基于 Gensim 的 Word2Vec 实践

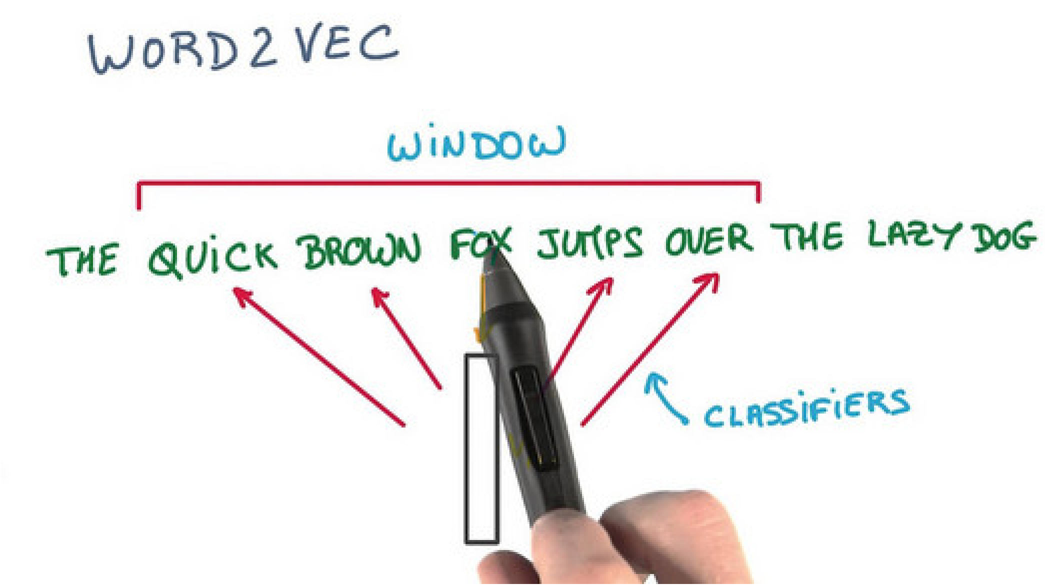
时间 2017-01-20 00:12:08 某熊的全栈之路 (/sites/aAV32u6)

原文 https://segmentfault.com/a/1190000008173404 (https://segmentfault.com/a/1190000008173404?utm_source=tuicool&utm_medium=referral)

主题 word2vec (/topics/11120166)

Word2Vec

基于 Gensim 的 Word2Vec 实践 (https://zhuanlan.zhihu.com/p/24961011) ，从属于笔者的 程序猿的数据科学与机器学习实战手册 (https://github.com/wxyyxc1992/DataScience-And-MachineLearning-Handbook-For-Coders) ，代码参考 gensim.ipynb (https://github.com/wxyyxc1992/DataScience-And-MachineLearning-Handbook-For-Coders/blob/master/code/python/nlp/genism/gensim.ipynb) 。推荐前置阅读 Python语法速览与机器学习开发环境搭建 (https://zhuanlan.zhihu.com/p/24536868) ， Scikit-Learn 备忘录 (https://zhuanlan.zhihu.com/p/24770526) 。



qU)

- 2. Gowild 王昊奋：知识图谱所面临的五大挑战 (/articles/BVNn2my)
- 3. 一文概述2017年深度学习NLP重大进展与趋势 (/articles/7FBrQ3F)
- 4. Google 何时回归中国？这个问题也许根本就不存在 (/articles/2euUviy)
- 5. 今日头条人工智能实验室主任李航：如何构建拥有长期记忆的智能问答系统 (/articles/amaMfyi)

- Word2Vec Tutorial (https://rare-technologies.com/word2vec-tutorial/)
- Getting Started with Word2Vec and GloVe in Python (http://textminingonline.com/getting-started-with-word2vec-and-glove-in-python)

模型创建

Gensim (http://radimrehurek.com/gensim/models/word2vec.html) 中 Word2Vec 模型的期望输入是进过分词的句子列表，即是某个二维数组。这里我们暂时使用 Python 内置的数组，不过其在输入数据集较大的情况下会占用大量的 RAM。Gensim 本身只是要求能够迭代的有序句子列表，因此在工程实践中我们可以使用自定义的生成器，只在内存中保存单条语句。

```
# 引入 word2vec
from gensim.models import word2vec

# 引入日志配置
import logging

logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=lo
gging.INFO)

# 引入数据集
raw_sentences = ["the quick brown fox jumps over the lazy dogs","yoyoyo you go h
ome now to sleep"]

# 切分词汇
sentences= [s.encode('utf-8').split() for s in sentences]

# 构建模型
model = word2vec.Word2Vec(sentences, min_count=1)

# 进行相关性比较
model.similarity('dogs','you')
```

