词表达考古史

提出问题

通用人工智能（AGI）的终极目标就是让计算机像人一样思考和说话。即让计算机能理解人类的语言，并且用人类的语言和人交流，和帮助人类处理各种事务。因此，我们提出了要解决的问题：

> 首当其冲要解决的问题，就是让计算机理解人类的语言

词表达（Word Presentation）就是指将自然语言中的词转换为机器可理解的形式或含义的过程。

解决问题

人类的语言

人类的语言有很多种，我们这里只讨论能表示成文字的那些语言。大致分为两个大类：**表音**和**表意**两类。
我们以英语（表音文字）和汉语（表意文字）为例，

- 英语的文本表示：字母 --> 单词 --> （词组） --> 句子
- 汉语的文本表示：字 --> （词语）--> 句子

文本的划分单元从小到大依次为单词、短语、句子、段落和篇章。其中，句子的表示法依赖于单词，而段落又依赖于句子，以此类推，可以说英文文本的表示是建立在单词的表示基础之上的。所以**单词的表示法**最为基础，也最为重要。相应的，中文中最重要的就是**汉字的表示法**。

计算机能处理的语言

众所周知，计算机只能处理0和1，即高低电平（两种状态）。所有的文本信息必须转化成数值型数据，才能被计算机处理。对于AI模型来说，非数值型的文本数据不能直接输入机器学习模型，要先经过编码转化成数值型数据才可用于模型训练或预测。而文本表示，就是研究如何将文本数据合理编码成数值型数据的技术。

计算机是如何表示文本

编码表

理科研究，有个共性的办法。找到最小颗粒度的问题，然后研究透彻，然后在组合出复杂的问题。比如物理学，需要研究问题基本粒子的问题。比如生物学，我们需要研究基因。
计算机也不例外，以英文为例，我们先找到最小的单元：**字母**； 然后，把这个转化为0和1。这个过程，专业术语叫**编码**。
鼎鼎大名的ASCII Table就是把字母，数字和特殊字符合在一起的编码表。比如：序号为97的字符a，对应的编码就是01100001。相应的，中文里也有类似的情况，比如GB2312编码表。随着计算机技术的普及，人们就有了统一各种语言的需求，于是乎，大家把各种主流语言的编码表做了一个统一，诞生了UTF-8这种字符表。

Tips:大家学习一下这种典型的计算机邻域解决问题的思路，当每个解决方案只能解决局部问题的时候，我们往往在这些解决方案上面加一层，用统一的办法来解决一类问题。软件领域，有个设计模式叫Adapter，就是这个思想。多层的网络协议栈，也是这种思路。

词的表示方法

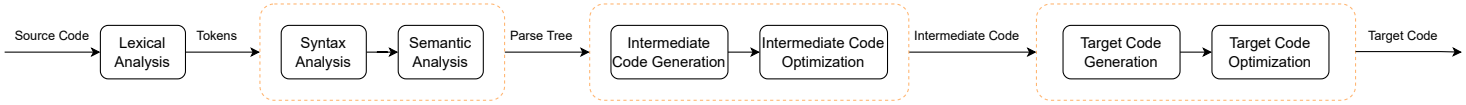
在自然语言中，文章是由段落构成，段落由句子组成，而句子由单词构成。然而，每个单词都有多种含义，因此在一片文章里，只看单词很难确定其所表达的含义。只有当单词放到句子中，结合上下文，才能确定单词所表达的含义，进而组合成句子，从而表达句子的语义。
基于这个原因，词表达不仅仅要把词本身用0，1表示出来，还需要结合上下文，精准的表示出其中的语义。
主流的词表达分为两类:

1. 词袋模型(Bag-of-words Model)
2. 词向量法(Word Embedding)

2024年，我们都采用就是词向量法。本文的重心也将放在介绍词向量法。现在，还有一个问题亟待解决。这个问题就是词(word)本身的定义。以中英文为例，如何让计算机统一处理英语单词（Word）和汉字?

分词（Tokenizer）

如何把各种人类的语言变成统一格式，交给AI处理呢？我们先在计算机范畴内，找找看有没有类似的问题。
有了，计算机需要处理很多种编程语言，如C/Java/Go/Javascript等。和我们NLP里，遇到这个问题类似。我们一起来重温一下编译原理的处理过程。



从上图可以看出，编译原理的第一步，词法分析就做了这个事情，把编程语言转化成Token，交给编译器处理。这个过程就叫分词（Tokenizer）。
同样的，在NLP种，我们也参考编译原理的处理方法。把各种人类的语言，转化为Token。交给AI来统一处理。

