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 **6** 月 **23** 日

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 6 月 23 日

摘　要

中文词语相似度计算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。本课题取自 NLPCC-ICCPOL 2016 会议中一个开放任务。本文使用了深度学习的二分类方法、深度学习辅助词嵌入方法与基于词典或知识库的方法计算词语相似度，并使用集成学习方法结合其他方法的结果。

深度学习的二分类方法使用了生成替换句的手段，将无监督回归问题转化为有监督分类问题，并且使用深层 LSTM 的技术解决了分类问题。深度学习辅助词嵌入方法利用词嵌入计算词语相似度，并改进了现有的 ivLBL 方法，利用双向 LSTM 计算上下文向量。基于词典或知识库的方法从同义词词林扩展版中提取有用的特征用于计算。集成学习方法使用了 stacking 方法，并尝试了线性回归、岭回归与 LASSO 三种回归模型。

词语相似度计算方法的性能使用系统输出与人工标注间的 Spearman 等级相关系数来衡量。本文所介绍的方法可以得到 0.4855 的成绩。在不依赖 PKU-500 数据集的情况下，成绩也可以达到 0.3667。

关键词：词语相似度；深度学习；LSTM；词嵌入；同义词词林；集成学习

Abstract

Chinese word similarity measurement is a task aiming to provide a quantized measurement of the similarity of a pair of Chinese words. This project came from a shared task from the conference NLPCC-ICCPOL 2016. This thesis utilized deep learning classification method, deep learning aided word embedding method and dictionary / knowledge base based method to calculate word similarity. Then, ensemble learning method is used to combine results generated from other methods.

Deep learning classification method uses a means of generating word-replaced sentences to transform the unsupervised regression problem into a supervised classification problem. After that, deep LSTMs are used to solve the classification problem. Deep learning aided word embedding method makes use of word embedding to calculate word similarity. It improves the existing ivLBL method, calculating context vectors with bidirectional LSTMs. Dictionary / knowledge base based method extracts useful features from Tongyici Cilin (Extended). Ensemble learning uses stacking method, makes attempts with linear regression, ridge regression and LASSO.

The performance of word similarity measurement methods is measured by the Spearman's rank correlation coefficient between the system outputs and human annotations. The method introduced by this thesis achieved a performance of 0.4855. Even if not to depend on the PKU-500 dataset, the performance can reach 0.3667.

Keywords: word similarity, deep learning, LSTM, word embedding, Tongyici Cilin, ensemble learning

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。

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[1]，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[2]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[3]，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[2]。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[4]。

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：

1. 领域：同时涉及传统书面语言与近期的网络语言。
2. 频率：应同时包含高频、中频与低频的词语。
3. 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）。
4. 字数：应包含 1 – 4 个字的词语。
5. 词义：应包含部分多义词。
6. 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

本课题的主要研究四种方法。一是深度学习的二分类方法，二是深度学习辅助词嵌入方法，三是基于词典或知识库的方法，四是集成学习方法。上述方法或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 深度学习的二分类方法

将困难问题规约到简单问题是科学研究中常用的手法，本方法将词语相似度计算这一无监督回归问题转化为有监督分类问题，以降低解决问题的难度。对于句子的分类问题，本方法使用循环神经网络以及它们的变种 LSTM 予以解决。

1.4.2 深度学习辅助词嵌入方法

词嵌入是自然语言处理中的一项非常成熟的技术，它可以将词语映射到向量空间中。而向量具有天然的相似度评价方法，这使得词嵌入技术经常被用于解决词语相似度计算问题。本方法查找现有词嵌入技术中的局限性，并使用深度学习方法拓展现有词嵌入技术，以改善它们的性能。

1.4.3 基于词典或知识库的方法

很多现有的语言学资料中提供了对词语语义的一些形式化的解释。这些解释可以被计算机解析和利用，因此也可以作为词语相似度计算的素材。本方法尝试使用现有的词典或知识库资源，并对业界常用的利用它们进行词语相似度计算的方法进行一些改进。

1.4.4 集成学习方法

实际的词语相似度计算系统多是多种方法的结合。这有助于发挥各方法的长处，同时提高系统的稳定性。在其它开放任务参赛队发表的系统中，集成多种技术最常见的方法是简单平均法和加权平均法。除此之外，很多人为制定的基于规则的结合策略也被用于此用途。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。尽管如此，发现一个好的结合规则仍然需要大量实验和一定的运气成分。因此，需要借助集成学习的技术来发现一种更可靠的、表现更为稳定的方法。

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_a 到 v_b 和从 v_b 到 v_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。