这里我们调用 `Word2Vec` 创建模型实际上会对数据执行两次迭代操作，第一轮操作会统计词频来构建内部的词典数结构，第二轮操作会进行神经网络训练，而这两个步骤是可以分步进行的，这样对于某些不可重复的流（譬如 Kafka 等流式数据中）可以手动控制：

```
model = gensim.models.Word2Vec(iter=1)  # an empty model, no training yet
model.build_vocab(some_sentences)  # can be a non-repeatable, 1-pass generator
model.train(other_sentences)  # can be a non-repeatable, 1-pass generator
```

Word2Vec 参数

- min_count

```
model = Word2Vec(sentences, min_count=10)  # default value is 5
```

在不同大小的语料集中，我们对于基准词频的需求也是不一样的。譬如在较大的语料集中，我们希望忽略那些只出现过一两次的单词，这里我们就可以通过设置 `min_count` 参数进行控制。一般而言，合理的参数值会设置在0~100之间。

- size

`size` 参数主要是用来设置神经网络的层数，Word2Vec 中的默认值是设置为100层。更大的层次设置意味着更多的输入数据，不过也能提升整体的准确度，合理的设置范围为 10~数百。

```
model = Word2Vec(sentences, size=200)  # default value is 100
```

- workers

`workers` 参数用于设置并发训练时候的线程数，不过仅当 `Cython` 安装的情况下才会起作用：

```
model = Word2Vec(sentences, workers=4) # default = 1 worker = no parallelization
```

外部语料集

在真实的训练场景中我们往往会使用较大的语料集进行训练，譬如这里以 Word2Vec 官方的 text8 (http://mattmahoney.net/dc/text8.zip) 为例，只要改变模型中的语料集开源即可：

```
sentences = word2vec.Text8Corpus('text8')
model = word2vec.Word2Vec(sentences, size=200)
```

这里语料集中的语句是经过分词的，因此可以直接使用。笔者在第一次使用该类时报错了，因此把 Gensim 中的源代码贴一下，也方便以后自定义处理其他语料集：

```
class Text8Corpus(object):
    """Iterate over sentences from the "text8" corpus, unzipped from http://matt
mahoney.net/dc/text8.zip ."""
    def __init__(self, fname, max_sentence_length=MAX_WORDS_IN_BATCH):
        self.fname = fname
        self.max_sentence_length = max_sentence_length

    def __iter__(self):
        # the entire corpus is one gigantic line -- there are no sentence marks
 at all
        # so just split the sequence of tokens arbitrarily: 1 sentence = 1000 to
kens
        sentence, rest = [], b''
        with utils.smart_open(self.fname) as fin:
            while True:
                text = rest + fin.read(8192)  # avoid loading the entire file (=
1 line) into RAM
                if text == rest:  # EOF
                    words = utils.to_unicode(text).split()
                    sentence.extend(words)  # return the last chunk of words, to
o (may be shorter/longer)
                    if sentence:
                        yield sentence
                    break
                last_token = text.rfind(b' ')  # last token may have been split
 in two... keep for next iteration
                words, rest = (utils.to_unicode(text[:last_token]).split(),
                               text[last_token:].strip()) if last_token >= 0 els
e ([], text)
                sentence.extend(words)
                while len(sentence) >= self.max_sentence_length:
                    yield sentence[:self.max_sentence_length]
                    sentence = sentence[self.max_sentence_length:]
```

我们在上文中也提及，如果是对于大量的输入语料集或者需要整合磁盘上多个文件夹下的数据，我们可以以迭代器的方式而不是一次性将全部内容读取到内存中来节省 RAM 空间：

```
class MySentences(object):
    def __init__(self, dirname):
        self.dirname = dirname

    def __iter__(self):
        for fname in os.listdir(self.dirname):
            for line in open(os.path.join(self.dirname, fname)):
                yield line.split()

sentences = MySentences('/some/directory') # a memory-friendly iterator
model = gensim.models.Word2Vec(sentences)
```

