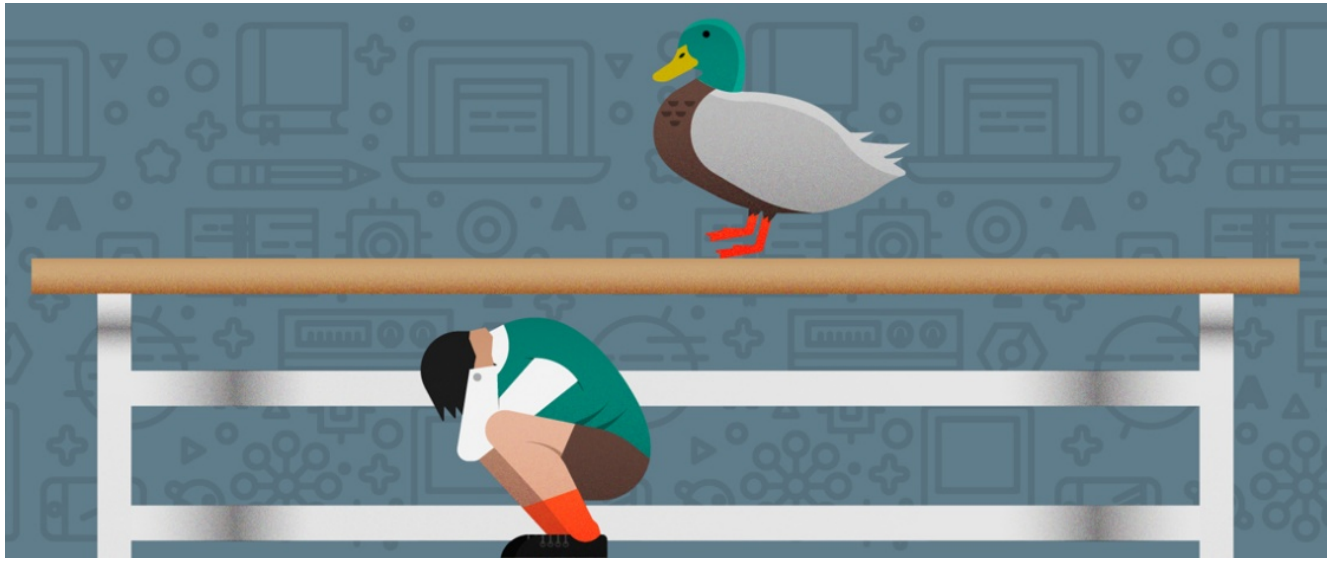


Sense2vec with spaCy and Gensim

Datartisan · 1 年前

Sense2vec with spaCy and Gensim

如果你在2015年做过文本分析项目，那么你大概率用的是word2vec模型。Sense2vec是基于**word2vec**的一个新模型，你可以利用它来获取更详细的、与上下文相关的词向量。本文主要介绍该模型的思想以及一些简单的实现。

多义性：word2vec遇到的问题

当人们编写字典和辞典时，我们会列出每个词语的不同含义。在自然语言处理过程中，利用文档的统计信息来定义词典的概念往往非常有效，其中**word2vec**系列模型是最常见的用于创建词典的模型。给定一个大规模的文本数据，**word2vec**模型将创建一个用于储存词语含义的词典，其中每行的数值代表一个词语的内在含义。此时要计算词典中两个单词之间的相似度，等价于计算这两行数据之间的相似性。

word2vec模型的问题在于词语的多义性。比如duck这个单词常见的含义有水禽或者下蹲，但对于 word2vec 模型来说，它倾向于将所有概念做归一化平滑处理，得到一个最终的表现形式。Nalisnick & Ravi注意到这个问题，他们认为模型应该考虑到词向量的多义性，这样我们可以更好地构建那些复杂的词向量。我们想要实现的功能是将不同含义的词语赋值成不同的词向量，同时我们也想知道给定上下文情况时，某个词语对应的具体含义。因此，我们需要分析上下文的内容，这正好是spaCy的用武之地。

Sense2vec: 利用 NLP 方法来构建更精确的词向量

sense2vec 模型的思想非常简单，如果要处理duck的多义性问题，我们只需要将两个不同含义的词语赋值成不同的词向量即可，即 duckNduckN 和 duckVduckV。我们一直在尝试实现这个模型，所以当Trask et al公布了其良好的模型试验结果后，我们很容易地认为这个想法是可行的。

我们跟随 Trask 等人的思路，并将部分的语音标签和名字标签纳入词向量中。此外，我们还合并了基本的名词短语和命名实体，从而获取了单一的词向量。虽然当前的模型只是个简单的草案，但是我们非常高兴可以得到这样的结果。沿着该模型的思路我们还可以做很多事情，比如处理多词问题或者单词拆解问题。

