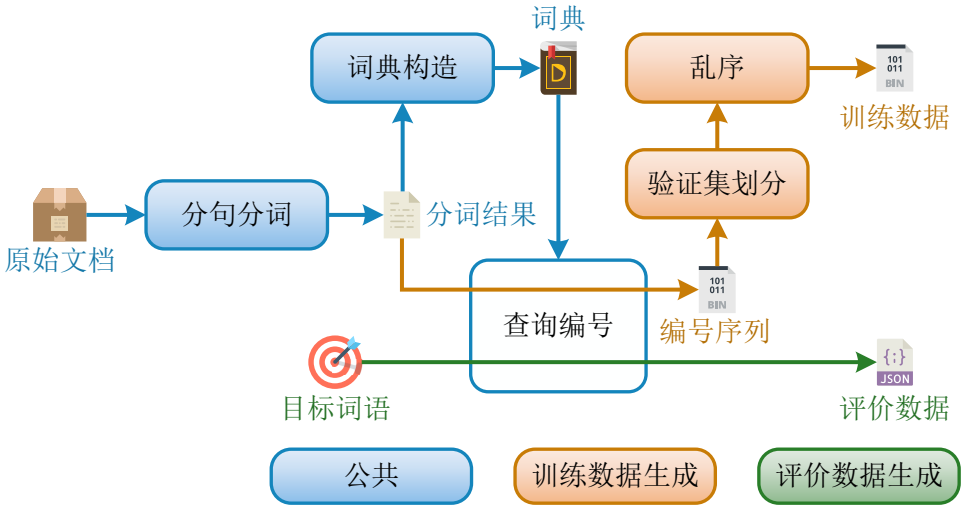

图 3-2 数据预处理

3.2.2 模型训练

模型训练过程除所构建的TensorFlow图与第2.2.2小节不同外，其余内容全部相同，本小节不再赘述。

3.2.3 模型评价

模型评价的程序也是使用Python语言的TensorFlow平台编写的。但是由于输出词语相似度结果不需要训练阶段构建的神经网络，而仅需要从词嵌入矩阵中提取向量，模型评价部分无法复用训练阶段构建的TensorFlow图。

评价程序首先会从训练阶段得到的最优存档点中恢复嵌入矩阵，然后从数据预处理阶段保存的JSON文件中读取目标词语的编号。编号查询和相似度计算的过程被构建在一个新的TensorFlow图中。程序将每一对词语的编号输入TensorFlow图，从词嵌入矩阵中查询对应的向量，并使用预先定义好的函数计算相似度，最终输出结果。对于包含未登录词的词语对，将会取其它词语相似度的平均值。

为了完善实验的结果，本课题分别尝试使用目标词嵌入矩阵E、上下文词嵌入矩阵E'与二者之和$E+E'$产生词嵌入，而相似度评价函数则尝试了余弦相似度与平方欧式距离，他们的定义如下：

$$\mathrm{sim}_{\mathrm{c}}(A,B)=\frac{A\cdot B}{|A||B|} \tag{3-22}$$

$$\mathrm{sim}_{\mathrm{e}}(A,B)=|A-B|^2 \tag{3-23}$$

3.3 实验结果

结　论

学位论文的结论作为论文正文的最后一章单独排写，但不加章标题序号。

结论应是作者在学位论文研究过程中所取得的创新性成果的概要总结，不能与摘要混为一谈。博士学位论文结论应包括论文的主要结果、创新点、展望三部分，在结论中应概括论文的核心观点，明确、客观地指出本研究内容的创新性成果（含新见解、新观点、方法创新、技术创新、理论创新），并指出今后进一步在本研究方向进行研究工作的展望与设想。对所取得的创新性成果应注意从定性和定量两方面给出科学、准确的评价，分（1）、（2）、（3）…条列出，宜用“提出了”、“建立了”等词叙述。

参考文献

[1] Wu Y, Li W. Overview of the NLPCC-ICCPOL 2016 Shared Task: Chinese Word Similarity Measurement[M/OL] // Lin C-Y, Xue N, Zhao D, et al. Natural Language Understanding and Intelligent Applications: 5th CCF Conference on Natural Language Processing and Chinese Computing, NLPCC 2016, and 24th International Conference on Computer Processing of Oriental Languages, ICCPOL 2016, Kunming, China, December 2–6, 2016, Proceedings. Cham : Springer International Publishing, 2016 : 828 – 839.
http://dx.doi.org/10.1007/978-3-319-50496-4_75.