模型保存与读取

```
model.save('text8.model')
2015-02-24 11:19:26,059 : INFO : saving Word2Vec object under text8.model, separ
ately None
2015-02-24 11:19:26,060 : INFO : not storing attribute syn0norm
2015-02-24 11:19:26,060 : INFO : storing numpy array 'syn0' to text8.model.syn0.
npy
2015-02-24 11:19:26,742 : INFO : storing numpy array 'syn1' to text8.model.syn1.
npy

model1 = Word2Vec.load('text8.model')

model.save_word2vec_format('text.model.bin', binary=True)
2015-02-24 11:19:52,341 : INFO : storing 71290x200 projection weights into text.
model.bin

model1 = word2vec.Word2Vec.load_word2vec_format('text.model.bin', binary=True)
2015-02-24 11:22:08,185 : INFO : loading projection weights from text.model.bin
2015-02-24 11:22:10,322 : INFO : loaded (71290, 200) matrix from text.model.bin
2015-02-24 11:22:10,322 : INFO : precomputing L2-norms of word weight vectors
```

模型预测

Word2Vec 最著名的效果即是以语义化的方式推断出相似词汇：

```
model.most_similar(positive=['woman', 'king'], negative=['man'], topn=1)
[('queen', 0.50882536)]
model.doesnt_match("breakfast cereal dinner lunch";.split())
'cereal'
model.similarity('woman', 'man')
0.73723527
model.most_similar(['man'])
[(u'woman', 0.5686948895454407),
 (u'girl', 0.4957364797592163),
 (u'young', 0.4457539916038513),
 (u'luckiest', 0.4420866759529114),
 (u'serpent', 0.42716869711875916),
 (u'girls', 0.42680859565734863),
 (u'smokes', 0.4265017509460449),
 (u'creature', 0.4227582812309265),
 (u'robot', 0.417464017868042),
 (u'mortal', 0.41728296875953674)]
```

如果我们希望直接获取某个单词的向量表示，直接以下标方式访问即可：

```
model['computer']  # raw NumPy vector of a word
array([-0.00449447, -0.00310097,  0.02421786, ...], dtype=float32)
```

模型评估

Word2Vec 的训练属于无监督模型，并没有太多的类似于监督学习里面的客观评判方式，更多的依赖于端应用。Google 之前公开了20000条左右的语法与语义化训练样本，每一条遵循 `A is to B as C is to D` 这个格式，地址在 这里 (https://word2vec.googlecode.com/svn/trunk/questions-words.txt)：

```
model.accuracy('/tmp/questions-words.txt')
2014-02-01 22:14:28,387 : INFO : family: 88.9% (304/342)
2014-02-01 22:29:24,006 : INFO : gram1-adjective-to-adverb: 32.4% (263/812)
2014-02-01 22:36:26,528 : INFO : gram2-opposite: 50.3% (191/380)
2014-02-01 23:00:52,406 : INFO : gram3-comparative: 91.7% (1222/1332)
2014-02-01 23:13:48,[illegible] : INFO : gram4-superlative: 87.9% (617/702)
2014-02-01 23:29:52,268 : INFO : gram5-present-participle: 79.4% (691/870)
2014-02-01 23:57:04,965 : INFO : gram7-past-tense: 67.1% (995/1482)
2014-02-02 00:15:18,525 : INFO : gram8-plural: 89.6% (889/992)
2014-02-02 00:28:18,140 : INFO : gram9-plural-verbs: 68.7% (482/702)
2014-02-02 00:28:18,140 : INFO : total: 74.3% (5654/7614)
```

还是需要强调下，训练集上表现的好也不意味着 Word2Vec 在真实应用中就会表现的很好，还是需要因地制宜。）

- Word2Vec Tutorial (https://rare-technologies.com/word2vec-tutorial/)
- Getting Started with Word2Vec and GloVe in Python (http://textminingonline.com/getting-started-with-word2vec-and-glove-in-python)