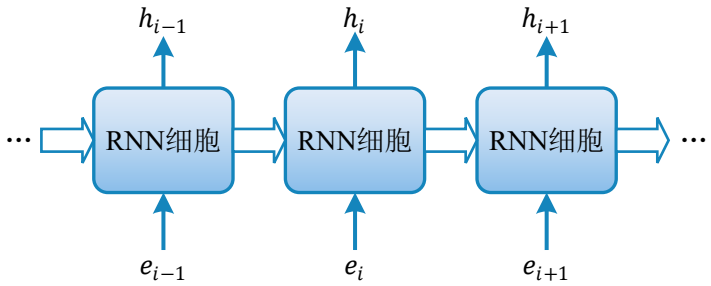


图 2-1 RNN 结构

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 [5] 中提出的 LSTM （Long short-term memory）。其经过了 [6] 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事件。这得益于 LSTM 中加入的遗忘门、输入门和输出门的设计。

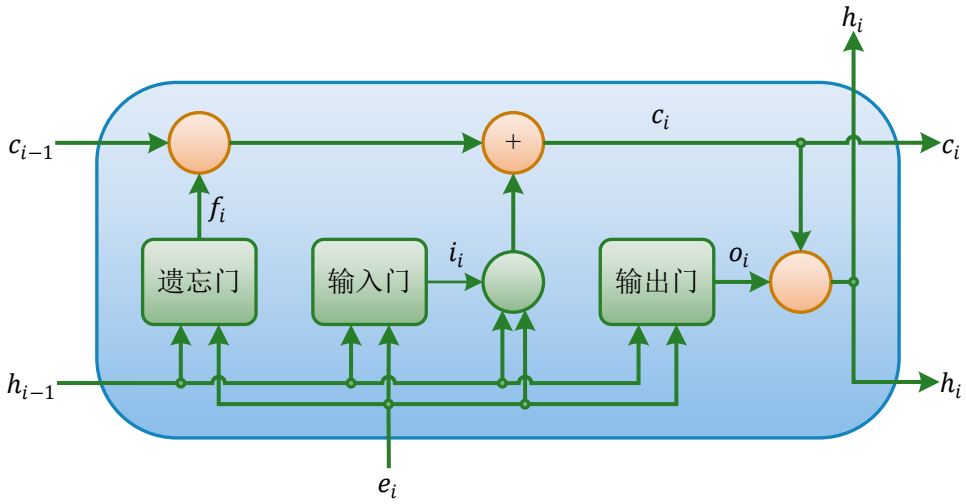


图 2-2 LSTM 细胞结构

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。

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

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

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

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

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

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 [7] 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。

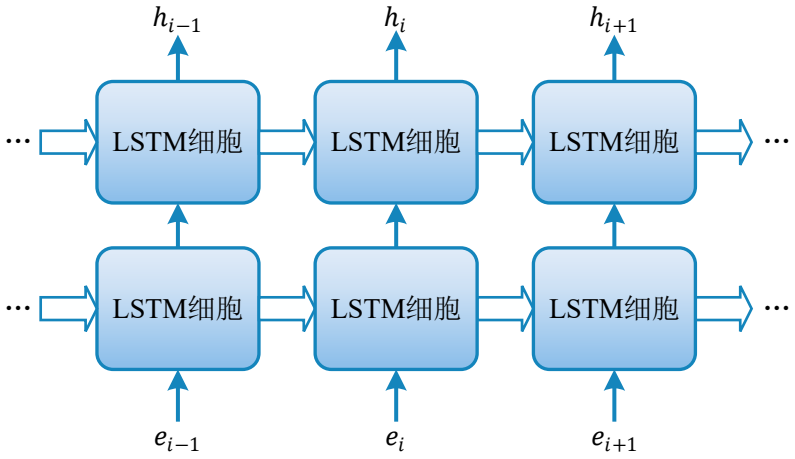


图 2-3 深层 LSTM

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准确的模型。

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[8]、Adadelta[9]、RMSprop[1]与 Adam[10]。为了降低实验的复杂程度，本课题使用了 [11] 中推荐的 Adam 算法进行

