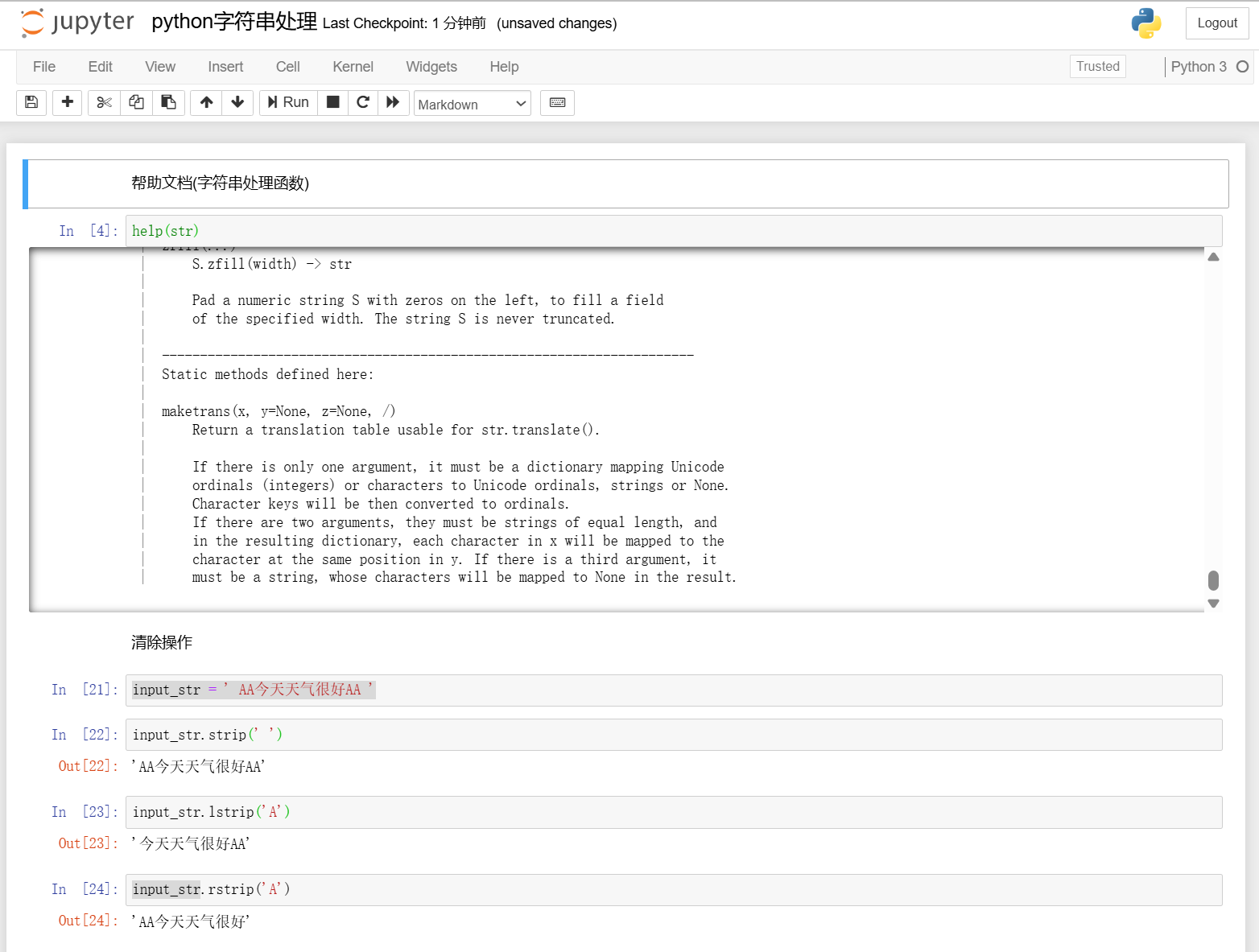
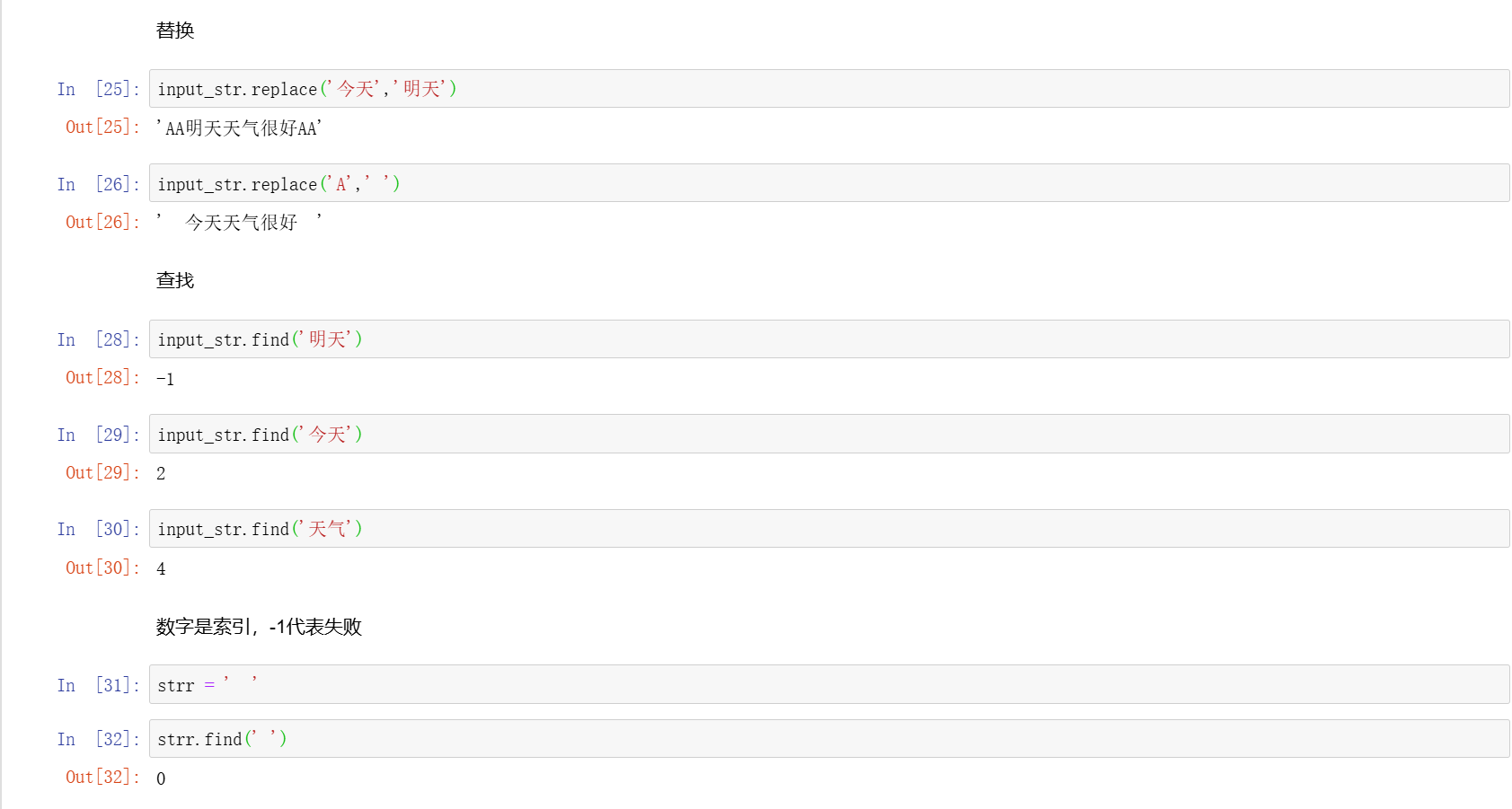
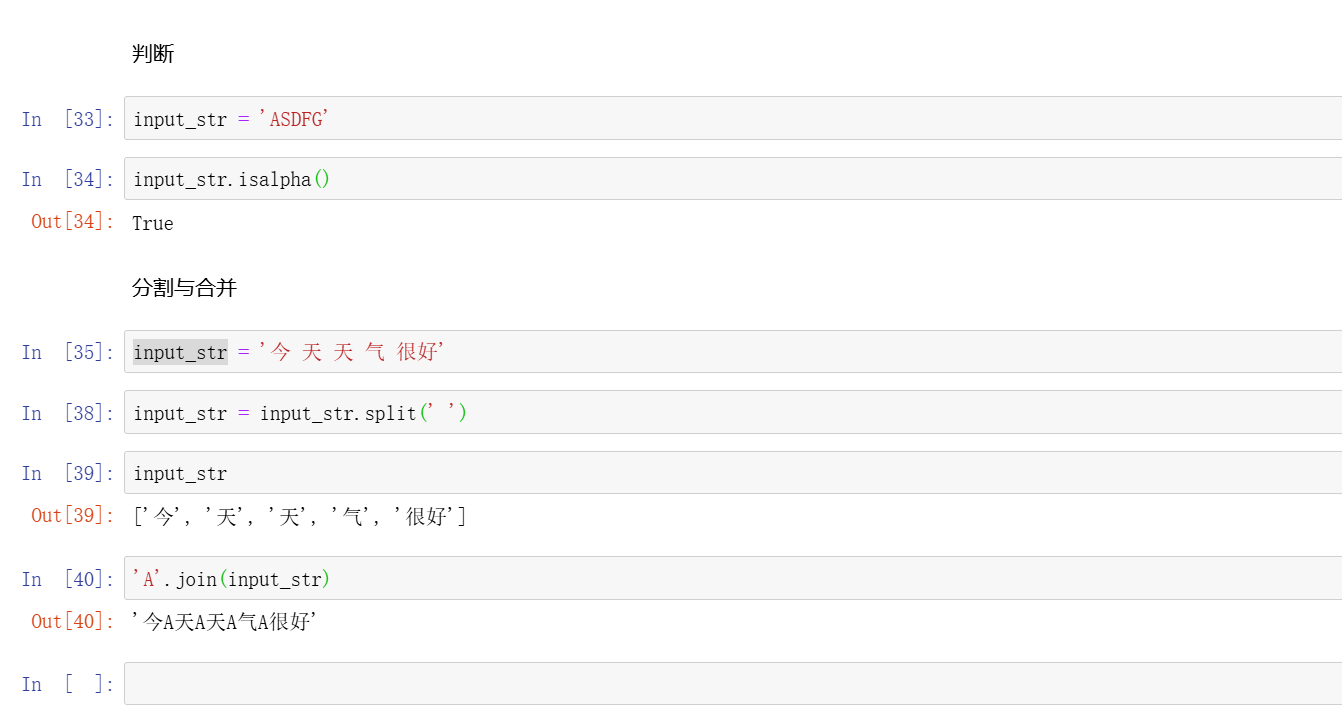
# 1.Python字符串处理