Tips: Token没有特别的中文翻译，后续文章提到词，指的就是统一格式的Token。

词袋模型(BoW:Bag-of-Words)表示

词袋模型是一种在自然语言处理（NLP）和信息检索（IR）中广泛使用的简化表达模型。举个例子，假如现在有1,000篇新闻文档，把这些文档拆成一个个的字，去重后得到3,000个字，然后把这3,000个字作为字典，进行文本表示的模型，叫做词袋模型。这种模型的特点是字典中的字没有特定的顺序，句子的总体结构也被舍弃了。在本章节里，我们将介绍三种最常用的，分别是one-hot（独热码），TF-IDF和n-gram。

这里，我们用一个简单的例子来讲解这三种方式。一个文本有两句话：

- 第一句：**小美女太可爱了，我爱小美女。**
- 第二句：**我要看小美女的演唱会。**

One-hot

针对上面提到的例子，One-hot是这样做的。

1. 先把这两句话拆成一个个字，一共有22个字。

小	美	女	太	可	爱	了	我	爱	小	美	女	我	要	看	小	美	女	的	演	唱	会

2. 去重，整理得到不重复的字，一共有14个字。

小	美	女	太	可	爱	了	我	要	看	的	演	唱	会

3. 统计第一句话中每个字在字典中出现的位置

小	美	女	太	可	爱	了	我	要	看	的	演	唱	会
1	1	1	1	1	1	1	1	0	0	0	0	0	0

4. 统计第二句话中每个字在字典中出现的位置，得到词汇频率表(Vocabulary Frequency Table)

小	美	女	太	可	爱	了	我	要	看	的	演	唱	会
1	1	1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	1	1	1	1	1	1	1

5. 向量化表示这两句话，得到这个文本（含两句话）的独热码。

- 第一句：[1,1,1,1,1,1,1,1,0,0,0,0,0,0]
- 第二句：[1,1,1,0,0,0,0,1,1,1,1,1,1,1]

独热码（One-Hot Encoding）的缺点：

1. **稀疏矩阵问题**：当类别数量较多时，独热编码会导致稀疏矩阵问题。例如，一个具有20个类别的特征，经过独热编码后会生成20列特征，其中大部分（如95%）都是0，这对于神经网络等模型来说很难优化。
2. **高维度问题**：独热编码增加了大量的维度，这些维度特征稀疏，且每个稀疏列之间具有线性关系，容易出现共线性问题。
3. **不总是必要**：如果特征是离散的，并且不用独热编码就可以很合理地计算出距离，那么进行独热编码可能并不是必要的。例如，基于树的算法（如决策树）在处理变量时并不基于向量空间度量，因此不需要进行独热编码。

TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) 是一种用于信息检索和文本挖掘的常用加权技术。它用以评估一个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。其数值通常会被用来作为文件搜索的相关性排名的一个指标。
TF-IDF 由两部分组成：

- **Term Frequency (TF)**: 词频，表示词在文档中出现的频率。
  数学公式如下：

$$(t, d) = \frac{\text{词 } t \text{ 在文档 } d \text{ 中出现的次数}}{\text{文档 } d \text{ 的总词数}}$$

其中，(t) 是某个词，(d) 是某个文档。

- **Inverse Document Frequency (IDF)**: 逆文档频率，表示包含某个词的文档的稀少性。如果一个词在很多文档中都很常见，那么它的IDF值就会很低；反之，如果它只在少数文档中出现，那么它的IDF值就会很高。
  数学公式如下：

$$F(t) = \log \frac{\text{文档集的总文档数}}{\text{包含词 } t \text{ 的文档数} + 1}$$

注意：分母加1是为了避免除数为0的情况。

- **TF-IDF** 的计算公式为：

$$TF\text{-}IDF(t, d) = TF(t, d) \times IDF(t)$$

这个公式结合了词频和逆文档频率，以评估一个词在特定文档中的重要性。如果一个词在文档中出现的频率很高，并且它在整个文档集中很少出现，那么它的TF-IDF值就会很高，表示这个词对于该文档来说很重要。

现在我们回到之前那个例子。

1. 和one-hot一样
2. 和one-hot一样，得到词汇表

小	美	女	太	可	爱	了	我	要	看	的	演	唱	会

3. 计算词频(TF)值.
   首先统计字典中每个字在句子中出现的频率：

小	美	女	太	可	爱	了	我	要	看	的	演	唱	会
2	2	2	1	2	1	1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	1	1	1	1	1	1	1

   为了防止数值过大，将其归一化，得到下表：

小	美	女	太	可	爱	了	我	要	看	的	演	唱	会
0.14	0.14	0.14	0.07	0.14	0.07	0.07	0.07	0	0	0	0	0	0
0.07	0.07	0.07	0	0	0	0	0.07	0.07	0.07	0.07	0.07	0.07	0.07