[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。

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。

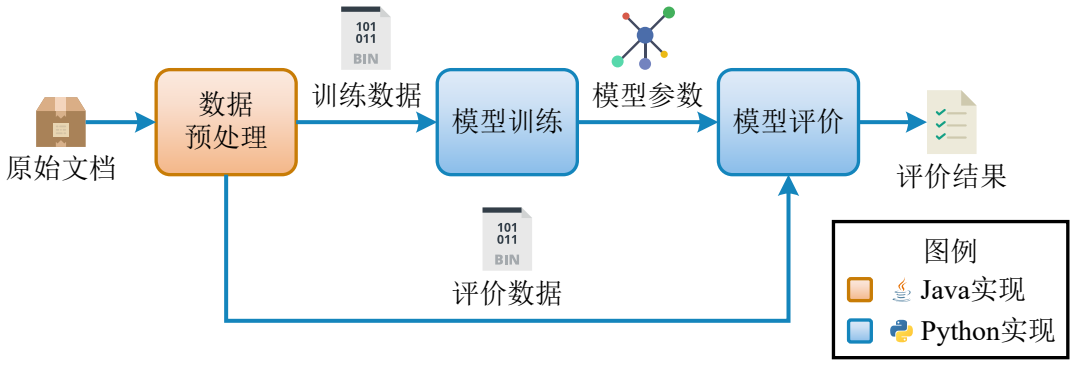


图 2-4 整体结构

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。

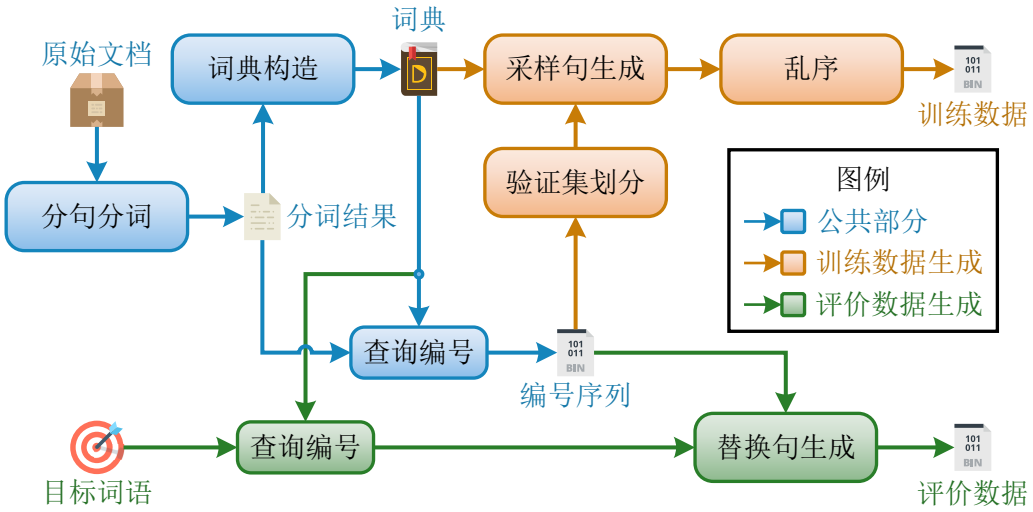


图 2-5 数据预处理

本课题使用了清华大学 THUCTC 工具包中提供的 THUCNews 数据集[12]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。

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，也可以使用基本操作对数据进行运算和调整。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。

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

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。每个队列需要一个单独的线程进行驱动。

使用 TensorFlow 框架开发程序的第一个步骤是数据读取。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 节中所列出的公式，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，并生成 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语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，由其他所有词语对的相似度评价的平均值代替本词语对的相似度评价。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使用 Python 语言的 SciPy 库完成。

2.3 实验结果

本章所使用的实验环境如表 2-1 所示。深度学习实验共进行了两次，分别采用不同的参数。其结果如表 2-2 所示。

表 2-1 环境配置

(a) 硬件环境

项目	配置
CPU	Intel Xeon E3-1230
内存	16GB
显卡	NVIDIA Tesla K40c
硬盘	240GB SSD

(b) 软件环境

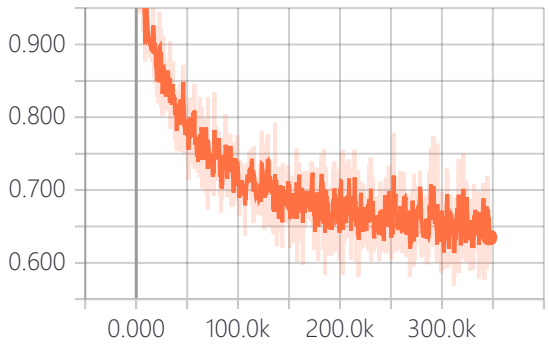
项目	配置
操作系统	Ubuntu 16.04
CUDA 版本	8.0
cuDNN 版本	5.1
TensorFlow 版本	1.1.0
Python 版本	Python 3.6 (Miniconda 3)

表 2-2 实验结果

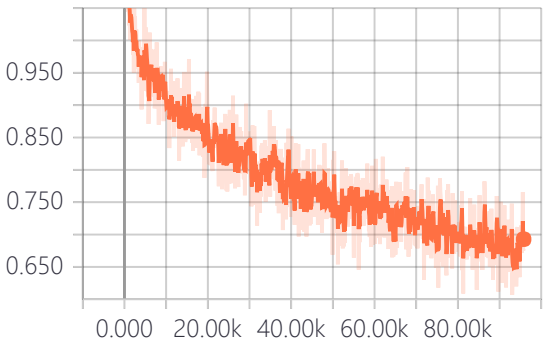
词嵌入维数	LSTM 单元数	LSTM 层数	全连接层数	损失函数值	Spearman 等级相关系数
300	128	2	0	0.5422	0.1313
300	300	3	0	0.5972	0.1484
300	128	2	1	0.5448	0.1579