模型创建

Gensim (http://radimrehurek.com/gensim/models/word2vec.html) 中 Word2Vec 模型的期望输入是进过分词的句子列表，即是某个二维数组。这里我们暂时使用 Python 内置的数组，不过其在输入数据集较大的情况下会占用大量的 RAM。Gensim 本身只是要求能够迭代的有序句子列表，因此在工程实践中我们可以使用自定义的生成器，只在内存中保存单条语句。

```
# 引入 word2vec
from gensim.models import word2vec

# 引入日志配置
import logging

logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=lo
gging.INFO)

# 引入数据集
raw_sentences = ["the quick brown fox jumps over the lazy dogs","yoyoyo you go h
ome now to sleep"]

# 切分词汇
sentences= [s.encode('utf-8').split() for s in sentences]

# 构建模型
model = word2vec.Word2Vec(sentences, min_count=1)

# 进行相关性比较
model.similarity('dogs','you')
```

这里我们调用 `Word2Vec` 创建模型实际上会对数据执行两次迭代操作，第一轮操作会统计词频来构建内部的词典数结构，第二轮操作会进行神经网络训练，而这两个步骤是可以分步进行的，这样对于某些不可重复的流（譬如 Kafka 等流式数据中）可以手动控制：

```
model = gensim.models.Word2Vec(iter=1)  # an empty model, no training yet
model.build_vocab(some_sentences)  # can be a non-repeatable, 1-pass generator
model.train(other_sentences)  # can be a non-repeatable, 1-pass generator
```

Word2Vec 参数

- min_count

```
model = Word2Vec(sentences, min_count=10)  # default value is 5
```

在不同大小的语料集中，我们对于基准词频的需求也是不一样的。譬如在较大的语料集中，我们希望忽略那些只出现过一两次的单词，这里我们就可以通过设置 `min_count` 参数进行控制。一般而言，合理的参数值会设置在0~100之间。

- size

`size` 参数主要是用来设置神经网络的层数，Word2Vec 中的默认值是设置为100层。更大的层次设置意味着更多的输入数据，不过也能提升整体的准确度，合理的设置范围为 10~数百。

```
model = Word2Vec(sentences, size=200)  # default value is 100
```

- workers

`workers` 参数用于设置并发训练时候的线程数，不过仅当 `Cython` 安装的情况下才会起作用：

```
model = Word2Vec(sentences, workers=4) # default = 1 worker = no parallelization
```

外部语料集

在真实的训练场景中我们往往会使用较大的语料集进行训练，譬如这里以 Word2Vec 官方的 text8 (http://mattmahoney.net/dc/text8.zip) 为例，只要改变模型中的语料集开源即可：

```
sentences = word2vec.Text8Corpus('text8')
model = word2vec.Word2Vec(sentences, size=200)
```

这里语料集中的语句是经过分词的，因此可以直接使用。笔者在第一次使用该类时报错了，因此把 Gensim 中的源代码贴一下，也方便以后自定义处理其他语料集：

```
class Text8Corpus(object):
    """Iterate over sentences from the "text8" corpus, unzipped from http://matt
mahoney.net/dc/text8.zip ."""
    def __init__(self, fname, max_sentence_length=MAX_WORDS_IN_BATCH):
        self.fname = fname
        self.max_sentence_length = max_sentence_length

    def __iter__(self):
        # the entire corpus is one gigantic line -- there are no sentence marks
 at all
        # so just split the sequence of tokens arbitrarily: 1 sentence = 1000 to
kens
        sentence, rest = [], b''
        with utils.smart_open(self.fname) as fin:
            while True:
                text = rest + fin.read(8192)  # avoid loading the entire file (=
1 line) into RAM
                if text == rest:  # EOF
                    words = utils.to_unicode(text).split()
                    sentence.extend(words)  # return the last chunk of words, to
o (may be shorter/longer)
                    if sentence:
                        yield sentence
                    break
                last_token = text.rfind(b' ')  # last token may have been split
 in two... keep for next iteration
                words, rest = (utils.to_unicode(text[:last_token]).split(),
                               text[last_token:].strip()) if last_token >= 0 els
e ([], text)
                sentence.extend(words)
                while len(sentence) >= self.max_sentence_length:
                    yield sentence[:self.max_sentence_length]
                    sentence = sentence[self.max_sentence_length:]
```

我们在上文中也提及，如果是对于大量的输入语料集或者需要整合磁盘上多个文件夹下的数据，我们可以以迭代器的方式而不是一次性将全部内容读取到内存中来节省 RAM 空间：

```
class MySentences(object):
    def __init__(self, dirname):
        self.dirname = dirname

    def __iter__(self):
        for fname in os.listdir(self.dirname):
            for line in open(os.path.join(self.dirname, fname)):
                yield line.split()

sentences = MySentences('/some/directory') # a memory-friendly iterator
model = gensim.models.Word2Vec(sentences)
```