用jupyter实操来演示：







# 正则表达式基本语法、常用符号以及常用函数

在Python中，正则表达式是一种强大的工具，可用于匹配和操作字符串。什么是正则表达式？ 正则表达式是一种模式匹配语言，用于匹配字符串中的特定模式。这些模式可以是字母、数字、字符组合或其他符号。正则表达式通常用于文本处理、网络编程、数据分析等领域。

  在 Python 中，正则表达式的实现主要基于 re 模块，该模块提供了一组函数和类，用于处理正则表达式的匹配、搜索和替换。

正则表达式引擎： Python 中的正则表达式引擎使用了正则表达式的编译和匹配两个主要阶段。

  编译阶段： 在编译阶段，正则表达式字符串会被解析并转换成一个内部的模式表示。这个模式表示了匹配规则，它包括普通字符、元字符、字符类、分组等。

  匹配阶段： 一旦正则表达式被编译，就可以用来匹配字符串。匹配过程是基于模式在字符串中的搜索和比对。引擎会从字符串的起始位置开始，尝试找到与模式匹配的子字符串。

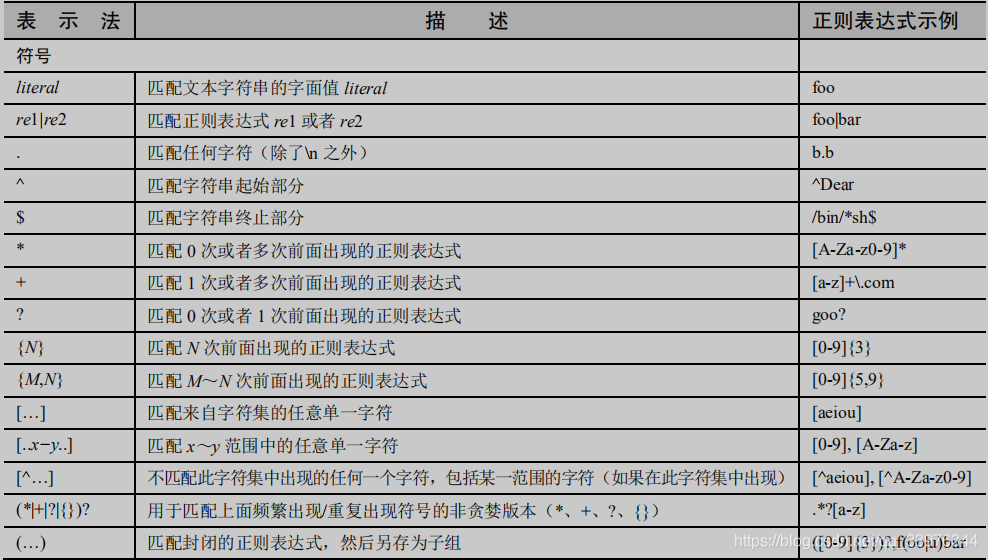
一个编译与匹配的演示:

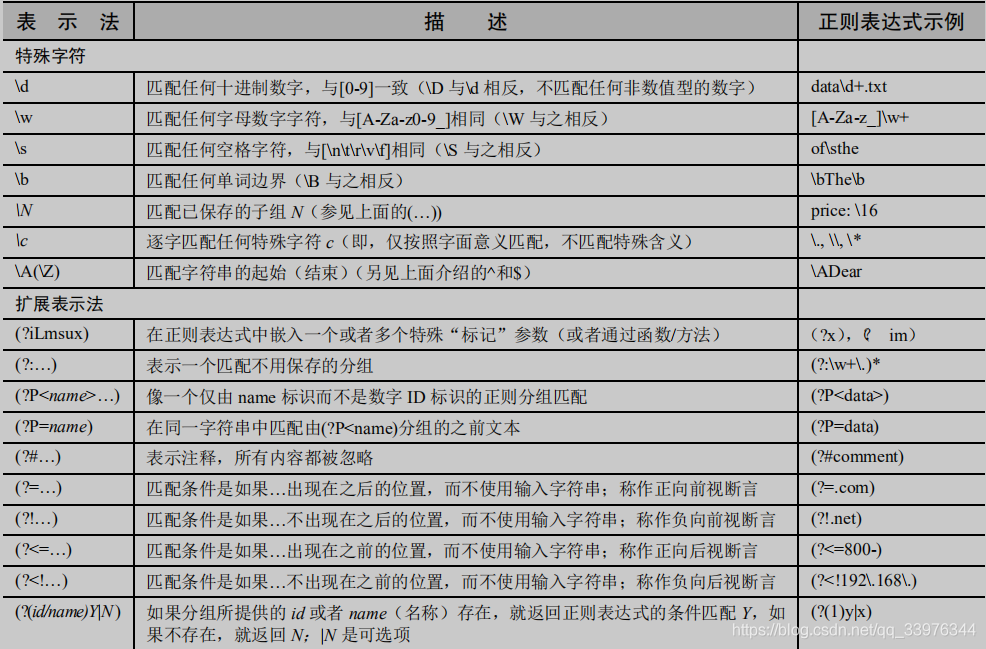
使用re.compile()对正则表达式提前进行编译，可以提高数据处理效率，注意要将经过re.compile()生成的对象，写在程序开端。