从实验结果中可以看出，适当增加 LSTM 单元数和层数，可以提高模型的最终性能，但这种性能提高少于一个全连接层所能带来的。鉴于训练全连接层的代价要小于 LSTM 层，因此后者是优化模型的有效方式。然而，三次训练得到的模型成绩都不是很高，这意味着深度学习的二分类模型并不适合单独用于解决词语相似度计算问题。尽管如此，这种方法输出的信息对于本课题后期的集成学习仍然是有正面意义的。

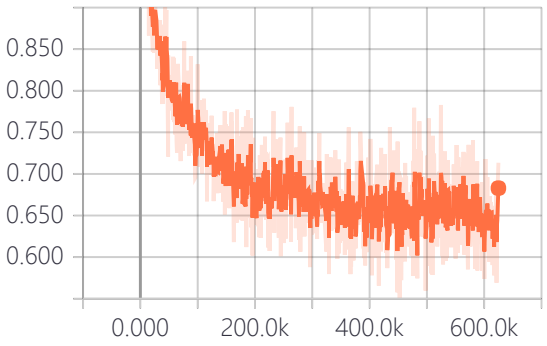
三次训练的损失函数下降情况如图 2-6 所示，其中横坐标是训练的轮数，而纵坐标是损失函数值。可以看出，三次训练中损失函数值均已收敛。



a) 第一次训练



b) 第二次训练



c) 第三次训练

图 2-6 损失函数下降情况

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[13]、LBL[14]、C&W[15]、CBOW[16]、Skip-gram[16]与 GloVe[17]。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。而另一种方法（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[18]。

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[19]进行改进。

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

[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[19]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。

范围内的词语（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[20]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。

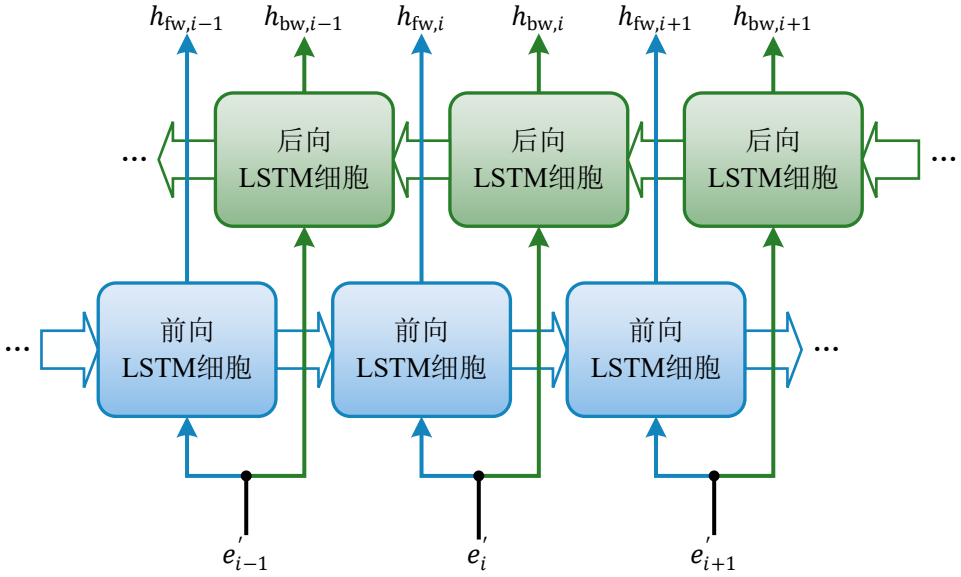


图 3-1 双向 LSTM 结构

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[20]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。