4. 计算逆文档频率(IDF)值

小	美	女	太	可	爱	了	我	要	看	的	演	唱	会
-0.41	-0.41	-0.41	0	0	0	0	-0.41	0	0	0	0	0	0

5. 计算TF-IDF值

小	美	女	太	可	爱	了	我	要	看	的	演	唱	会
-0.05	-0.05	-0.05	0	0	0	0	-0.02	0	0	0	0	0	0
-0.02	-0.02	-0.02	0	0	0	0	-0.02	0	0	0	0	0	0

TF-IDF的思想比较简单，却很实用。然而，这种方法还是存在数据稀疏的问题，也没有考虑字的前后信息。TF-IDF的缺点：

1. **受分词效果影响大**：TF-IDF的性能在很大程度上取决于分词的效果。
2. **缺乏语义相似度**：TF-IDF只考虑了词频和逆文档频率，没有考虑词与词之间的语义相似度。
3. **没有语序信息**：TF-IDF基于词袋模型，因此不考虑词序。
4. **能力范围有限**：TF-IDF主要用于关键词提取等简单任务，对于更复杂的任务（如机器翻译和实体挖掘）来说，其能力有限。
5. **受样本不均衡影响**：如果训练数据中的样本分布不均衡，可能会对TF-IDF的结果产生较大影响。

n元语法(n-gram)

`n-gram` 是一种在自然语言处理（NLP）和信息检索中常见的概念，用于表示连续的文本或词序列。在 `n-gram` 中， `n` 代表序列中元素的数量。

具体来说， `n-gram` 是从一个文本或句子中连续抽取的 `n` 个词或字符的序列。例如：

- **1-gram (unigram)**: 单独的词或字符。例如，“我”、“是”、“学生”。
- **2-gram (bigram)**: 连续的两个词或字符。例如，“我是”、“是学”、“学生”。
- **3-gram (trigram)**: 连续的三个词或字符。例如，“我是学生”。

以此类推，还有4-gram、5-gram等，但通常随着 `n` 的增大，序列在文本中出现的频率会迅速降低，因此在实际应用中，3-gram或4-gram是较为常见的选择。

现在，我们用2-gram把上面的例子做一遍，即把下面这两句话：

- 第一句：**小美女太可爱了，我爱小美女。**
- 第二句：**我要看小美女的演唱会。**

分解为2-gram词汇表，去重后得到28个元素的词汇表。

小	小美	美	美女	女	女太	太	太可	可	可爱	爱	爱了	了	了我	我	我爱	爱小	我要	要	要看	看小

如果结合One-hot，向量化表示这两句话
第一句：[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0]
第二句：[1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,0,1,1,1,1,1,1,1,1,1,]

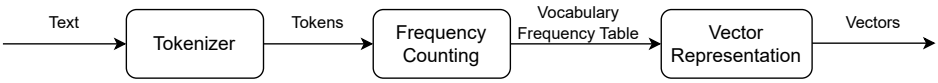
当然，也可以用TF-IDF编码，向量化文本。

需要注意的是，虽然 `n-gram` 模型简单且易于实现，但它也存在一些局限性，例如数据稀疏性（对于某些不常见的 `n-gram` ，其统计信息可能不足）和缺乏上下文信息（n-gram`只考虑了词或字符的局部顺序，而没有考虑整个句子的上下文）。n-gram的缺点：

1. **数据稀疏性**：随着n的增大，n-gram的参数空间呈指数增长，导致数据稀疏性问题。这需要通过平滑算法来解决，否则模型的性能会受到影响。
2. **缺乏长期依赖**：n-gram模型只能建模到前n-1个词，对于长距离的依赖关系无法有效地捕捉。
3. **泛化能力差**：由于是基于统计的模型，对于未在训练数据中出现过的n-gram组合，模型无法进行有效的预测。

词袋模型工作流程

结合上面的例子，我们得到BoW模型的工作流程，如下：



第一步：分词，即将原始的文本，改成计算机统一能处理的最小单位Token的集合。
第二步：统计各Token在文本中出现的频率，得到基于频率的词汇表。
第三步：参考第二步得到基于频率的词汇表，将第一步得到Tokens表示后的文本，用向量化表示出来。

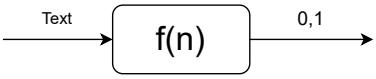
词向量(Word Embedding)表示

BoW模型的痛点

传统的方法，如上述介绍的BoW模型方法，大都是基于规则或模板的方法，很难处理复杂的语义关系。自然语言处理任务往往需要深入理解文本的语义信息。

提出问题

需要一种不同于BoW模型的方法，并且把原始文本表示成计算机理解的0，1后，还要蕴含文本内在的语义信息。
把这句话用计算机能处理方法（即，数学里函数概念）表示出来，如下图所示：



