文本表征学习 Lab 2 Word2vec

PB22111599 杨映川

Pt.1 实验概述

基于enwik8语料库，分别使用CBOW(Continuous Bag of Words)和Skip-gram算法建立模型，并分别结合Hierarchical Softmax和Negative Sample算法进行优化，并使用Spearman相关系数，以老师提供的`gold_standard.txt`文件为标准，对四种算法进行性能评估。

- **CBOW + HS**
- **CBOW + NS**
- **SG + HS**
- **SG + NS**

Pt.2 数据处理与实现过程

2.1 预处理

2.1.1 为什么要预处理？

源代码在分词时会使用空格、制表符、换行符作为分词符号。但Wiki的语料库中存在大量的标注符号如`[[Mercury|Venus|Earth]]`或`**{{forum}}`或`&quot`等。
对网页数据完全不做预处理的情况下会发现这些符号会比较显著地影响分词这项任务，导致一些糟糕的情况发生（比如`China`和`|China`被认定为两个不同的词，但彼此有最大的cosine distance）。

2.1.2 预处理过程

- **适当修改** 将分词符号扩充为`space`、`\t`、`\n`、`.`、`,`、`(`、`)`、`[`、`]`、`{`、`}`、`|`、`!`、`?`、`;`、`/`、`&`、`:`、`=`、`+`、`*`。
- **例外** 值得注意的是`'`和`-`被保留作了词语的一部分，这是出于对于形如`ice-cream`和`Alice's`的词语的考虑。
- **局限性** 这样的做法未能对句首引号做正确的处理，以及对于`U.S.`这样的词会进行错误的分词。

2.2 模型训练

编译

使用源代码中提供的shell脚本文件`demo-word.sh`获取数据集并对数据集进行训练。

参数设置

- `-size 200 -window 5 -min-count 5 -iter 15`，即词向量的维度为200，窗口大小为+/-5词，词频过滤为5；
- `-cbow <bool>`控制是否使用CBOW算法，否则使用Skip-gram算法；
- `-hs <bool>`控制是否使用Hierarchical Softmax算法进行优化；

- `-negative <num>`控制负采样的个数（在此次实验中均使用了25个负采样，尽量地提高了robustness）。

Pt.3 训练结果

3.1 数据处理

- 计算出相应词对的cosine distance，并使用这一列数据与`gold_standard.txt`中的数据计算Spearman相关系数，将其作为算法性能的评估标准。
- 语料库训练完成后无法找到“asylum madhouse”词对和“cup tableware”词对，出于不影响后续评估的考虑，将这两个词对删除。

3.2 结果

相关系数R(p)	CBOW	Skip-gram
HS	0.682(6.953e-29)	0.746(5.863e-37)
NS	0.746(5.917e-37)	0.772(5.972e-41)

- 括号外的R值是相关系数(-1 < R < 1)，越接近1表明越正相关；
- 括号内的p值告诉我们观察到的相关性是偶然出现的概率有多大，较低的p值（如<0.05）意味着它们之间确实存在统计上显著的相关性。

Pt.4 分析

1. 可见Skip-gram是要优于CBOW的，因为使用的语料库相对比较庞大，而Skip-gram预测任务更多，捕捉到的信息粒度也更细致；
2. NS算法要优于HS算法，因为使用的语料库相对比较庞大，且负采样的数量达到了较高的25个；
3. 四种情况的p值都非常低，表明相关性效果非常好；

```
Vocab size: 101697
Words in train file: 15346695
```

4. 训练时间上 **CBOW+NS < CBOW+HS << SG+HS < SG+NS**，最快的约8分钟，最慢的达到半个多小时。因为CBOW的预测任务要明显多余Skip-gram的预测任务。

Pt.5 一些探索

源代码中还有许多比较有趣的函数实现以及配套的脚本文件。使用训练效果最好的SG+NS一个组合尝试运行这些函数。

5.1 word-analogy.c

这个文件要求输入三个词A、B、C并基于模型预测第四个词，原理基于`A - B + C = ?`，即“A is to B as C is to D.”

example 1

```
Enter three words (EXIT to break): king man queen

Word: king  Position in vocabulary: 860

Word: man  Position in vocabulary: 633

Word: queen  Position in vocabulary: 5346

                                              Word              Distance
------------------------------------------------------------------------
                                             woman              0.564011
                                           balding              0.468434
                                          Westenra              0.466000
                                            steamy              0.464106
                                          Millarca              0.455878
```

