NLP Lab #2 - word2vec 模型

Part #1 实验简介

本次实验要求使用 word2vec 架构，利用两种预测模型（CBOW 和 Skip-Gram）以及两种优化方法（Hierarchy-Softmax 和 Negative-Sampling）的组合，即四种学习策略，在维基百科英文语料上进行词向量学习。

本次实验的重心在于测试不同训练策略的组合，以获得对于 word2vec 架构和训练策略的感性认知。此外，如何处理从网页爬取所得的文本数据也是本次实验的重点难点。

Part #2 技术细节

a) 网页基本处理

首先，本次实验的基础步骤，是将爬取所得的 HTML 结构文本进行简单的文本提取。在这里，本人仅介绍本人所使用的处理流程：

1. 根据 HTML 结构中的 `<page>` 和 `</page>` 对，将文本划分为多个页面单元，以方便后续处理；
2. 删除每个页面中明显无用的部分，如其余的 HTML 标签和维基百科的语言选项；
3. 利用正则表达式，将文本中所有的转移字符对应的字符串（如 `&amp` 等字符串）删除；
4. 删除大括号对，即 `{}` 中间的任何内容；
5. 删除明显不属于文本的字符，如 `#` 等；
6. 删除网址以及其附属内容。

b) 文本符号化

仅仅使用从网页中提取的文本是不充分的，我们需要对于文本中所有的 token 进行如下的一系列处理：

1. 对于 [[A|B|C]] 的处理

维基百科中充斥着许多术语，这些术语会被嵌套在双层中括号中。有时，双层中括号中还会出现竖线，以代表同一术语的不同表达方式。如果不处理这一类符号，将会导致 token 库中出现许多单边括号和词语结合产生的 token。为了解决这一问题，本人构思了如下三种处理方法：

1. 直接去掉任何中括号并将中括号中的竖线替换为空格，因此每个近义选项都会原地保留；
2. 从多个竖线分隔出的短语中随机选择一个填入句子中，余下的直接丢弃；
3. 如果有多个同义选项，那么为每一个同义选项重复整个句子一次；如果同一个句子有多个多项术语，那么对于它们的任意组合重复句子。

在最终的实现里，本人尝试了方法 1 和 2 的效果，没有明显差别；本人没有实现方案 3，因为其实现成本过高并且会大大增加语料的规模。

2. 对于正常文本标点的处理

正常的文本中时常会出现 `.`、`?`、`,` 等标点符号，这些符号有时会和单词相互连接，而 `word2vec` 仅依靠空格、回车、缩进等来区分不同的词语。因此，如果不加以额外处理，所有携带标点的词语都会被 word2vec 识别为与原单词不同的词语。

为了解决这个问题，一个符合直觉的处理方案是，将所有这些标点符号的左右侧各插入一个空格。这个方案也确实是本步骤处理的主要方式。但是，这一简单的处理方法会产生如下的问题：

1. 单词 `U.S.` 会被拆成多个部分，即首字符缩写词会被错误处理；
2. 数字 `1,000,000` 以及 `123.45` 也会被拆成多个部分（实际上我们并不关心过多的数字组合，但是这个问题值得思考）；
3. 单词 `ice-cream` 会被拆成多个部分，即组合词会被特殊处理；
4. 符号 `...` 即省略号会被拆散；

   ……

本人在这里的对于问题的处理一部分是启发式的，即在开发后期，本人在单词表中看到异常 token 才开始思考其出现原因，因此以上的总结可能并不完善。

基于如上的一些观察，本人最后的解决方案是进行 token 级别的标点处理，即将文本先根据空格分割，接着对于每个分割所得的部分，考虑其是否需要被进一步分割。

值得提到的是，本人遇到了一些有趣的情况，这些情况往往与句点有关：

1. 某次观察词库时，本人发现了 `U.S..`，这个是由单词 `U.S.` 被放到句尾时产生的。根据本人所知，另一种书写风格是最后一个字符是句点的缩略词放在句尾时，若句尾标点是句点那么该句点应当省略；
2. 某次观察词库时，本人发现了 `..`，值得注意的是这个并不是因为错误的分割所导致的。事实上，文本中的确存在不少的 “双句点省略号”。

3. 单词大小写的处理

之所以要处理单词的大小写，是因为我们不希望同一个语义的单次被独立多次统计。具体来说，当一个首字母非大写的词语被放到句首时，其首字母将被大写，如果不加以处理，它们在 `word2vec` 中会被当做两个词语来处理。

一种符合直觉的解决方案（其实也已经做的足够好）是将所有字母均视为小写处理。但是，这种处理方式将会导致如下的两类问题：

1. `US` 会被处理为 `us`，前者是国家名称而后者是一个代词；
2. `China` 是国家名称而 `china` 有瓷器的含义，但是两者在这种方案下会被当做同一个词语；

   ……

本人给出了一个更完善的处理方案，这一方案基于统计意义：

1. 在修改所有单词的大小写时，先区分所有大小写，统计所有词汇的频率；
2. 考虑第一个字母大小写均出现过的单词，统计其首字母大写的次数 n_1，并统计其位于句首的次数 n_2；
3. 如果 $\frac{n_2}{n_1} > 1-\delta$，那么我们断定这个单词所有的大写是因为其被放到句首所导致的，我们断定其仅有小写形式；如果 $\frac{n_2}{n_1} < \delta$，那我们断定这个单词只有大写形式；否则，本人承认此单词本身同时存在大写与小写形式，在这种情况下，本人不额外做任何操作。在实验中，本人取 $\delta = 0.01$，没有观察到明显问题。

c) 未学习词汇的词向量处理

在本次实验的预测任务中，出现了在语料库中出现频率小于 5 的词汇。这样的词汇不会被 word2vec 模型所学习。为了顺利完成预测任务，本人考虑使用所有词汇的算数平均向量来代替未学习的向量。本人承认这一做法比较草率，但是根据本人观察，对于这一类单词对应向量的不同处理方式最终导致的效果变化，小于多次重复同一实验时所产生的的随机涨落，因此本人采用了这种处理方法。

Part #3 实验结果与分析

在所有测试中，本人使用所有词汇对的词向量相似度和人工相似度的总相似度来作为性能评价指标。

总相似度*	CBOW	Skip-Gram
HS	69.38 (1.10)	71.53 (1.33)
NS	69.02 (0.44)	71.38 (0.57)

*数据项均为数据集总相似度的 100 倍保留 2 位小数后的结果，每个数据项重复了 5 次实验，括号内是 5 次实验结果的方差；

结果分析

1. 从结果来看，和老师上课分析的一样，Skip-Gram 由于预测任务较多，表现更好；HS 训练所得词向量效果的方差比 NS 的高，但是词向量的平均效果略好。
2. 在实验中，训练速度的大小关系是 `NS-CBOW` > `HS-CBOW` >> `HS-SG` > `NS-SG`，这一结果表明 `CBOW` 的预测任务确实比 `SG` 少得多。