下述代码是数据预处理函数，考虑到篇幅问题，我将剩余部分的代码托管在Github。

```
def transform_texts(texts):
    # Load the annotation models
    nlp = English()
    # Stream texts through the models. We accumulate a buffer and release
    # the GIL around the parser, for efficient multi-threading.
    for doc in nlp.pipe(texts, n_threads=4):
        # Iterate over base NPs, e.g. "all their good ideas"
        for np in doc.noun_chunks:
            # Only keep adjectives and nouns, e.g. "good ideas"
            while len(np) > 1 and np[0].dep_ not in ('amod', 'compound'):
                np = np[1:]
            if len(np) > 1:
                # Merge the tokens, e.g. good_ideas
                np.merge(np.root.tag_, np.text, np.root.ent_type_)
            # Iterate over named entities
            for ent in doc.ents:
                if len(ent) > 1:
                    # Merge them into single tokens
                    ent.merge(ent.root.tag_, ent.text, ent.label_)
        token_strings = []
        for token in tokens:
            text = token.text.replace(' ', '_')
            tag = token.ent_type_ or token.pos_
            token_strings.append('%s|%s' % (text, tag))
        yield ' '.join(token_strings)
```

虽然需要这些预处理过程，但是我们仍然可以利用该模型进行大规模的建模分析。因为 spaCy 使用 Cython 写的，它允许多线程操作，在四线程环境中该模型每秒可以处理 100,000 个单

词。

数据预处理之后，我们可以利用常规的方法来训练词向量，比如原始的 C 语言代码、Gensim 或者 GloVe。只要数据集中单词由空格分隔，且句子由换行符分隔开就没有问题。唯一需要注意的地方是该模型不应该试图利用其自身的符号，否则可能会错误地拆分标签信息。

我们利用 Gensim 中的基于负抽样方法的 Skip-Gram 模型来训练词向量，其中频数阀值为10或 5。模型训练后我们将频数阀值设为50，从而减少模型的运算时间。

案例

当我们利用这些词向量来分析问题时，我们发现了许多有趣的事情，以下是一些简单的说明：

语义合成性

该模型训练出来的词向量可以很好地提取合成词的语义信息，比如该模型知道 fair game 不是一个游戏类型，而 multiplayer game 是一种游戏类型。

```
>>> model.similarity('fair_game|NOUN', 'game|NOUN')
0.034977455677555599
>>> model.similarity('multiplayer_game|NOUN', 'game|NOUN')
0.54464530644393849
```

同样地，该模型知道 class action 和 action 之间的相似度很低，而 class action lawsuit 和 lawsuit 之间有很高的相似度：

```
>>> model.similarity('class_action|NOUN', 'action|NOUN')
0.14957825452335169
>>> model.similarity('class_action_lawsuit|NOUN', 'lawsuit|NOUN')
0.69595765453644187
```

词语之间的相似性

以下是 Reddit 网上关于川普的词向量信息：

```
>>> model.most_similar(['Donald_Trump|PERSON'])
(u'Sarah_Palin|PERSON', 0.854670465),
(u'Mitt_Romney|PERSON', 0.8245523572),
(u'Barrack_Obama|PERSON', 0.808201313),
(u'Bill_Clinton|PERSON', 0.8045649529),
(u'Oprah|GPE', 0.8042222261),
(u'Paris_Hilton|ORG', 0.7962667942),
(u'Oprah_Winfrey|PERSON', 0.7941152453),
(u'Stephen_Colbert|PERSON', 0.7926792502),
(u'Herman_Cain|PERSON', 0.7869615555),
(u'Michael_Moore|PERSON', 0.7835546732)]
```

该模型返回了与‘川普’之间相似度较高的词语，从上述结果中可以看出该模型很好地识别出川普政治家和真人秀明星的身份。我对模型返回的 Michael Moore 非常感兴趣，我怀疑很多人都是他两的粉丝。如果我必须选择出一个异常值的话，那么我会选择 Oprah，该词条和其他词语的相似度较低。

该模型发现 Oprah|GPE 和 Oprah_Winfrey|PERSON 之间的相似度较高，这意味着命名实体识别器还存在一定的问题，具有提升的空间。

word2vec模型可以很好地识别出命名实体，特别是音乐领域的信息。这让我想起我曾经获取推荐音乐的方式：留意经常和我喜欢的乐队一起被提到的歌手。当然现在我们已经拥有更强大的推荐模型，比如观察成千上万人的行为进而得出相应的规律。但是对我来说，该模型在分析乐队相似度时仍存在一些奇怪的问题。

以下是该模型揭示的 Carrot Top 和 Kate Mara 之间潜在的联系：

```
>>> model.most_similar(['Carrot_Top|PERSON'])
[(u'Kate_Mara|PERSON', 0.5347248911857605),
 (u'Andy_Samberg|PERSON', 0.5336876511573792),
 (u'Ryan_Gosling|PERSON', 0.5287898182868958),
 (u'Emma_Stone|PERSON', 0.5243821740150452),
 (u'Charlie_Sheen|PERSON', 0.5209298133850098),
 (u'Joseph_Gordon_Levitt|PERSON', 0.5196050405502319),
 (u'Jonah_Hill|PERSON', 0.5151286125183105),
 (u'Zooey_Deschanel|PERSON', 0.514430582523346),
 (u'Gerard_Butler|PERSON', 0.5115377902984619),
 (u'Ellen_Page|PERSON', 0.5094753503799438)]
```