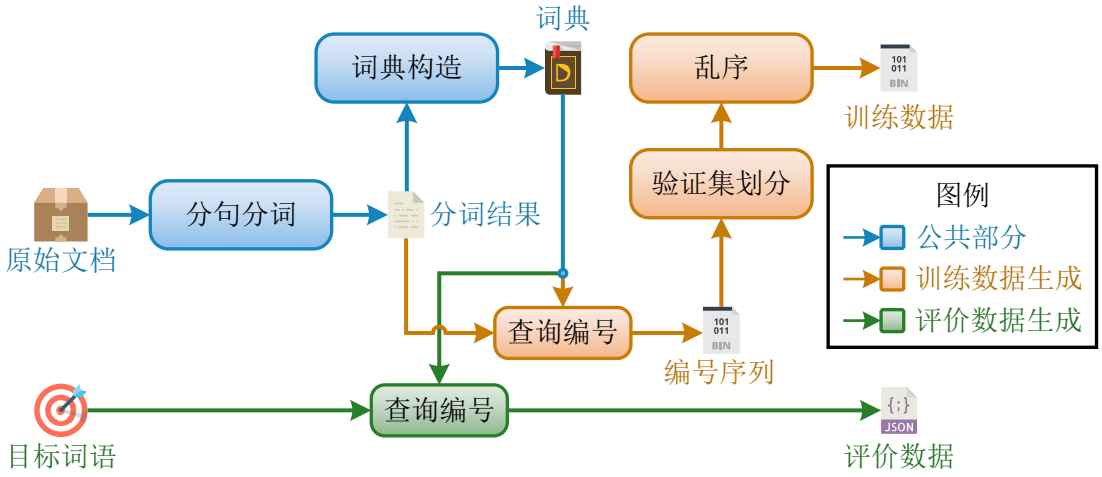


图 3-2 数据预处理

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，将会取其它词语对相似度的平均值。

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：

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

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

3.3 实验结果

本章所使用的实验环境与表 2-1 相同，词嵌入的训练进行了一次，其超参数设置与训练情况如表 3-1 所示，损失函数的下降情况如图 3-3，而结果如表 3-2 所示。

表 3-1 训练情况

词嵌入维数	LSTM 单元数	损失函数值
300	128	14081.98

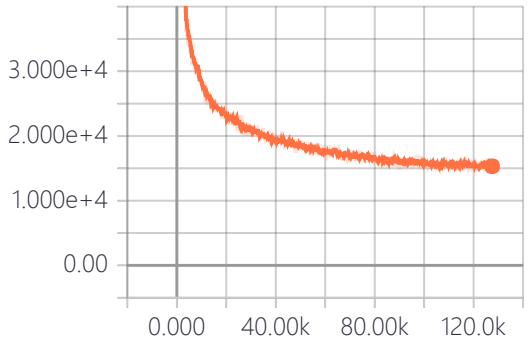


图 3-3 损失函数下降情况

表 3-2 实验结果

相似度函数	词嵌入矩阵	Spearman 等级相关系数
$\mathrm{sim_c}$	E	0.3860
$\mathrm{sim_c}$	E'	0.3542
$\mathrm{sim_c}$	$E + E'$	0.3756
$\mathrm{sim_e}$	E	0.2017
$\mathrm{sim_e}$	E'	0.0304
$\mathrm{sim_e}$	$E + E'$	0.0943

通过损失函数的下降情况，可以确认损失函数已经收敛。实验结果表明，使用余弦相似度评价目标词嵌入矩阵可以取得最好的性能。特别地，平方欧氏距离并不适合用于衡量词嵌入向量间的相似度，其效果与余弦相似度差距巨大。本方法单独用于解决中文词语相似度问题已经能够得到非常好的成绩，甚至能超过参加开放任务的 2/3 的系统。

第 4 章 基于词典或知识库的方法与集成学习方法

词典或知识库是一种描述词语语义的语言学资料。这里的词典并不像我们常用的《现代汉语词典》那样使用自然语言解释词语，它会将词语组织成一定的结构，并且使用形式化的方式进行描述。这就为词语的语义提供了一种计算机容易理解的表示形式。

在英文方面，最著名的词典或知识库是普林斯顿大学开发的 WordNet[21]。而在中文方面，则有董振东教授开发的知网（HowNet）与梅家驹等人开发的同义词词林。后者经过了哈尔滨工业大学信息检索实验室的改进，形成了同义词词林扩展版。

集成学习是一种将多个个体学习器进行组合，而获得更高性能的学习器的机器学习方法。这种方法通常可以得到好于任何一个个体学习器的性能。

主流的集成学习方法分为三种，分别被称为 boosting、bagging 和 stacking。其中，boosting 使用迭代方法训练一系列学习器，每次迭代都根据上一个学习器的输出改变样本的权重，使得上一个学习器分类错误的样本获得更高权重；Bagging 使用一种有放回的“自助采样”方法生成多个样本集，用每个样本集训练一个基学习器，并使用投票方式将这些学习器的输出进行组合；Stacking 则使用一个额外的学习器来组合基学习器的输出。本课题中使用 stacking 方法进行集成学习。

本章将会给出一种基于同义词词林扩展版的词语相似度计算方法，并使用集成学习的方法来结合本文中先前介绍过的各种方法。由于二者结合非常紧密，故合并在一章内介绍。

4.1 原理介绍

4.1.1 同义词词林扩展版

最初版本的同义词词林包含 53859 个词条，随着大家对中文使用习惯的变化，有 14706 个词条的使用频率降低，成为了罕用词与非常用词。同义词词林扩展版删除了这些词语，并且增补完善了一定数量的词语，使得最终的词条数量达到了 77343 条。

在结构上，同义词词林扩展版将词条分类组织为一个树状结构，如图 4-1 所示。分类的层次共有 5 层，随着层次的增加，词语语义的划分越来越细。第五层

的词类称为原子词群。一个原子词群中的词语可能是同义词，可能是互相相关的非同义词，也可能一个原子词群中仅有一个词语。同一个词语可能同时出现在多个不同的原子词群中，这意味着这个词语是一个多义词。原子词群的编码方式如图 4-2 所示，其中，“=”表示一个原子词群是同义词，而“#”表示一个原子词群是相关的非同义词，“@”表示原子词群内只有一个词语。