模型保存与读取

```
model.save('text8.model')
2015-02-24 11:19:26,059 : INFO : saving Word2Vec object under text8.model, separ
ately None
2015-02-24 11:19:26,060 : INFO : not storing attribute syn0norm
2015-02-24 11:19:26,060 : INFO : storing numpy array 'syn0' to text8.model.syn0.
npy
2015-02-24 11:19:26,742 : INFO : storing numpy array 'syn1' to text8.model.syn1.
npy

model1 = Word2Vec.load('text8.model')

model.save_word2vec_format('text.model.bin', binary=True)
2015-02-24 11:19:52,341 : INFO : storing 71290x200 projection weights into text.
model.bin

model1 = word2vec.Word2Vec.load_word2vec_format('text.model.bin', binary=True)
2015-02-24 11:22:08,185 : INFO : loading projection weights from text.model.bin
2015-02-24 11:22:10,322 : INFO : loaded (71290, 200) matrix from text.model.bin
2015-02-24 11:22:10,322 : INFO : precomputing L2-norms of word weight vectors
```

模型预测

Word2Vec 最著名的效果即是以语义化的方式推断出相似词汇：

```
model.most_similar(positive=['woman', 'king'], negative=['man'], topn=1)
[('queen', 0.50882536)]
model.doesnt_match("breakfast cereal dinner lunch";.split())
'cereal'
model.similarity('woman', 'man')
0.73723527
model.most_similar(['man'])
[(u'woman', 0.5686948895454407),
 (u'girl', 0.4957364797592163),
 (u'young', 0.4457539916038513),
 (u'luckiest', 0.4420626759529114),
 (u'serpent', 0.42716869711875916),
 (u'girls', 0.42680859565734863),
 (u'smokes', 0.4265017509460449),
 (u'creature', 0.4227582812309265),
 (u'robot', 0.417464017868042),
 (u'mortal', 0.41728296875953674)]
```

如果我们希望直接获取某个单词的向量表示，直接以下标方式访问即可：

```
model['computer']  # raw NumPy vector of a word
array([-0.00449447, -0.00310097,  0.02421786, ...], dtype=float32)
```

模型评估

Word2Vec 的训练属于无监督模型，并没有太多的类似于监督学习里面的客观评判方式，更多的依赖于端应用。Google 之前公开了20000条左右的语法与语义化训练样本，每一条遵循 `A is to B as C is to D` 这个格式，地址在 这里 (https://word2vec.googlecode.com/svn/trunk/questions-words.txt)：

```
model.accuracy('/tmp/questions-words.txt')
2014-02-01 22:14:28,387 : INFO : family: 88.9% (304/342)
2014-02-01 22:29:24,006 : INFO : gram1-adjective-to-adverb: 32.4% (263/812)
2014-02-01 22:36:26,528 : INFO : gram2-opposite: 50.3% (191/380)
2014-02-01 23:00:52,406 : INFO : gram3-comparative: 91.7% (1222/1332)
2014-02-01 23:13:48,24[illegible] : INFO : gram4-superlative: 87.[illegible]% (617/702)
2014-02-01 23:29:52,268 : INFO : gram5-present-participle: 79.4% (691/870)
2014-02-01 23:57:04,965 : INFO : gram7-past-tense: 67.1% (995/1482)
2014-02-02 00:15:18,525 : INFO : gram8-plural: 89.6% (889/992)
2014-02-02 00:28:18,140 : INFO : gram9-plural-verbs: 68.7% (482/702)
2014-02-02 00:28:18,[illegible] : INFO : total: 74.3% (5654/7614)
```

还是需要强调下，训练集上表现的好也不意味着 Word2Vec 在真实应用中就会表现的很好，还是需要因地制宜。

分享

☆ 收藏 ⚠ 纠错

推荐文章

- 1. 自然语言处理系列（2）：中文Wiki语料库词向量的训练 (/articles/bYJZrqi)
- 2. 自然语言处理系列（1）：词向量与统计语言模型 (/articles/ieaeuyq)
- 3. 谷歌发布全新端到端语音识别系统：词错率降低至5.6% (/articles/m2YJvem)
- 4. Gowild 王昊奋：知识图谱所面临的五大挑战 (/articles/BVNn2my)
- 5. 百度AAAI 2018论文提出新型NMT模型，性能堪比深层模型 (/articles/fMVVZvV)
- 6. 一文概述2017年深度学习NLP重大进展与趋势 (/articles/7FBrQ3F)