  回溯和优化： 在匹配过程中，可能会涉及到回溯的操作，即引擎试图在字符串中不同位置匹配模式。为了提高性能，引擎会使用一些优化策略，避免不必要的回溯。

下面给出正则表达式的常用语法表：





操作示例，用jupyter notebook打开



# NTLK工具包

非常实用的文本处理工具，主要用于英文数据，历史悠久

具体的操作过程文档已经添加注释



# Spacy工具包

Spacy，[NLTK](https://so.csdn.net/so/search?q=NLTK&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_41862755/article/details/_blank)都是自然语言处理（NLP）库。这些库都可以处理中文和英文文本。下面是它们各自的优缺点：

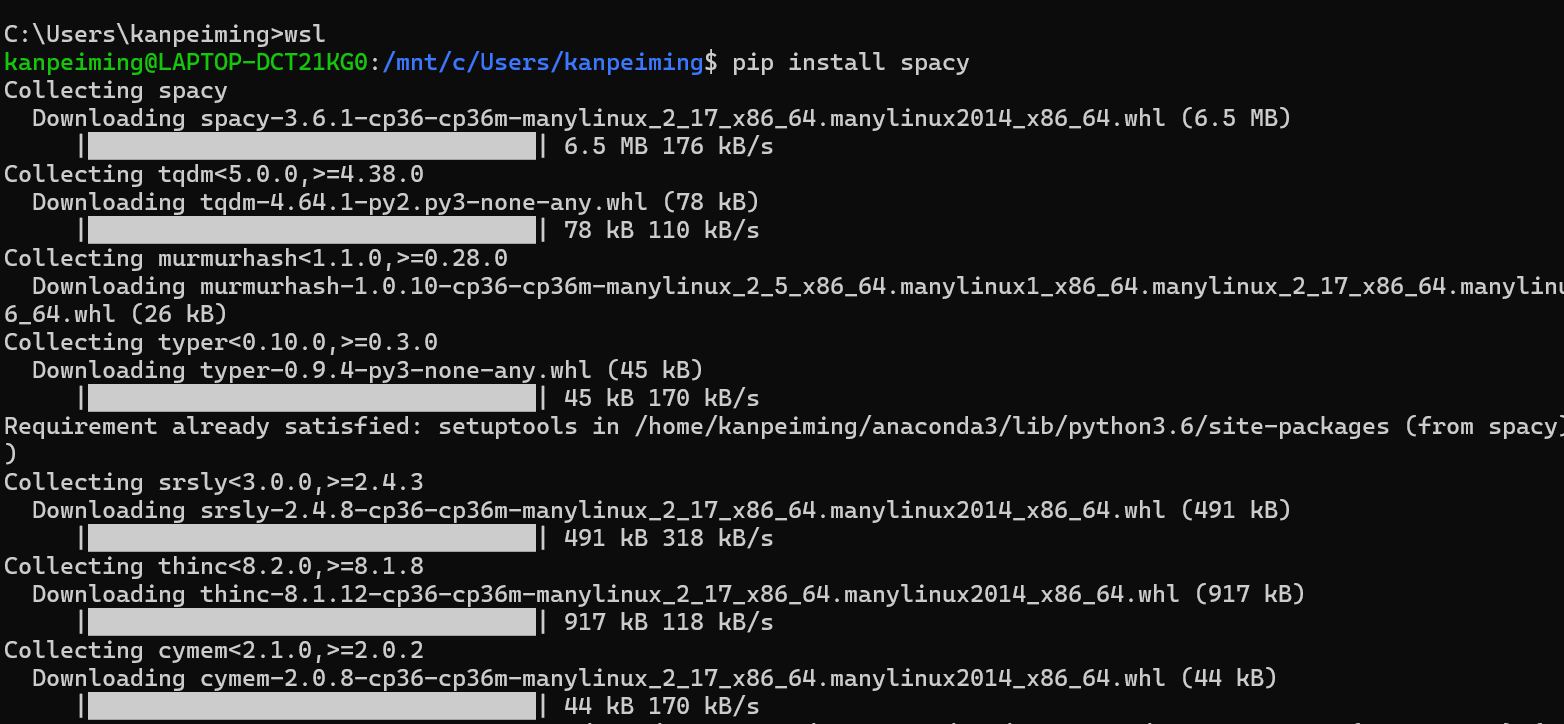
Spacy：速度快，内存占用少，易于使用，支持多语言，但是对于一些特殊任务可能不太适用。