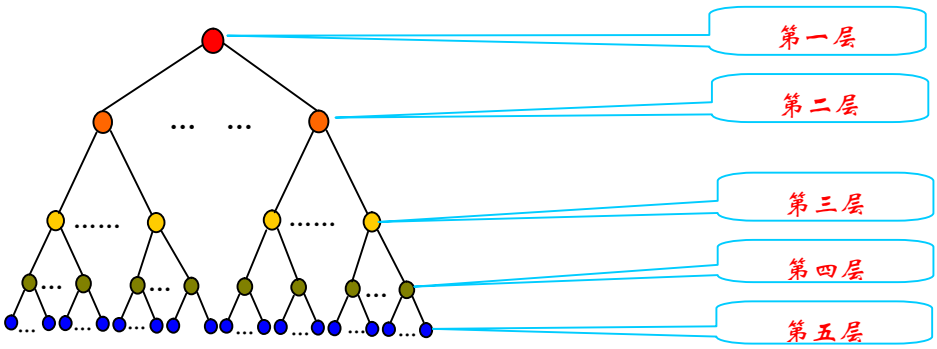


图 4-1 同义词词林扩展版的结构（来自同义词词林扩展版说明）

编码位	1	2	3	4	5	6	7	8
符号举例	D	a	1	5	B	0	2	= \ # \ @
符号性质	大类	中类	小类		词群	原子词群		
级别	第 1 级	第 2 级	第 3 级		第 4 级	第 5 级		

图 4-2 原子词群的编码方式（来自同义词词林扩展版说明）

4.1.2 Stacking 方法

在上文中提到的三种集成学习方法中，stacking 方法是唯一一种能够集成异构的学习器的方法。由于本课题需要集成来自不同机器学习方法的结果，因此必须使用这种方法。Stacking 方法用一个学习器来组合其它学习器的输出。其中，被组合的学习器称为初级学习器 M_i，而用于组合的学习器称为次级学习器 M_e。

设集成学习系统的输入的特征为 x，而标注为 y，则每个初级学习器的训练过程与一般的机器学习过程相同。每一个初级学习器使用同样的特征与标注独立地进行训练，即训练目标为 $M_i(x) \rightarrow y$。训练完成后，每个初级学习器都会给出对标注的一个预测 $M_i(x) = y'_i$。

次级学习器 M_e 则以初级学习器的输出 y'_i 作为输入，其训练目标为 $M_e(y'_1, \ldots, y'_n) \rightarrow y$。这样，次级学习器就起到了组合初级学习器的输出的作用。

在 stacking 方法中，每个初级学习器通常是用不同的方法构造出来的，这意味着初级学习器的数量不会很多（例如本课题中仅有三个，人工构建出大量的初级学习器的工作量是难以估量的）。于是，stacking 方法一般会使用相对较为简单的机器学习方法。本课题中分别尝试使用线性回归、岭回归与 LASSO 三种方法构造次级学习器。

4.1.3 线性回归

线性回归是最为简单的回归方式之一，它使用一个线性函数产生对标注值的预测。假设我们有一个数据集 $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$，其中 $x_i \in \mathbb{R}^m, y_i \in \mathbb{R}$。线性回归具有如下的形式：

$$y_i' = \beta^\intercal x_i + b \tag{4-1}$$

其中，系数 β 与置偏 b 都是线性回归的系数。

线性回归的最优解很容易通过最小二乘法得到。最小二乘法令预测值与实际值之差的平方和最小，即：

$$(\beta^*, b^*) = \underset{(\beta,b)}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - y_i')^2 \tag{4-2}$$

将公式 (4-1) 中的变量进行一些变形，设：

$$X = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ 1 & 1 & \cdots & 1 \end{pmatrix}^\intercal \tag{4-3}$$

$$Y = (y_1, y_2, \ldots, y_n)^\intercal \tag{4-4}$$

$$\hat{\beta} = \begin{pmatrix} \beta \\ b \end{pmatrix} \tag{4-5}$$

则最小二乘法的目标变为：

$$\hat{\beta}^* = \underset{\hat{\beta}}{\operatorname{argmin}} (Y - X\hat{\beta})^\intercal (Y - X\hat{\beta}) \tag{4-6}$$

通过求导的方式，可以得到：

$$\hat{\beta}^* = (X^\intercal X)^{-1} X^\intercal Y \tag{4-7}$$

即是线性回归的最优解。

4.1.4 岭回归与 LASSO

在样本特征较多，而样本数量较少时，线性回归容易出现过拟合的问题。解

决这个问题的方式是对模型参数添加规范化项。

如果规范化项为 L2 规范化，则称为岭回归：

$$\hat{\beta}^* = \underset{\hat{\beta}}{\operatorname{argmin}}((Y - X\hat{\beta})^\intercal(Y - X\hat{\beta}) + \lambda||\hat{\beta}||_2^2) \tag{4-8}$$

