利用 PyTorch 实现基于 Word2Vec_Skip-Gram 的文本相似度计算

梨花先雪

摘要

本实验旨在通过实现 Skip-Gram 模型对中文语料进行词向量训练，并利用余弦相似度评估词语之间的语义相似性，从而加深对词向量嵌入技术的理解。实验首先对中文对话数据进行预处理，包括去除停用词和基于 jieba 的分词操作。随后使用 Skip-Gram 模型训练词向量，并通过余弦相似度计算词语之间的相似程度。实验采用了小规模中文数据集，在优化模型参数的基础上完成了词向量训练。最终的实验结果显示，即便在小样本条件下，模型仍能较好地学习出词语之间的语义关系，验证了 Skip-Gram 模型在词向量表示中的有效性和泛化能力。本实验不仅提升了作者对词嵌入技术的理解，还对未来在交叉学科中的模型迁移与应用提供了有益的思路。

关键词：Skip-Gram；词向量；jieba 分词；余弦相似度；文本预处理

1. 引言

随着自然语言处理技术的发展，词向量（Word Embedding）作为基础表示方法之一，已广泛应用于文本分类、机器翻译、情感分析等领域。传统的词语表示方式如独热编码虽然简单直观，但存在高维稀疏、不表达语义等问题。为解决此类问题，Skip-Gram 模型被提出并广泛应用，其通过上下文预测中心词，能够在低维空间中捕捉词语间的语义关系。

当前主流研究多集中于大规模语料库下的词向量训练，但在小样本环境下模型的表现与泛化能力仍缺乏系统探讨。此外，中文语言的复杂性也为分词、语义建模等任务带来挑战。因此，如何在资源受限的情况下利用有效工具与模型进行中文词向量训练成为一个具有实践意义的问题。

本实验聚焦于使用 Skip-Gram 模型对中文对话语料进行词向量学习，探究其在小规模数据上的表现，并评估训练得到的向量在语义相似度计算中的有效性。在预处理环节，结合正则清洗、jieba 分词与停用词去除等技术，保障输入数据的质量；在模型设计中，通过调整窗口大小、嵌入维度等参数以适应小数据集训练；最后结合余弦相似度进行向量效果验证，为后续词向量应用与优化提供参考。

2. 研究方法

本次实验分为三部分，数据分词、Skip-Gram 训练词向量表示以及计算相似度。

2.1 数据分词

在进行分词之前，模型首先利用 Python 的正则表达式模块 re 预处理文本，仅保留其中的中文和英文字符。具体做法是基于 Unicode 编码定义需要保留的字符范围，并使用 re.findall 函数匹配符合规则的字符，从而筛选出仅包含中文和英文的文本。

接下来，采用 jieba 进行分词。jieba 分词的核心原理是结合前缀词典匹配和基于隐马尔可夫模型（HMM, Hidden Markov Model）的序列标注方法。

- **前缀词典匹配**：jieba 内置了一个包含丰富词汇的前缀词典，其中存储了大量常见词语的前缀信息及其词性标注。在分词过程中，jieba 会基于该词典进行最长匹配，优先选择匹配最长的词语作为分词结果。
- **HMM 处理新词**：HMM 是一种统计模型，能够描述观测序列（文本）与隐藏状态序列（词性或词边界）之间的关系。在 jieba 中，HMM 模型用于识别新词或长词，尤其是在词典无法匹配时，HMM 通过计算汉字序列的概率分布，预测合理的分词方案。

jieba 在分词时，会先构建一个 DAG（有向无环图，Directed Acyclic Graph），其中每个节点代表可能的词语，边表示词语之间的连接关系。随后，jieba 采用动态规划（Dynamic Programming, DP） 算法，寻找最大概率路径，即最优的分词结果。

最终的分词结果大致如下：

马晓旭 意外 受伤 国奥 警惕 无奈 大雨 格外
商瑞华 首战 复仇 心切 中国 玫瑰 美国 方式
冠军 球队 迎新 欢乐 派对 黄旭获 大奖 张军
辽足 签约 危机 引 注册 难关 高层 威逼利诱
揭秘 谢亚龙 带走 总局 电话 骗局 复制 南杨