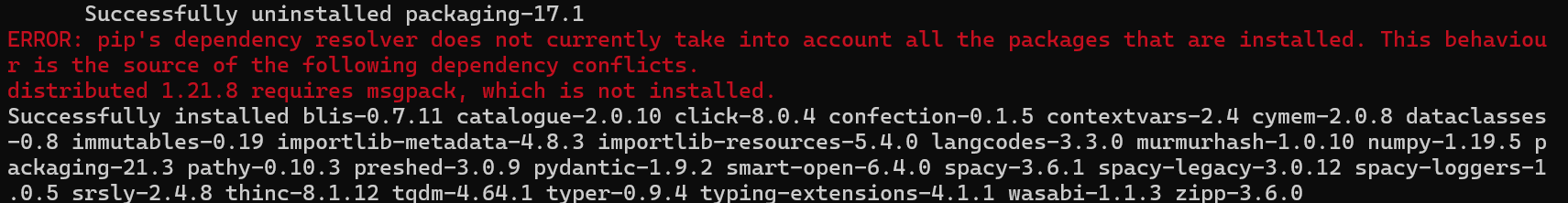
NLTK：功能强大，支持多种语言，有大量的文档和教程，但是速度较慢。

英文文本

1. 首先，需要安装Spacy, 你可以使用以下命令在终端中安装Spacy：

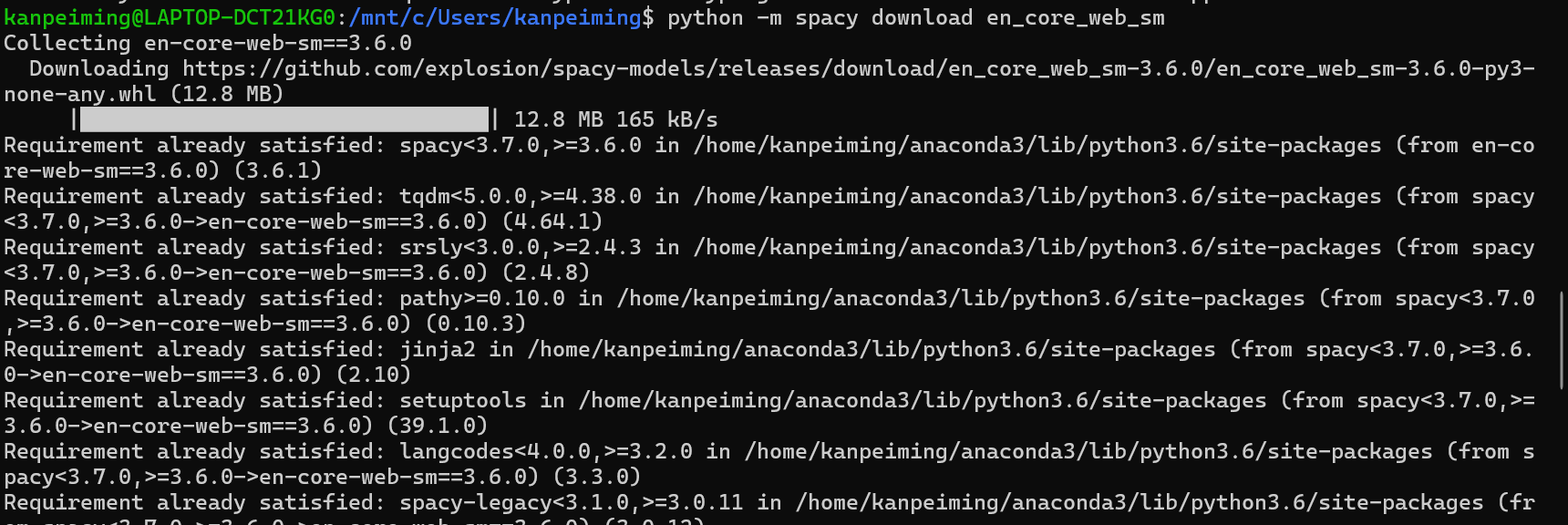
pip install spacy

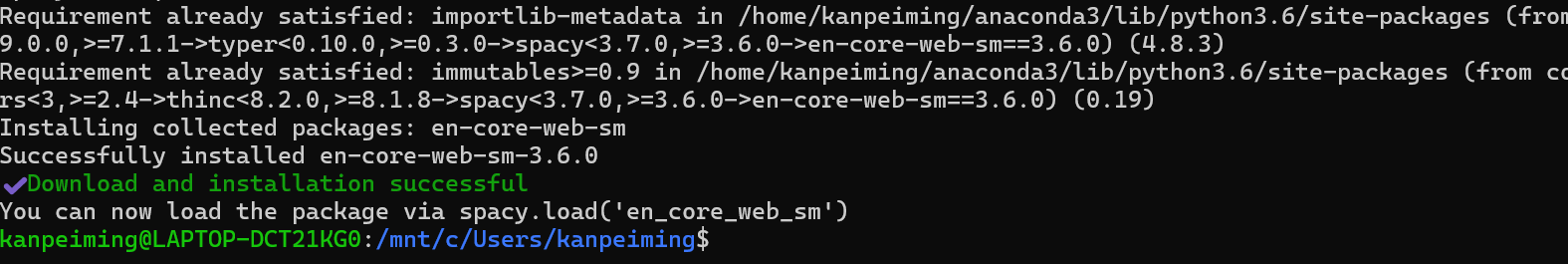




2. 安装完成后，你需要下载一个预训练的模型。你可以使用以下命令下载英文模型：

python -m spacy download en\_core\_web\_sm





导入工具包和英文模型后即可使用

操作示例，注释请见文档：



# 结巴分词器

结巴分词是当前效果较好的一种中文分词器，支持中文简体、中文繁体分词，同时还支持自定义词库。

pip install jieba

结巴分词支持三种分词模式：精确模式、全模式和搜索引擎模式。

精确模式是试图将句子最精确的进行切分，适合用于文本分析；

全模式的原理是把句子中全部可以成词的词语全部扫描出来，它的分词速度快，缺点是无法识别歧义词句；

搜索引擎模式是在精确模式的基础上进一步处理的，它对较长的词语再进行分割，将召回率提高，适合于搜索引擎分词。

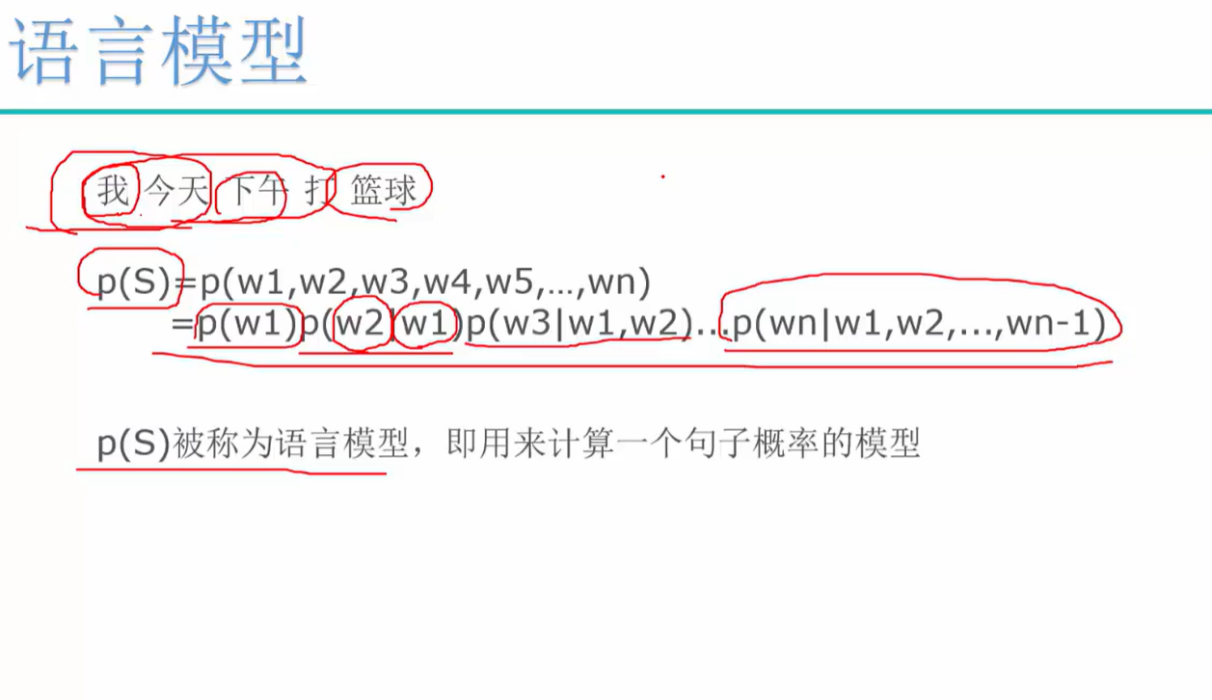
结巴分词自带一个词典，内含20000多条词，及词条出现的次数与词性。结巴分词用到的算法有基于Trie树的词图遍历，会生成一个DAG图，该图是由句子中所有汉字的所有可能组合决定，采用动态规划（DP）查找概率最大的路径，从而找出基于词频的最大切分组合，而对于结巴分词词典中未记录的词，其使用了基于汉字成词能力的HMM模型，以及Viterbi算法。