如果规范化项为 L1 规范化，则称为 LASSO：

$$\hat{\beta}^* = \underset{\hat{\beta}}{\operatorname{argmin}}((Y - X\hat{\beta})^\intercal(Y - X\hat{\beta}) + \lambda||\hat{\beta}||_1) \tag{4-9}$$

超参数 λ 是规范化项的系数。本课题的实验中将会手动调整这个系数的值，直到取得最好的模型性能。

4.1.5 辅助特征与词典或知识库特征的提取

除了各方法输出的词语相似度计算结果以外，本课题还尝试进行一些额外的特征提取工作，包括深度学习模型运行过程中产生的信息，以及来自词典或知识库的特征。

对于深度学习的二分类方法，考虑公式 (2-5) 的内容。要针对一个词语对计算数学期望，就必须对 M_C 的输出求平均值。本课题除需要求得的平均值之外，还将输出的方差作为辅助特征输入集成学习器。除此之外，一个词语对是否包含词典外词语（这表明了模型使用的究竟是真实输出还是使用平均值代替的输出）也会作为辅助特征被提取出来。

对于深度学习辅助词嵌入方法，虽没有其它的辅助特征可以提取，但词语对是否包含词典外词语的信息依然有效。

对于基于词典或知识库的方法，本课题直接从同义词词林扩展版的结构中提取特征。对于每一个词语对，本方法提取的特征是一个三维向量 (d, s_1, s_2)。我们先设 d' 为同时包含两个词语的最深的类别的深度（也就是最近公共祖先），如果有词语同时属于多个原子词群，则选 d' 值最大的。当两个词语不属于相同的一级类别时，$d' = 0$。对于 $d' < 5$ 的情况，$d = d'$，而当 $d' = 5$（两个词语属于同一个原子词群）时，需要分情况讨论。当原子词群是同义词时，$d = d' + 1$，否则 $d = d'$。s_1 与 s_2 分别是包含每个词语的原子类群数量（每个词语的词义个数）。

值得注意的是，本文中介绍的基于词典或知识库的方法并没有独立产生词语相似度计算的结果，而是将提取到的特征直接输入到集成学习器中。与此相反，传统的方法会利用线性映射或其它公式使用这些特征计算词语相似度结果，而这样的计算方法与集成学习器的功能是类似的。如果在本课题的系统中如此实现，将会使系统产生不必要的冗余。

4.1.6 模型训练

由于集成学习器属于有监督学习，如果要训练集成学习器，就必须找到一些有标注的词语相似度数据集作为训练数据。

一种方法是利用本课题的背景任务——NLPCC-ICCPOL 2016 会议中的“中文词语相似度计算”开放任务的组织者在比赛开始前提供的样本数据。这些样本数据中有 40 个词语对，其格式与 PKU-500 数据集相同。使用初级学习器在这 40 个词语对上的输出，就可以训练次级学习器。

另一种方法是在 PKU-500 数据集上使用交叉验证法。对于数据集 D，交叉验证法将其分为 k 个等大小的互斥子集，即 $D = \bigcup_{i=1}^{k} D_i, \forall (i, j), D_i \cap D_j = \emptyset$。这样，对于每一个子集 D_i，可以用 $D - D_i$ 训练集成学习器，而令集成学习器产生 D_i 的输出。这样，就可以利用 PKU-500 数据集的标注训练模型，而不必担心训练数据与评价数据发生重叠。在极端情况下，令 $k = |D|$，这样每一个子集中只包含一个样本。这种方式称为留一法。k 的值越大，就越能充分利用有标注的数据集，同时需要训练的模型数量也越多。鉴于数据集的规模较小[1]，本课题使用了留一法进行集成学习器的训练。

4.2 程序实现

本方法使用 Python 语言的 scikit-learn 库来完成集成学习。得益于 scikit-learn 库的接口易用性，本方法的程序实现非常的简单。

两种深度学习模型的输出被存储在文本文件中。进行集成学习的 Python 程序需读取它们的内容。而同义词词林扩展版则以自己的文本格式保存在文件中。程序首先解析并读取该文件，接下来读取词语对，并从同义词词林扩展版中查询得到要提取的特征。

上面所有的输出与特征将会转化为浮点型向量，并且拼接在一起。最终形成一个 8 维向量，其内容如表 4-1 所述。

数据集的人工标注以代码形式存储在项目中。使用 scikit-learn 建立线性回归、岭回归与 LASSO 模型，输入上述向量和数据集中的标注即可进行训练。运行模型评价时，将训练数据的特征输入训练好的模型，并利用 SciPy 计算模型输出与标准结果之间的 Spearman 等级相关系数。