国王&男人--皇后&女人

example 2

```
Enter three words (EXIT to break): Beijing China Tokyo

Word: Beijing  Position in vocabulary: 4165

Word: China  Position in vocabulary: 506

Word: Tokyo  Position in vocabulary: 6175

                                              Word              Distance
------------------------------------------------------------------------
                                             Japan              0.604366
                                        Shimonoseki              0.546541
                                           Michiko              0.541244
                                            Toyama              0.540558
                                             Korea              0.536292
```

北京&中国--东京&日本

example 3

```
Enter three words (EXIT to break): human alien water

Word: human  Position in vocabulary: 328

Word: alien  Position in vocabulary: 5883

Word: water  Position in vocabulary: 371

                                              Word              Distance
------------------------------------------------------------------------
                                            slurry              0.464934
                                         absorptive             0.458701
                                        superheated             0.445751
                                           chiller              0.443941
                                             marsh              0.439031
```

人类&外星人--水&泥浆

5.2 word2phrase.c

这个程序可以将两个或多个经常一起出现的单词（如“New York”）组合成单个短语标记（如“New_York”），可以大大提高word2vec对上下文的理解力。

5.3 distance.c

这个程序会读入训练好的模型，并要求输入一个单词或句子，然后返回N个与输入cosine distance最佳的词语。

example 1

```
Enter word or sentence (EXIT to break): Abraham

Word: Abraham  Position in vocabulary: 2787

                                              Word       Cosine distance
------------------------------------------------------------------------
                                           Lincoln              0.646687
                                         Lincoln''              0.607963
                                            Maslow              0.601324
                                            Moivre              0.600385
                                         Abraham's              0.561253
                                           Ishmael              0.555809
```

Abraham -> Lincoln

example 2

```
Enter word or sentence (EXIT to break): Linux

Word: Linux  Position in vocabulary: 3367

                                             Word       Cosine distance
--------------------------------------------------------------------------
                                           Ubuntu              0.730639
                                           NetBSD              0.724555
                                           LiveCD              0.701432
                                    Cross-platform             0.698189
                                         Unix-like             0.696128
                                              GNU              0.695766
                                              KDE              0.695358
                                           OpenVMS             0.695014
```

Linux -> Ubuntu

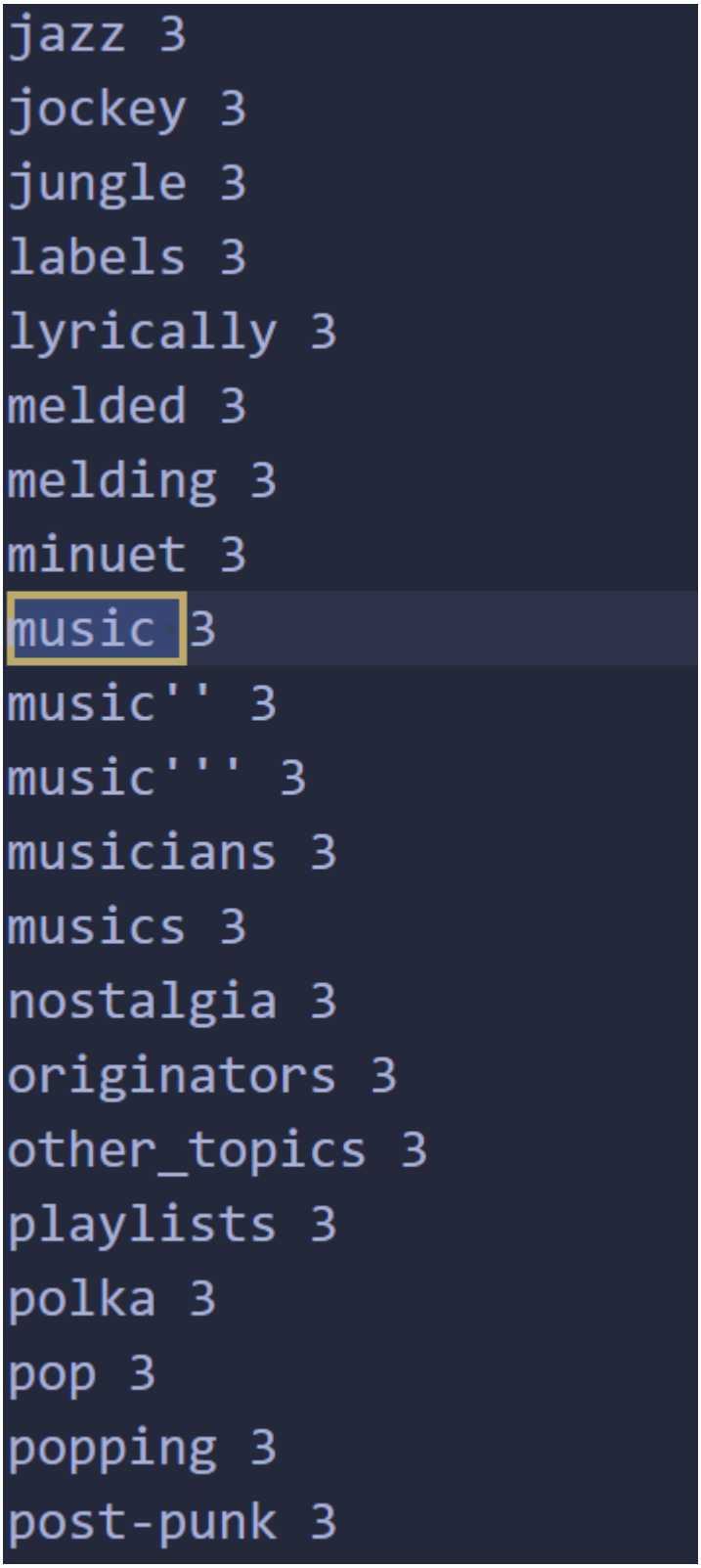
example 3

```
Enter word or sentence (EXIT to break): alien

Word: alien  Position in vocabulary: 5883

                                             Word       Cosine distance
--------------------------------------------------------------------------
                                  extraterrestrial              0.662890
                                           aliens               0.630504
                                 extraterrestrials              0.616615
                                  Extraterrestrial              0.581638
                                        Earth-like              0.559981
                                         reptilian              0.556821
                                            planet              0.543356
                                        intelligent             0.542142
                                          humanoid              0.540773
```