操作实例：

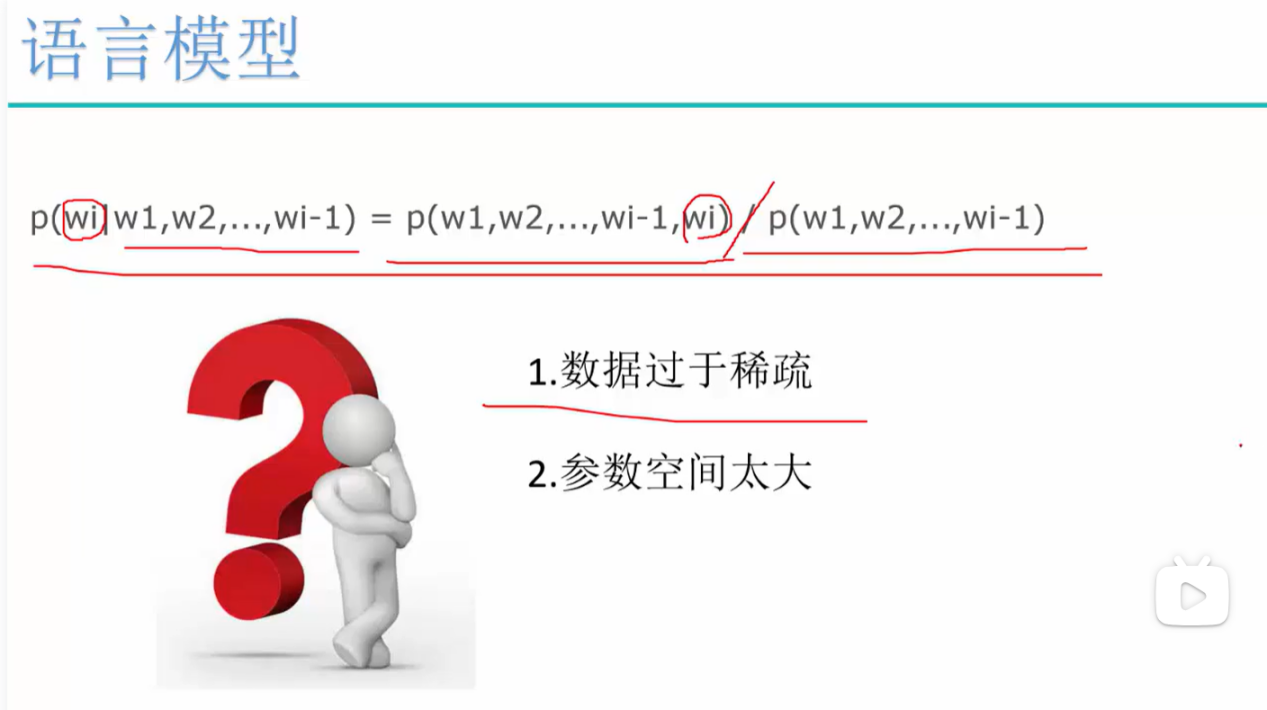


# 语言模型

## （1）简介



语言模型是一个基于概率的判别模型，它的输入是一句话（单词的顺序序列），输出是这句话出现的概率，即这些单词的联合概率。每个词语的出现跟前面的词有关，这种简单的模型有什么问题？



对于这两个缺点请参考：

[详解语言模型NGram及困惑度Perplexity-CSDN博客](https://blog.csdn.net/qq_27825451/article/details/102457058?ops_request_misc=&request_id=&biz_id=102&utm_term=%E8%AF%AD%E8%A8%80%E6%A8%A1%E5%9E%8B%E6%95%B0%E6%8D%AE%E7%A8%80%E7%96%8F%E6%98%AF%E4%BB%80%E4%B9%88%E6%84%8F%E6%80%9D&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-4-102457058.142^v100^pc_search_result_base7&spm=1018.2226.3001.4187)

## **N-gram模型**

1. Gram（N元模型）是自然语言处理中一个非常重要的概念。N-gram模型也是一种语言模型，是一种生成式模型。

假定文本中的每个词 和前面 N-1 个词有关，而与更前面的词无关。这种假设被称为N-1阶马尔可夫假设，对应的语言模型称为N元模型。习惯上，1-gram叫unigram，2-gram称为bigram（也被称为一阶马尔可夫链），3-gram是trigram（也被称为二阶马尔可夫链）。还有four-gram、five-gram等，不过大于n>5的应用很少见。常用的是 Bi-gram (N = 2) 和 Tri-gram (N = 3)，一般已经够用了。



M为语料库词汇量

## **词向量**

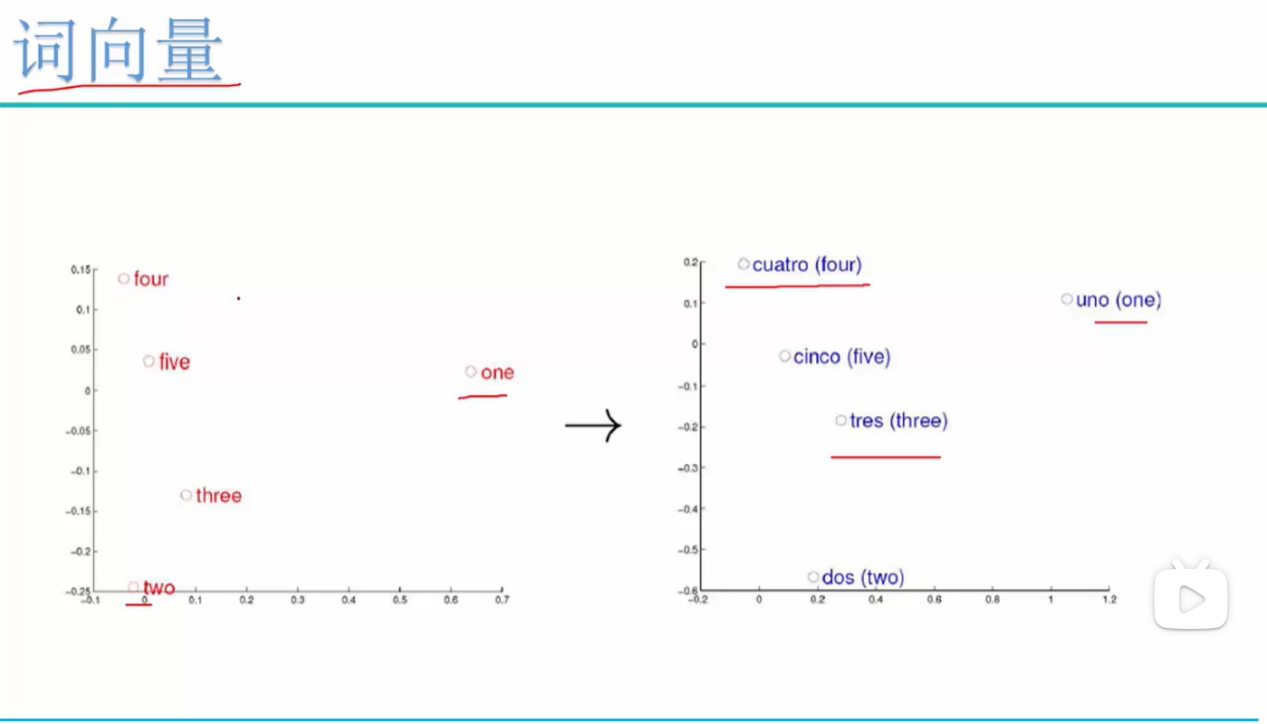
单词-->向量（有意义） 用Word2Vec工具实现

[一文读懂：词向量 Word2Vec - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/371147732)

