11.

这章本意在为对大语言模型(LLM) 和自然语言处理(NLP)感兴趣的初学者和开发者提供一个解释基础为主的指南。

NLP，即自然语言处理，是一个研究如何使计算机能够理解、解释和生成人类语言的学科。简单地说，NLP的目标是使计算机能够“理解”和“产生”人类语言，从而使机器能够与人类进行更自然的互动。

近年来，NLP已经在很多领域得到了广泛的应用，例如聊天机器人、搜索引擎优化、情感分析、自动文摘、机器翻译等。这是因为技术的进步使得NLP的应用更为普及，且其性能也在持续提高。

LLM，即大型语言模型，是这一技术进步的核心。LLM是一种特殊的NLP模型，它是通过在大量文本数据上进行训练来构建的。由于其规模之大，LLM能够捕获语言中的细微差异和复杂关系，从而在各种任务上实现出色的性能。

例如，OpenAI的GPT-4就是一个LLM的例子，它在100多种语言任务上都展现了出色的性能，甚至在某些任务上接近或超越了人类的表现。

了解LLM的核心知识和基本概念在开发LLM应用时是非常关键的，正如建筑的稳定性取决于其基础的坚固程度，同样，开发LLM应用的成功根基也依靠LLM的核心知识和基本概念。接下来，我们将探讨LLM的核心知识。这包括：

嵌入：这是一个关于如何在高维空间中表示单词和短语，以及如何使用这些表示来测量不同文本之间相似性的技术。

注意力机制：这是一种独特的技术，它允许模型在处理时专注于输入的某些特定部分。

Transformer模型架构：这是支撑许多当今最先进语言模型的关键架构。它的革命性之处在于其对NLP领域的影响。

语义搜索：我们将探索如何利用LLM理解查询的含义并找到最相关的结果。特别是，我们会看到这项技术如何彻底改变搜索行业。

随后，我们会深入NLP和机器学习的基础知识，包括：

历史背景：为初学者提供NLP的发展历程，帮助他们理解这个领域的起源和演变。

语言预处理技术：在实际应用NLP之前，必须对文本数据进行清洗和格式化，这是其中的关键步骤。

机器学习模型：自从NLP的起源以来，有哪些模型被开发并使用，以及它们的主要功能是什么。

分类器：这是一个关于如何训练和评估分类器模型的综合指南。

## 11.1 LLM的核心知识

首先，我们将探讨嵌入，这是在高维空间中的单词和短语的表示，以及如何使用它们来测量不同文本之间的相似性。

接下来，我们将探索注意机制，这是一种允许模型在处理期间专注于输入的特定部分的机制。我们将了解Transformer模型架构，这是许多最先进的语言模型的支柱，以及它是如何革新NLP领域的。

最后，我们将深入探讨语义搜索，即理解查询的含义并找到最相关结果的过程。我们将讨论如何使用LLM进行语义搜索，以及这项技术是如何转变搜索行业的。

### 11.1.2 文本嵌入的定义

文本嵌入，包括词和句子的嵌入，是语言模型的核心部分。在“2001太空漫游”这样的老式科幻电影中，主计算机（HAL）能够轻松地与人类交谈并理解他们说的话。当时，让计算机理解和产生语言似乎是不可能的任务，但最新的LLM（如GPT-4）已经能够做到这一点，使人类几乎无法判断他们是与另一个人还是计算机交谈。

NLP的基本任务是理解人类语言。但是，人类用词语和句子交谈，而计算机只能理解和处理数字。那么如何以连贯的方式将词语和句子转化为数字呢？这就是词嵌入所做的事情。我们可以将词嵌入视为为单词分配分数，这些分数具有一些很好的特性（很快我们就会了解到）。