alien -> extraterrestrial

5.4 demo-classes.sh

这个程序可以基于训练好的模型对词汇进行分类（基于K-means算法）



可以看到与music同类的有jazz, jungle, pop, post-punk, raggae, rap, rock, synthpop techno, grunge等等一系列的音乐风格。

5.5 demo-train-big-model-v1.sh

这个文件中包含一段对Wikipedia中的文本进行预处理的代码，能够删除一些非文本的符号。

```
  # Written by Matt Mahoney, June 10, 2006.  This program is released to the
public domain.
```

```
  $/=">";                     # input record separator
  while (<>) {
    if (/<text /) {$text=1;}  # remove all but between <text> ... </text>
    if (/#redirect/i) {$text=0;}  # remove #REDIRECT
    if ($text) {

      # Remove any text not normally visible
      if (/<\/text>/) {$text=0;}
      s/<.*>//;               # remove xml tags
      s/&amp;/&/g;            # decode URL encoded chars
      s/&lt;/</g;
      s/&gt;/>/g;
      s/<ref[^<]*<\/ref>//g;  # remove references <ref...> ... </ref>
      s/<[^>]*>//g;           # remove xhtml tags
      s/\[http:[^] ]*/[/g;    # remove normal url, preserve visible text
      s/\|thumb//ig;          # remove images links, preserve caption
      s/\|left//ig;
      s/\|right//ig;
      s/\|\d+px//ig;
      s/\[\[image:[^\[\]]*\|//ig;
      s/\[\[category:([^|\]]*)[^]]*\]\]/[[$1]]/ig;  # show categories
without markup
      s/\[\[[a-z\-]*:[^\]]*\]\]//g;  # remove links to other languages
      s/\[\[[^\|\]]*\|/[[/g;  # remove wiki url, preserve visible text
      s/{{[^}]*}}//g;         # remove {{icons}} and {tables}
      s/{[^}]*}//g;
      s/\[//g;                # remove [ and ]
      s/\]//g;
      s/&[^;]*;/ /g;          # remove URL encoded chars

      $_=" $_ ";
      chop;
      print $_;
    }
  }
```

Pt.6 Debugging Problems

1. 在Linux上对C语言文件进行编译时，如果使用了`math.h`库中的函数，需要在命令行添加`-lm`指令；
2. 学习了如何在Linux中使用shell脚本文件。