例如在语言空间上，相近含义的词语距离较近，用向量将这种关系表示出来，那么向量就有了意义。

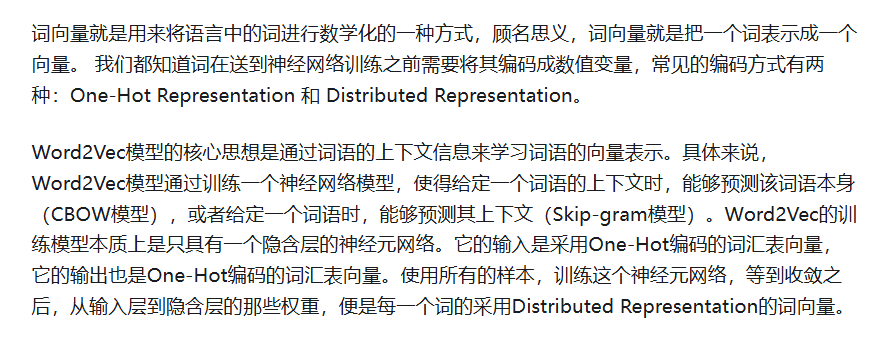


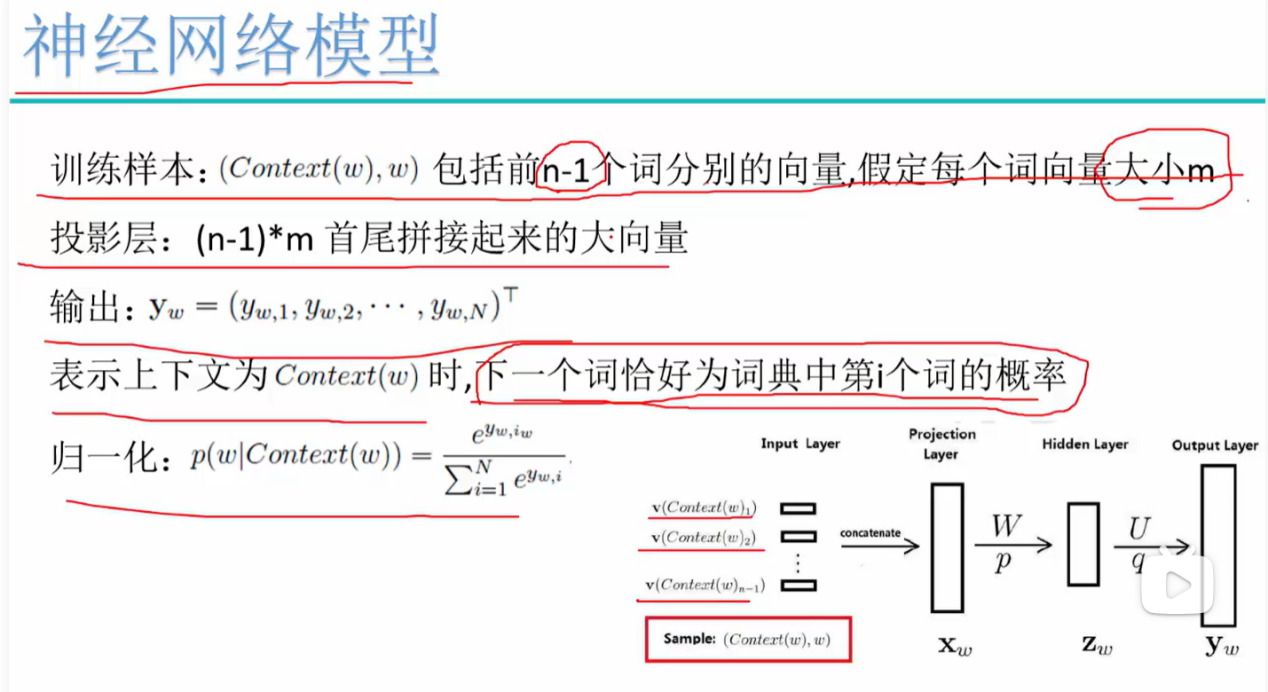
词向量的意义更关注于整体的上下文逻辑，而不是词语本身。例如，以下展示的语言模型，语言虽然不同，但是模型分布类似。