[1]即使使用了留一法，本章所述的全部程序依然能够在 1s 的时间内运行完成。

表 4-1 集成学习器的输入向量

维度	模型	内容
1	深度学习的二分类方法	输出均值
2		输出方差
3		是否包含词典外词语
4	深度学习辅助词嵌入方法	输出相似度
5		是否包含词典外词语
6	基于词典与知识库的方法	最近公共祖先深度
7		左词语义项数
8		右词语义项数

4.3 实验结果

本章的实验平台为作者的笔记本电脑，其配置如表 4-2 所示。实验结果如表 4-3 所示。可以发现，使用岭回归进行集成学习的性能最高，但三种回归模型的性能差距并不显著。使用样本数据进行训练的效果并不是十分理想，结果甚至弱于深度学习辅助词嵌入方法的独立结果。这可能是由于训练集过小导致的。而交叉验证的方法由于使用的训练集更大，可以得到十分喜人的结果。

表 4-2 环境配置

(a) 硬件环境

项目	配置
CPU	Intel Core i7-4700MQ
内存	8GB
硬盘	256GB SSD + 1TB HDD

(b) 软件环境

项目	配置
操作系统	Windows 10 专业版
scikit-learn 版本	0.18.1
Python 版本	Python 3.6 (Anaconda 3)

表 4-3 实验结果

训练集划分	集成学习器	最优超参数设置	Spearman 等级相关系数
样本数据	线性回归	无	0.3610
	岭回归	$\alpha = 0.63$	0.3667
	LASSO	$\alpha = 0.001$	0.3649
交叉验证	线性回归	无	0.4849
	岭回归	$\alpha = 0.55$	0.4855
	LASSO	$\alpha = 10^{-4}$	0.4853

结　论

本课题得到了一套系统完整的中文词语相似度计算方法，其性能在已知的方法中处于较高的地位。高性能的词语相似度计算将会为多种下游自然语言处理问题提供可靠的保障。

深度学习的二分类方法创新性地将无监督回归问题转化为了有监督分类问题，使得广泛存在于互联网中的生语料库可以被用于词语相似度计算问题中。深度学习辅助词嵌入方法利用词嵌入技术解决了词语相似度计算问题，并使用双向LSTM改进了已有的ivLBL方法，充分将LSTM的序列学习能力运用到上下文向量的生成中，扩大了上下文的学习范围，使得词嵌入的质量得到显著的提升。基于或知识库的方法与集成学习方法相结合，改变了传统的基于规则组合特征的方式。集成学习方法有效地组合了上述方法的输出，并得到了比每个个体学习器更强的性能。

然而，本方法也有一定的不足之处。其中一个问题在于本方法无法处理词典外词汇，对于他们只能使用平均值等数据来填充。另一个问题在于集成学习的性能严重受制于有标注数据集的大小和质量。随着日后中文词语相似度标注数据集的增多，本方法的性能也会得到更大的提升。

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, 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] Pei J, Zhang C, Huang D, et al. Combining Word Embedding and Semantic Lexicon for Chinese Word Similarity Computation[M/OL] // Lin C, 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 : 766 – 777.
http://dx.doi.org/10.1007/978-3-319-50496-4_69.

[3] Ma S, Zhang X, Zhang C. NLPCC 2016 Shared Task Chinese Words Similarity Measure via Ensemble Learning Based on Multiple Resources[M/OL] // Lin C, 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 : 862 – 869.
http://dx.doi.org/10.1007/978-3-319-50496-4_79.

[4] 张硕望, 欧阳纯萍, 阳小华, 等. 融合《知网》和搜索引擎的词汇语义相似度计算[J/OL]. 计算机应用, 2017, 37(4) : 1056.
http://www.joca.cn/CN/abstract/article_20422.shtml.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

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

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

[12] 孙茂松, 李景阳, 郭志芃, 等. THUCTC：一个高效的中文文本分类工具包[CP/OL]. 2016.
http://thuctc.thunlp.org/.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] Miller G A. WordNet: A Lexical Database for English[J/OL]. Commun. ACM, 1995, 38(11) : 39 – 41.
http://doi.acm.org/10.1145/219717.219748.

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

感谢我的毕业设计指导老师孙大烈老师和刘旭东老师在答辩和论文方面给予我的指导。

感谢我的硕士生导师李舟军老师教授我本课题中使用的关键技术。

感谢傅忠传老师为本课题的深度学习实验提供硬件平台，并感谢张源悍学长在硬件平台配置过程中提供的帮助。

感谢车万翔老师为本课题提供实验数据。

本课题使用了北京大学的 PKU-500 数据集、清华大学的 THUCNews 数据集以及哈尔滨工业大学的同义词词林扩展版，感谢这些数据集的作者做出的贡献。

论文的插图中使用了 Flaticon 网站中由 Madebyoliver、Freepik、Roundicons、Vectors Market 与 Maxim Basinski 提供的图标资源，这些图标以 CC 3.0 BY 协议发布。

最后，感谢我最爱的人郭怡心，在我本科四年的生活中给予我的陪伴、鼓励与支持。