在结合一下BoW模型的工作流程，发现第一步的分词（Tokenizer）还是可以复用的（Reuse）。而输出（Output）部分，用向量化表示0，1是个不错的选择，兼顾扩展性和并行计算的便利。于是乎，我们改进一下上面这张图，得到下图：

现在，我们的问题就转化为聚焦$f(n)$,把文本所蕴含的语义信息，表示成向量表示。
那么如何表示语义信息呢？这里需要提一下，词向量的理论依据——语言分布式假说。

理论依据：语言分布式假说

泽利格·哈里斯（Zellig S. Harris）提出的分布假说（Distributional Hypothesis）认为，如果一个词在多个文本上下文中与其他词有相似的分布，那么这些词在语义上也是相似的。也就是说，词语的含义存在语境中，由其上下文单词的搭配关系体现。

基于分布式假说思想，就把语义表达的问题就简化了，即学习词之间的搭配关系。

> 词向量模型就是学习词(Tokens)之间的搭配关系

解决问题

顺着这个思路，我们考虑学习词之间的搭配关系，很容易想到2种方法。

1. **CBOW：** 给定上下文，预测中心词
2. **Skip-gram：** 给定中心词，预测上下文

第一种方法，就是俗称“完型填空”的CBOW(Continuous Bag of Words)。第二种方法，正式的名词为Skip-gram。和CBOW正好相反。从实际效果来看，Skip-gram要好于CBOW的。这里，笔者再介绍一种方法，即**给定一组词，判断他们是否搭配。**这个方法属于Skip-gram方法范畴，这种方法极大减少计算量同时并未明显牺牲训练效果。

Tips: 为了控制文章长度和行文流畅性，笔者会另起一章来详细介绍Skip-gram。

至此，我们找到了如何把语义表示成向量的方法。沿着词向量的发展历程，又可以细分为静态词向量和动态词向量两个阶段。

静态词向量模型

静态词向量是指在训练词向量时，将每个词语表示为一个固定的向量，即在训练过程中将每个词(Token)映射为一个固定维度的向量，这意味着同一个词在不同的上下文中都会有相同的向量表示。换句话说，静态词向量是指词和向量之间形成了一一映射的关系，通过训练好的向量字典形成固定的映射表，来将文本向量化。

常见的静态词向量模型有Word2Vec、GloVe等。这些模型在大规模语料库上进行训练，通过学习词语之间的共现关系或者上下文信息，将每个词语映射为一个固定维度的向量。

值得注意的是，静态词向量不论语境，都将同一Token编码成同一种词向量。无法表达“一词多义”。这个观点在基础篇有较深入的探讨，这里就不做重复描述了。

Word2Vec和GloVe

Word2Vec是静态词向量的主流模型之一，其中包括Skip-gram和CBOW两种训练方式。
值得一提的是，Word2Vec会根据指定窗口的上下文之间的搭配关系来训练词向量。而这上下文窗口大小是一个重要的参数，窗口越大，训练出来的词向量捕捉Token间共性信息（即文本全局信息）能力越强，但是捕捉Token个性信息能力越弱。

GloVe是一种融合矩阵分解(SVD)和Skip-gram模型优点的静态词向量模型，它同时具有二者的优点，是Word2Vec一种改良。主要改进的部分在上下文窗口。在Skip-gram基础上，加入全局语料特征考量的改进算法，是目前最常用的静态词向量训练法。

静态词向量的优缺点

- 优点：静态词向量具有固定的表示形式，简单且易于实现，适用于各种自然语言处理任务。
- 缺点：由于静态词向量无法捕捉上下文信息，因此在处理具有复杂语义的文本时可能存在一定的局限性。

一言以蔽之，优点是简单易于实现，缺点是无法处理“一词多义”。

动态词向量模型

动态词向量（Dynamic Word Embedding）是自然语言处理中一种重要的技术，用于捕捉词在不同上下文中的动态语义。与传统的静态词向量（如Word2Vec、GloVe等）相比，动态词向量能够根据上下文的变化动态地调整词的表示，从而更准确地反映词在特定语境下的含义。
常见的动态词向量模型有ELMo、BERT等。这些模型通过预训练和微调的方式，将每个词(Token)表示为一个动态的向量。最常用的就是BERT，笔者在基础篇中，有较为详细的介绍，这里就不重复展开了。

参考文献

1. 文本离散表示（一）：词袋模型
2. 自然语言处理（NLP）之二：文本表示（词向量）