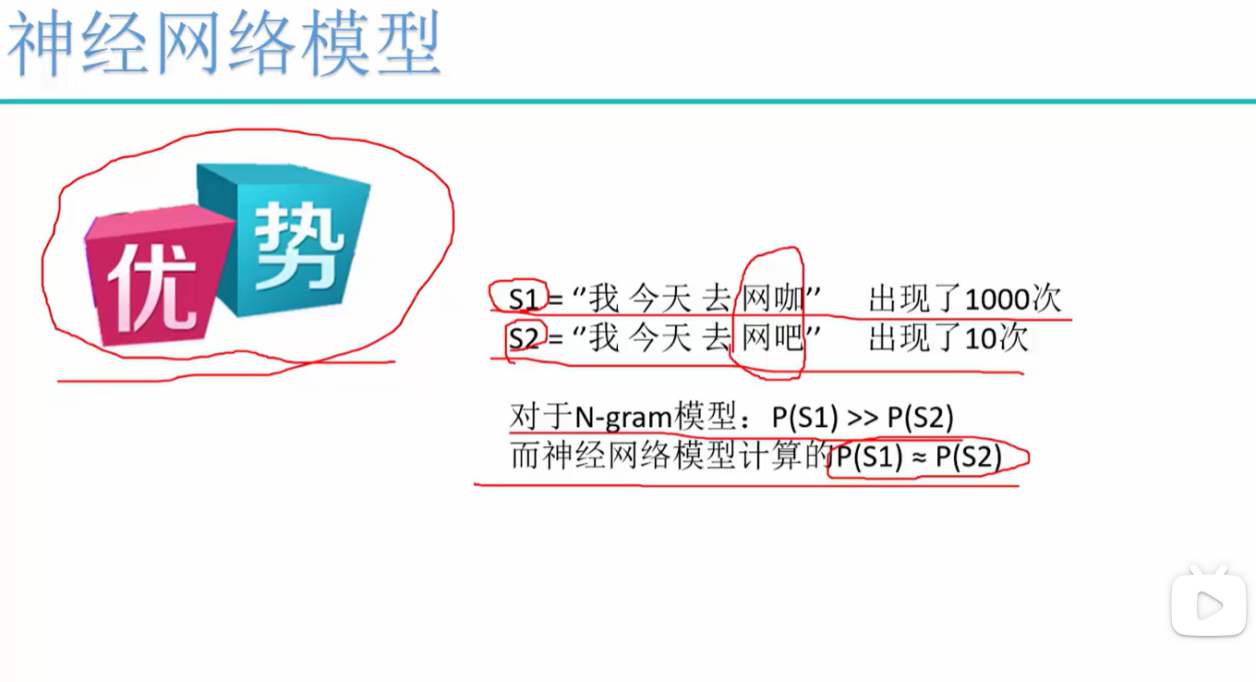
## **神经网络模型**

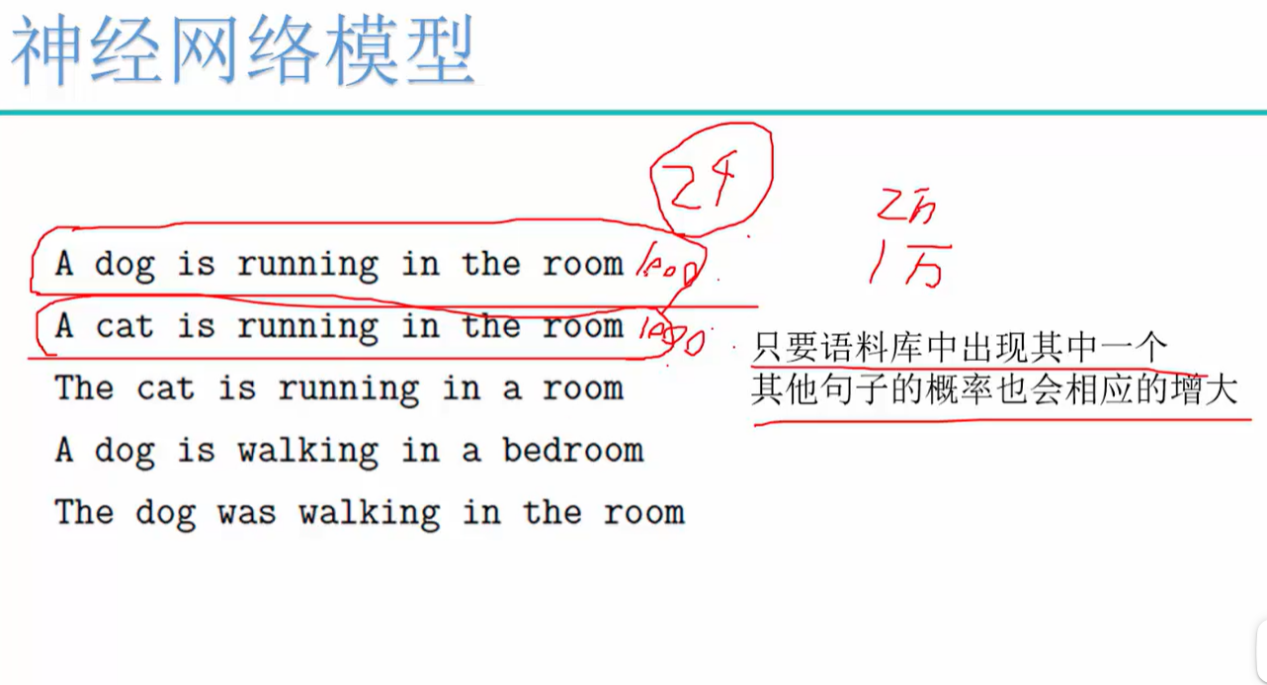
基于神经网络模型求解每个词语的词向量



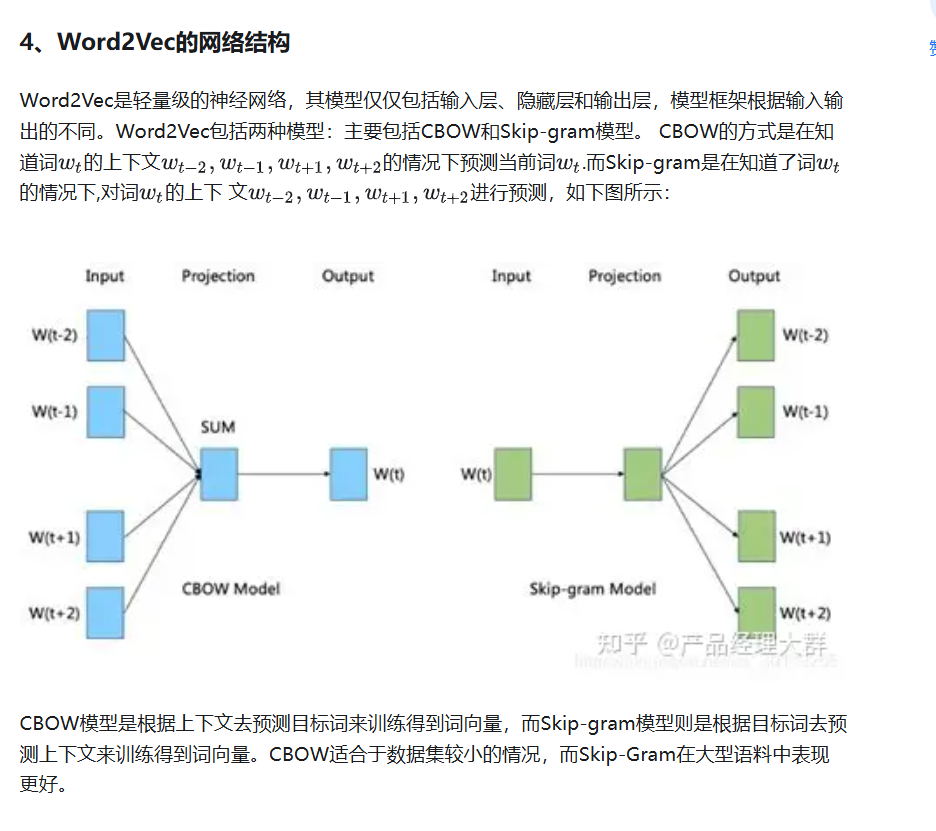


神经网络模型的优势：N-gram模型的预测与词语出现的频率有关，即使意思相近，出现的概率也会有较大差别，而通过神经网络，词义相近的两个词向量会非常相似，因此预测的概率也几乎相等。