我花了好多时间在思考这个问题，但是并没有得到任何有意义的结果。也许这里面存在更深层次的逻辑关系，我们需要进一步探究才能得到结果。但是当我们往模型中加入更多的数据时，该现象就消失了，就和 Carrot Top 一样。

食品领域

```
>>> model.most_similar(['onion_rings|NOUN'])
[(u'hashbrowns|NOUN', 0.8040812611579895),
 (u'hot_dogs|NOUN', 0.7978234887123108),
 (u'chicken_wings|NOUN', 0.793393611907959),
 (u'sandwiches|NOUN', 0.7903584241867065),
 (u'fries|NOUN', 0.7885469198226929),
 (u'tater_tots|NOUN', 0.7821801900863647),
 (u'bagels|NOUN', 0.7788236141204834),
 (u'chicken_nuggets|NOUN', 0.7787706255912781),
 (u'coleslaw|NOUN', 0.7771176099777222),
 (u'nachos|NOUN', 0.7755396366119385)]
```

Reddit 网站上关于食品的一些评论非常有趣，比如 bacon 和 brocoll 之间的相似度非常高：

```
>>> model.similarity('bacon|NOUN', 'broccoli|NOUN')
0.83276615202851845
```

此外，模型的结果显示热狗和沙拉之间也非常相似：

```
>>> model.similarity('hot_dogs|NOUN', 'salad|NOUN')
0.76765100035460465
>>> model.similarity('hot_dogs|NOUN', 'entrails|NOUN')
0.28360725445449464
```

Using the demo

你可以通过搜索单词或短语来探索相关概念。如果你想要更精确的信息，你可以在查询语句中加入标签信息，比如query phrase|NOUN。如果你没有添加标签信息，那么该模型将会返回关

联度最高的单词。标签信息主要由包含了上下文信息的统计模型预测所得。

如果你输入 serve，该模型将从 serve|VERB, serve|NOUN, serve|ADJ 等标签信息中搜寻相关单词。由于 serve|VERB是最常见的标签信息，该模型将返回这个结果。但是如果你输入 serve|NOUN，你将得到完全不一样的结果，因为 serve|NOUN 和网球之间的关系非常紧密，而动词形式则表示其他含义。

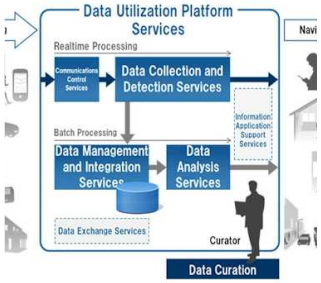
我们采用了基于频率的方法来区分大小写的情况。如果你的查询命令是小写单词且没有标签信息，我们将假设它是不区分大小写的，同时寻找最常见的标签和单词。如果你的查询命令中包含大写字母或者标签信息，我们将假设你的查询命令是区分大小写的。

原文链接：Sense2vec with spaCy and Gensim

原文作者：MATTHEW HONNIBAL

译者：Fibears

大数据

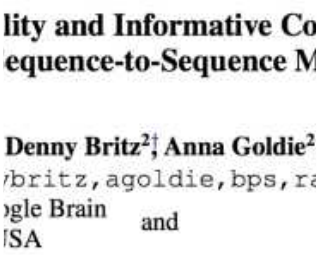


Datartisan · 1 年前 · 发表于 Datartisan数据工匠

看完了这期《奇葩说》，你觉得爱上人工智能算不算爱情丨数据工匠（6.13）

看完了这期《奇葩说》，你觉得爱上人工智能算不算爱情丨数据工匠（6.13）看完了这期《奇葩说... 查看全文 ›

Datartisan · 2 年前 · 发表于 Datartisan数据工匠


lity and Informative Co
equence-to-Sequence M

Denny Britz[2], Anna Goldie[2]
/britz,agoldie,bps,ra
ogle Brain and
/SA

《Generating High-Quality and Informative Conversation Responses with Seq2Seq Models》阅读笔记

标题：《Generating High-Quality and Informative Conversation Responses with Sequence-t... 查看全文 ›

不读不读不读 · 2 个月前

Neural Headline Generation with Minimum Risk Training

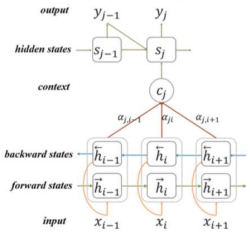


Figure 1: The framework of neural headline generation.

今天分享的paper是Neural Headline Generation with Minimum Risk Training。本文通过将评价… 查看全文 >

张俊 · 1 年前 · 发表于 PaperWeekly