可以发现确实省略了中文中的停用词。

2.2 Skip-Gram 训练词向量

Skip-Gram 的核心思想是：对于每个中心词，希望它的邻近词在该中心词的条件下出现的概率最大化。例如，在分词结果的某一行：“一直”“懒”“现在”“终于”“动”“起来”了，若选取“终于”作为中心词，目标是通过训练使条件概率 P(“一直”,“懒”,“现在”,“动”,“起来”｜“终于”)最大化。Skip-Gram 模型通常设定一个窗口大小，以限制中心词所涉及的邻近词数量。

在训练过程中，每个分词结果首先被转换为独热编码（One-Hot Encoding）。这些独热向量经过隐层的权重矩阵映射到低维空间，生成密集词向量。输出层使用 Softmax 分类器，预测目标词的上下文词语。在训练完成后，最终得到的词向量由输出层权重矩阵中的嵌入向量表示，并用于后续的 NLP 任务。

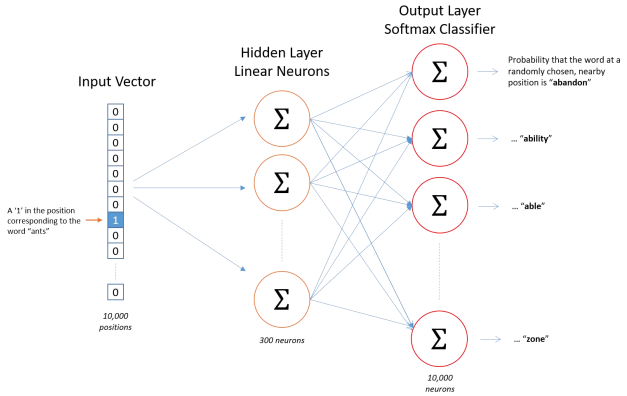

图 1 Skip-Gram 模型结构

2.3 相似度计算

本次实验采用 余弦相似度计算词向量之间的相似程度。

余弦相似度的核心思想是利用余弦公式 计算两个向量的夹角余弦值。在常规计算中，余弦公式通常应用于二维或三维空间，但它可以自然地扩展到高维空间。在词向量表示中，两个词向量的夹角越小，其余弦相似度越接近 1，说明它们的语义相似度越高。

最终，通过计算多个词向量与目标词的余弦相似度，并对结果进行排序，即可得出与目标词最相似的若干个词。

3. 实验与结果分析

3.1 数据处理

因为课堂提供的数据集过大，训练成本太高，因此我选择了一个较小的数据集，一个 10000 行的中文对话数据集，数据集格式如下：

不洗个头吗，翠花　哼！
前夫哥小时候就可爱　小时候脸就不小哇！
最近在干嘛呢，　没干嘛
适合自己的才是最好的！　是滴
没有人夸一下我的自动回复吗！　怎么夸。
我有酒把你心交给我　我有故事，你能接受吗？
我失眠的终点只会是没钱花……　我也没钱花只能想

针对本次实验的模型，在数据分词之前只需进行去除停用词的操作。而数据分词用到了一定的算法模型，将在后面详细阐述。

去除停用词：什么是停用词呢？观察给出的数据，我大致认为停用词就是如标点符号，连接词以及频繁出现但意义不大的词，中文的停用词具体来说包括以下几种：

1. 常见词语：像“的”、“是”、“在”、“和”、“也”、“但”等常见的连接词、副词、介词等，它们在文本中频繁出现但对整体语义影响较小。

2. 数字和符号：数字、标点符号等一些不携带特定语义的词语通常也会被视作停用词，因为它们对文本内容的分析没有帮助。

3. 特定领域的停用词：在某些特定的领域中，会有一些词语对整体语义影响较小，可以视作停用词。例如，在医学领域，常见的“疾病”、“症状”等词语在一些文本处理任务中可能被视作停用词。