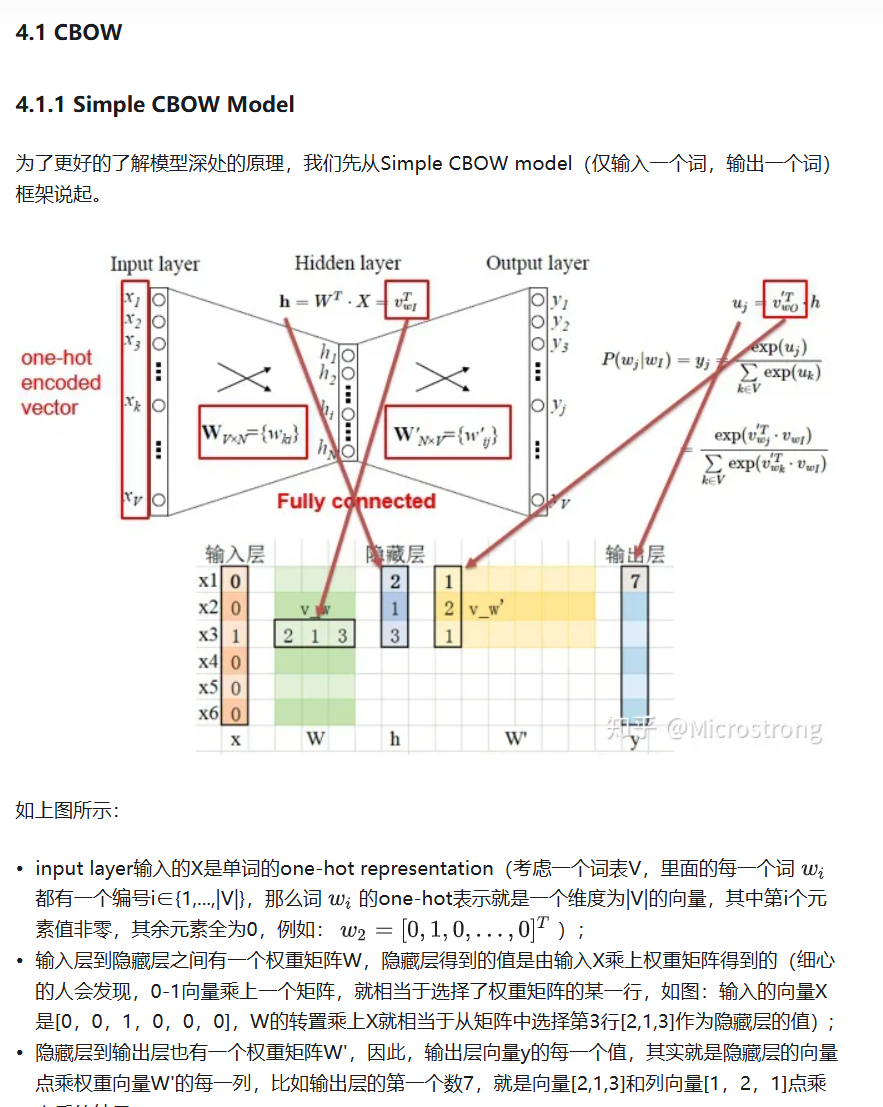


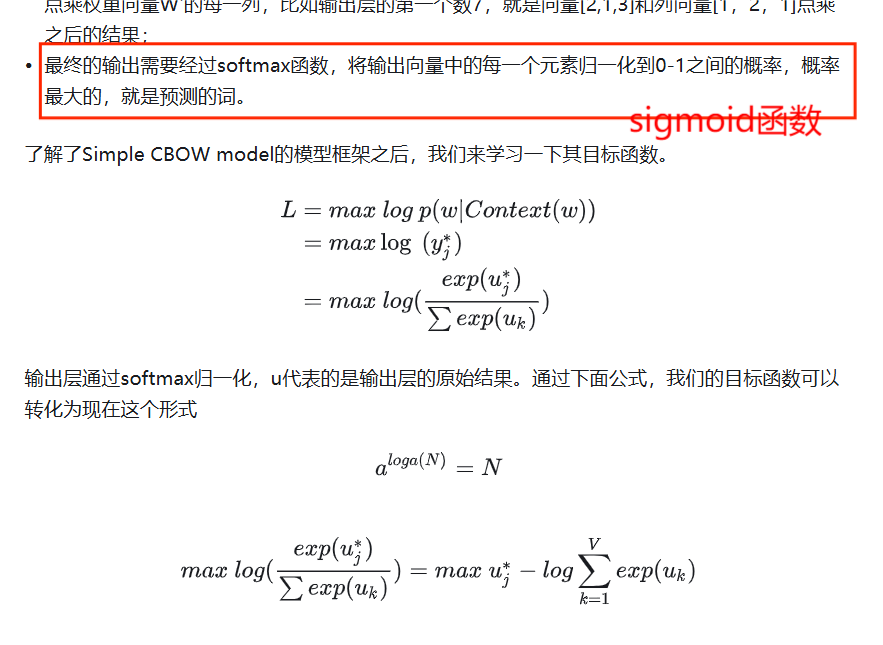


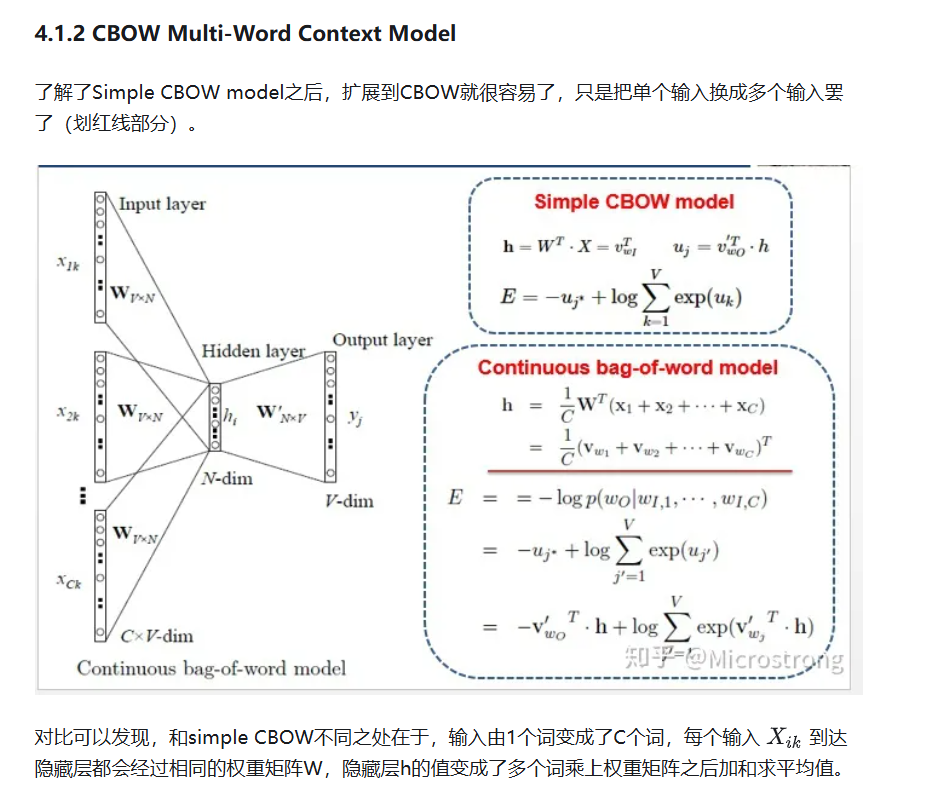
## **Word2Vec的网络结构**



## **简介CBOW**







# 自然语言处理的不同模型

# **词袋模型**

[传统NLP之Bag of Words（词袋模型）](https://zhuanlan.zhihu.com/p/667280452)

1.词袋模型

文本[数据预处理](https://so.csdn.net/so/search?q=%E6%95%B0%E6%8D%AE%E9%A2%84%E5%A4%84%E7%90%86&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/m0_37870649/article/details/_blank)的第一步通常是进行分词，分词后会进行向量化的操作。

词袋模型假设我们不考虑文本中词与词之间的上下文关系，仅仅只考虑所有词的权重。而权重与词在文本中出现的频率有关。

词袋模型首先会进行分词，在分词之后，通过统计每个词在文本中出现的次数，我们就可以得到该文本基于词的特征，如果将各个文本样本的这些词与对应的词频放在一起，就是我们常说的向量化。向量化完毕后一般也会使用 TF-IDF 进行特征的权重修正，再将特征进行标准化。 再进行一些其他的特征工程后，就可以将数据带入机器学习模型中计算。

词袋模型的三部曲：分词（tokenizing），统计修订词特征值（counting）与标准化（normalizing）。

词袋模型有很大的局限性，因为它仅仅考虑了词频，没有考虑上下文的关系，因此会丢失一部分文本的语义。

在词袋模型统计词频的时候，可以使用 sklearn 中的 CountVectorizer 来完成。

2.词频向量化

CountVectorizer 类会将文本中的词语转换为词频矩阵，例如矩阵中包含一个元素a[i][j]，它表示j词在i类文本下的词频。它通过 fit\_transform 函数计算各个词语出现的次数，通过get\_feature\_names()可获取词袋中所有文本的关键字，通过 toarray()可看到词频矩阵的结果。

# **TF-IDF处理**

然而有些词在文本中尽管词频高，但是并不重要，这个时候就可以用TF-IDF技术。

TF-IDF（Term Frequency–Inverse Document Frequency）是一种用于资讯检索与文本挖掘的常用加权技术。TF-IDF是一种统计方法，用以评估一个字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF-IDF加权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。

TF-IDF的主要思想是：如果某个词或短语在一篇文章中出现的频率TF高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。TF-IDF实际上是：TF \* IDF。

TF：Term Frequency，即词频。它表示一个词在内容(如某文章)中出现的次数。为了消除文档本身大小的影响，通常，它的定义是：

TF ＝ 某个词在文档中出现的次数 ／ 文档的总词数

某个词出现越多，表示它约重要。比如某篇新闻中，“剑术”出现了5次，“电视”出现了1次，很可能这是一个剑术赛事报道。

如果这篇新闻中，“中国”和“剑术”出现的次数一样多，是不是表示两者同等重要呢？答案是否定的，因为中国这个词很常见，它难以表达文档的特性。而剑术很少见，更能表达文章的特性。





请参考：

[自然语言处理之文本向量化（词袋模型、TF-IDF）-CSDN博客](https://blog.csdn.net/m0_37870649/article/details/79475387?ops_request_misc=&request_id=&biz_id=102&utm_term=%E8%AF%8D%E8%A2%8B%E6%A8%A1%E5%9E%8B&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-4-79475387.142^v100^pc_search_result_base7&spm=1018.2226.3001.4187)