[2] Hochreiter S, Schmidhuber J. Long Short-Term Memory[J/OL]. Neural Computation, 1997, 9(8) : 1735 – 1780.
http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[3] Gers F A, Schmidhuber J, Cummins F. Learning to Forget: Continual Prediction with LSTM[J/OL]. Neural Computation, 2000, 12(10) : 2451 – 2471.
http://dx.doi.org/10.1162/089976600300015015.

[4] Sak H, Senior A W, Beaufays F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling[C/OL] // Li H, Meng H M, Ma B, et al. INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014. [S.l.] : ISCA, 2014 : 338 – 342.
http://www.isca-speech.org/archive/interspeech_2014/i14_0338.html.

[5] Duchi J, Hazan E, Singer Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization[J/OL]. J. Mach. Learn. Res., 2011, 12 : 2121 – 2159.
http://dl.acm.org/citation.cfm?id=1953048.2021068.

[6] Zeiler M D. ADADELTA: An Adaptive Learning Rate Method[J/OL]. CoRR, 2012, abs/1212.5701.
http://arxiv.org/abs/1212.5701.

[7] Kingma D P, Ba J. Adam: A Method for Stochastic Optimization[J/OL]. CoRR, 2014, abs/1412.6980.
http://arxiv.org/abs/1412.6980.

[8] Ruder S. An overview of gradient descent optimization algorithms[J/OL]. CoRR, 2016, abs/1609.04747.
http://arxiv.org/abs/1609.04747.

[9] Bengio Y, Ducharme R, Vincent P, et al. A Neural Probabilistic Language Model[J/OL]. J. Mach. Learn. Res., 2003, 3 : 1137 – 1155.
http://dl.acm.org/citation.cfm?id=944919.944966.

[10] Mnih A, Hinton G. Three New Graphical Models for Statistical Language Modelling[C/OL] // ICML '07 : Proceedings of the 24th International Conference on Machine Learning. New York, NY, USA : ACM, 2007 : 641 – 648.
http://doi.acm.org/10.1145/1273496.1273577.

[11] Collobert R, Weston J. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning[C/OL] // ICML '08 : Proceedings of the 25th International Conference on Machine Learning. New York, NY, USA : ACM, 2008 : 160 – 167.
http://doi.acm.org/10.1145/1390156.1390177.

[12] Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space[J/OL]. CoRR, 2013, abs/1301.3781.
http://arxiv.org/abs/1301.3781.

[13] Pennington J, Socher R, Manning C D. Glove: Global Vectors for Word Representation[C/OL] // Moschitti A, Pang B, Daelemans W. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL. [S.l.] : ACL, 2014 : 1532 – 1543.
http://aclweb.org/anthology/D/D14/D14-1162.pdf.

[14] Lai S, Liu K, He S, et al. How to Generate a Good Word Embedding[J/OL]. IEEE Intelligent Systems, 2016, 31(6) : 5 – 14.
http://dx.doi.org/10.1109/MIS.2016.45.

[15] Mnih A, Kavukcuoglu K. Learning word embeddings efficiently with noise-contrastive estimation[G/OL] // Burges C J C, Bottou L, Welling M, et al. Advances in Neural Information Processing Systems 26. [S.l.] : Curran Associates, Inc., 2013 : 2265 – 2273.
http://papers.nips.cc/paper/5165-learning-word-embeddings-efficiently-with-noise-contrastive-estimation.pdf.

[16] Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures[J/OL]. Neural Networks, 2005, 18(5–6) : 602 – 610.
http://www.sciencedirect.com/science/article/pii/S0893608005001206.

哈尔滨工业大学本科毕业设计（论文）原创性声明

本人郑重声明：在哈尔滨工业大学攻读学士学位期间，所提交的毕业设计（论文）《中文词语相似度计算》，是本人在导师指导下独立进行研究工作所取得的成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明，其它未注明部分不包含他人已发表或撰写过的研究成果，不存在购买、由他人代写、剽窃和伪造数据等作假行为。

本人愿为此声明承担法律责任。

作者签名：　　日期：　　年　　月　　日

致　谢

（大概就是这些？）

附录 A 据说还要写文献翻译好气哦!