4. 人称代词：像“我”、“你”、“他”、“她”等人称代词在一些文本处理任务中可能也会被视作停用词，因为它们通常不会影响整个文本的语义。

那么，为什么要去除停用词呢？查阅资料之后，我总结了以下几个原因：

1. 降低噪音影响：停用词出现频率高，但对文本内容的理解和分析并没有太大帮助，因此移除这些词语可以降低文本数据的噪音水平，有助于提取关键信息。

2. 减少维度：在自然语言处理任务中，文本数据往往会经过词袋模型或者词嵌入等方式表示为向量。如果不移除停用词，这些常见的无意义词语会导致向量维度过高，增加模型训练和计算的复杂度。移除停用词可以减少向量维度，提高计算效率。

3. 改善模型性能：在一些文本分析任务中，移除停用词可以提高模型的性能和准确性。因为停用词通常出现在大多数文本中，如果保留这些词语，可能会造成模

型过度拟合，而移除停用词可以使模型更加关注那些真正具有区分性的词语。

4. 提升文本质量：移除停用词可以使文本更加干净和精炼，更符合人类的阅读习惯。去除了大量的常见词语后，文本更容易被理解和分析，也更有助于后续的文本挖掘和语义分析。

3.2 实验流程

首先运行 dataprocess.py 对数据进行去除停用词、分词处理，得到分词后的数据。

在进行训练之前，首先基于数据集对模型参数进行了修改。因为新的数据集数据量较小（10000 行），因此我将 batchsize 调成了 50，保证模型得到充分的训练，同时将 SkipGram 的窗口从 5 改成了 4，因为新的数据集每行的文字量较小，不适合太大的窗口。同样是为了适应更小的数据集，将嵌入的维数从 100 改为 50。

我舍弃了原本对模型轮数的计算方式，针对较小的数据集进行 100000 轮的训练。

最终运行 test.py 测试模型的效果。

想要系统地评估词向量嵌入的效果，对于一个未经过人工标注的数据集来说难度还是比较大的。本次实验使用的数据集原本并不是用来进行词向量嵌入，因此使用系统的方式检验模型的效果较为困难。

那么，按照常规的办法，我首先绘制 loss 的变化图来判断模型的收敛效果如何。

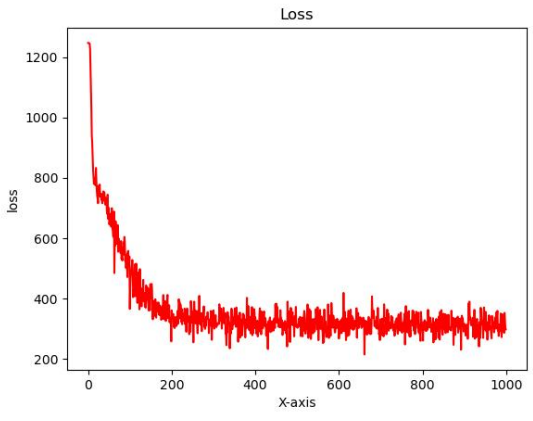

图 2 loss 收敛图

我们可以发现 loss 从原本的 1000 以上收敛到了 300 左右，可以说收敛效果还是很好的，同时收敛速度较快；学习率的取值也较为合理，在之后没有出现过拟合的情况。