相关推刊

推酷 (https://www.tuicool.com/)

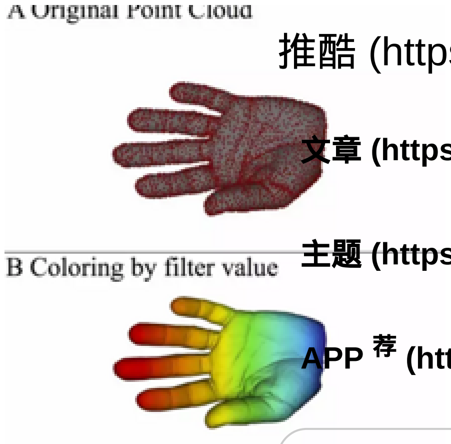
文章 (https://www.tuicool.com/ah) **站点 (https://www.tuicool.com/sites/hot)**

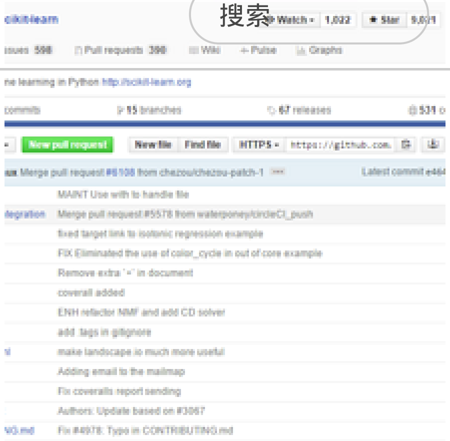
主题 (https://www.tuicool.com/topics) **活动 (https://huodong.tuicool.com/)**

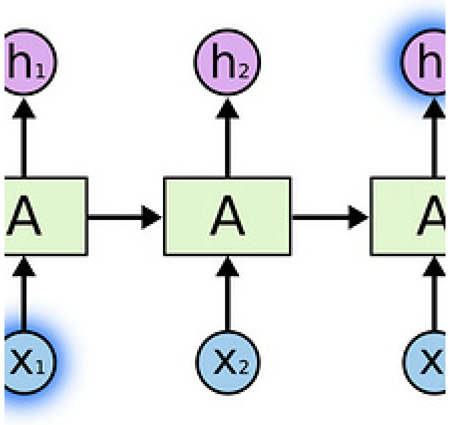
APP 荐 (https://www.tuicool.com/mobile) **周刊** **更多**

搜索

登录 (https://www.tuicool.com/login)

- 

《匿名收藏》 23
- by 晓亮 

ne learning in Python http://scikit-learn.org
commits 15 branches 67 releases 531 c
Merge pull request #6108 from chezou/chezou-patch-1 Latest commit e464
MAINT Use with to handle file
Merge pull request #5578 from waterponey/circleCI_push
fixed target link to isotonic regression example
FIX Eliminated the use of color_cycle in out of core example
Remove extra '+' in document
coverall added
ENH refactor NMF and add CD solver
add .tags in gitignore
make landscape.io much more useful
Adding email to the mailmap
Fix coveralls report sending
Authors: Update based on #3067
Fix #4978: Typo in CONTRIBUTING.md
(/kans/1584957199) 《python》 (/kans/1584957199) *291*
- by 木木的俊杰君 

(/kans/4192727130) 《算法》 (/kans/4192727130) *71*

我来评几句

请输入评论内容...

登录后评论

已发表评论数(0)

推酷 (https://www.tuicool.com/)

文章 (https://www.tuicool.com/ah) **站点 (https://www.tuicool.com/sites/hot)**

主题 (https://www.tuicool.com/topics) **活动 (https://huodong.tuicool.com/)**

APP 荐 (https://www.tuicool.com/mobile) **周刊** **更多**

搜索

登录 (https://www.tuicool.com/login)

关于我们 (https://www.tuicool.com/about) 移动应用 (https://www.tuicool.com/mobile) 意见反馈 (https://www.tuicool.com/ ol2012)