最后，我运行 test.py，通过直接判断的办法判断得到的词向量效果是否好。

我随机测试了几个词：

```
Word: 口红
('口红', 0.9999999999999998)
('包包', 0.6963334670844933)
('热情', 0.5538689648888622)
('实话', 0.5445569795669907)
('好久', 0.5196675909432629)
('有钱', 0.5132556411700148)
('名字', 0.5103054846225729)
('遇到', 0.5094798678291534)
('好笑', 0.5037539677213758)
('黑色', 0.5002116822483298)
('对象', 0.4870239091672058)
```

```
Word: 生日
('生日', 0.9999999999999999)
('害羞', 0.7294738354283237)
('生日快乐', 0.6997131604999214)
('棒棒', 0.5909274155047397)
('很快', 0.5857306306731586)
('竟然', 0.5744250390628869)
('祝福', 0.518124329801203)
('第一个', 0.5105363224955308)
('好巧', 0.5061638464879825)
('节日快乐', 0.5034145941028492)
('嫌弃', 0.4972569158351275)
('过分', 0.4897105703660365)
('欢迎', 0.48958500899756496)
('人心', 0.4839225572678)
('走开', 0.4835809208009395)
```

```
Word: 美丽
('美丽', 1.0000000000000004)
('阳光', 0.6758150217113373)
('垃圾', 0.6011572434028885)
('算是', 0.5837150515023402)
('太难', 0.5649883930302103)
('只不过', 0.5607370432368056)
('温暖', 0.5411927420536478)
('存在', 0.5337723825630012)
('办法', 0.5196500157373265)
('欢迎', 0.5189407776416987)
('锻炼', 0.5108531919998007)
('人心', 0.5098894015911483)
('建议', 0.5047474243381326)
('真棒', 0.503253499277314)
('愉快', 0.49540788827557863)
```

```
Word: 人心
('人心', 0.9999999999999999)
('温暖', 0.6354729641046508)
('快点', 0.5976005523176067)
('脖子', 0.5677375625036569)
('赚钱', 0.5499668822768853)
('爱情', 0.5441011189909932)
('最佳', 0.5313347121926806)
('很棒', 0.5248377356316576)
('头发', 0.5228635401872516)
('意思', 0.5219849003798691)
('美丽', 0.5098894015911483)
('热情', 0.508969565588453)
('干净', 0.5047912166563473)
('幸运', 0.5000105129218899)
('不让', 0.49800052079149165)
```

```
Word: 幽默
('幽默', 1.0)
('搞笑', 0.6158841126902197)
('说话', 0.61257566572815)
('建议', 0.6004222222979729)
('嫌弃', 0.5704064965795432)
('低调', 0.5392669016400566)
('垃圾', 0.5251895931742638)
('对象', 0.5107608834152346)
('强行', 0.4972543701401088 7)
('来不及', 0.4884394257910293)
('口味', 0.487380070698875)
('黑色', 0.48674886063683953)
('下雨', 0.4843510122372599)
('运动', 0.48045673308376624)
```

```
Word: 说话
('说话', 1.0000000000000002)
('幽默', 0.61257566572815)
('低调', 0.612151141508077)
('搞笑', 0.5794755670908859)
('明白', 0.5586841812513867)
('受不了', 0.5361787085025527)
('经常', 0.5242114749295287)
('能量', 0.5196279810025273)
('样子', 0.518139542300529)
('有钱', 0.5027698646691432)
('联系', 0.4907799857878028)
('自由', 0.4878212441650048)
('未来', 0.4848966962210115)
('放不下', 0.4764210131195853)
('怀念', 0.4747905667450752)
('放松', 0.4703874909292872)
('收拾', 0.4614070753043682)
```

我们发现在如此小的样本下，模型依然训练出了词语之间不错的相似性。根据常识这些词汇都有比较高的相似性。

4. 结论

本次实验我最大的收获就是学习到了一个词向量嵌入算法 Skip-Gram，同时通过相似度的计算明白了词向量嵌入的意义。词向量不仅是将稀疏的独热矩阵转化成了稠密矩阵方便训练，同时，相似度的检验也验证了这种压缩编码的合理性，只有相似的词语嵌入出的向量才有较高的相似性。

之前我对词向量嵌入没有一个清晰的认识，以为只有自然语言才能够进行词向量嵌入，在处理交叉学科问题，如基于分子描述符（一种字符串形式的分子编码）预测分子性质，当用 NPL 模型处理这类问题时，我从未想过使用词向量嵌入，而是直接使用暴力的独热编码。这次实验给了我启发，让我能够尝试一种新的数据读入方式，期待能够得到更好的训练结果。

5. 参考文献

[1] July, "Word2Vec 原理详解：从 Skip-Gram 到 CBOW", CSDN 博客, 2019.
[2] 龙心尘, "Word2Vec 中的数学原理详解", 知乎专栏, 2017.
[3] 寒小阳, "Word2Vec 原理(一) CBOW 与 Skip-Gram 模型基础", 知乎专栏, 2020.
[4] zyx2495310073, "Word2Vec 模型详解：从理论到实践", CSDN 博客, 2022.