什么是词嵌入？

让我们通过一个直观的测试来理解。在下图中，我在平面上标出了12个单词，包括“Banana”、“Basketball”等。

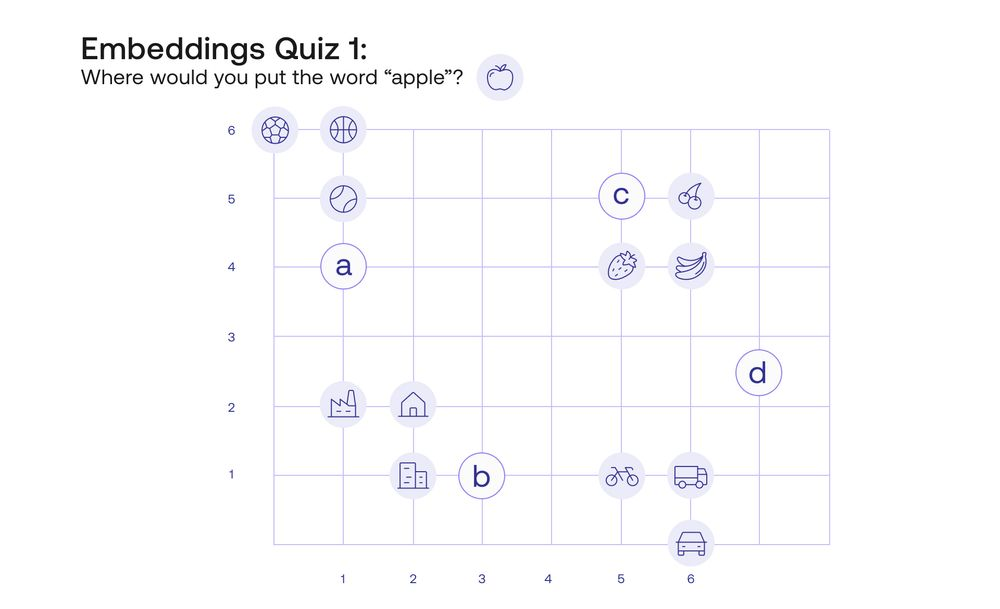


图 11- 1

现在的问题是，你会在这个平面上的哪个位置放置“Apple”这个词呢？最理想的位置是C点，因为“Apple”这个词与“Banana”、“Strawberry”和“Cherry”这些词都很接近，而与“House”、“Car”或“Tennis”这样的词距离较远。这就是词嵌入的实质。我们为每个单词分配的数字是什么呢？简单说，就是词的位置的横坐标和纵坐标。这样，“Apple”这个词就被分配到了[5,5]这个坐标，而“Bicycle”这个词被分配到了[5,1]这个坐标。

对于一个良好的词嵌入，它应具有以下特性：（1）相似的词应对应于接近的点（或等效地，对应于相似的分数）。（2）不同的词应对应于相隔较远的点（或等效地，对应于明显不同的分数）。

句子嵌入的重要性

词嵌入对于理解文本非常有用，但实际上，人类语言远比简单拼凑的词汇更为复杂。它拥有结构、句子等特点。那么，如何表示一个句子呢？

考虑一个简单的方法：如何对句子中的所有单词的分数进行求和？例如，有一个词嵌入为以下单词分配以下分数：

No: [1,0,0,0]

I: [0,2,0,0]

Am: [-1,0,1,0]

Good: [0,0,1,3]

那么，“No, I am good!”这个句子对应的向量是[0,2,2,3]。然而，“I am no good”这个句子也对应同样的向量[0,2,2,3]。这两个句子的含义相差甚远，但它们被解释为完全相同，这显然是不合适的。因此，需要更好的嵌入方法，考虑到单词的顺序、语言的语义和句子的实际含义。

这就引入了句子嵌入的概念。句子嵌入与词嵌入类似，只是它将每个句子与一个充满数字的向量相关联。这种关联方式保证了相似的句子被分配到相似的向量，不同的句子被分配到不同的向量，并且向量的每个坐标都表示句子的某种属性。

嵌入的实用性

文本嵌入已经证明了其重要性，现在是时候开始探索它们的实用性了。以下短语为例：

I like my dog  
I love my dog  
I adore my dog  
Hello, how are you?  
Hey, how's it going?  
Hi, what's up?  
I love watching soccer  
I enjoyed watching the world cup  
I like watching soccer matches

模型返回的嵌入数据显示,相同含义的语句，在向量空间内距离接近。

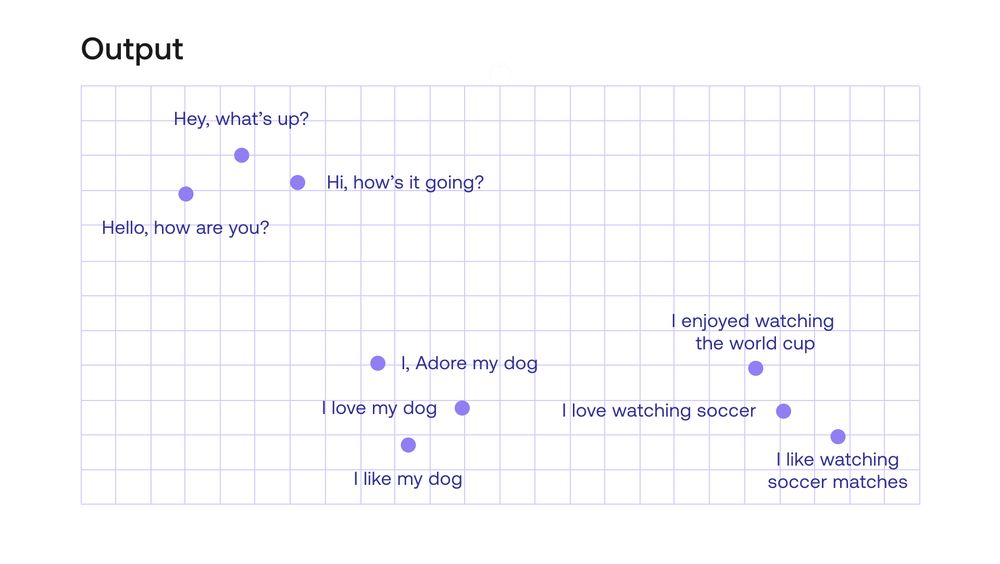


图 11- 2

多语言句子嵌入

大多数词和句子嵌入都依赖于模型受过训练的语言。但在全球化的今天，多语言模型变得尤为重要。OpenAI已经训练了一个大型的多语言模型，支持超过100种语言。以下是几个英语、法语和西班牙语的句子示例：

The bear lives in the woods  
El oso vive en el bosque  
L’ours vit dans la foret  
The world cup is in Qatar  
El mundial es en Qatar  
La coupe du monde est au Qatar  
An apple is a fruit  
Una manzana es una fruta  
Une pomme est un fruit  
El cielo es azul  
The sky is blue  
Le ciel est bleu

模型返回的嵌入数据显示，它能够识别关于熊、足球、苹果和天空的句子，即使它们是用不同的语言编写的。

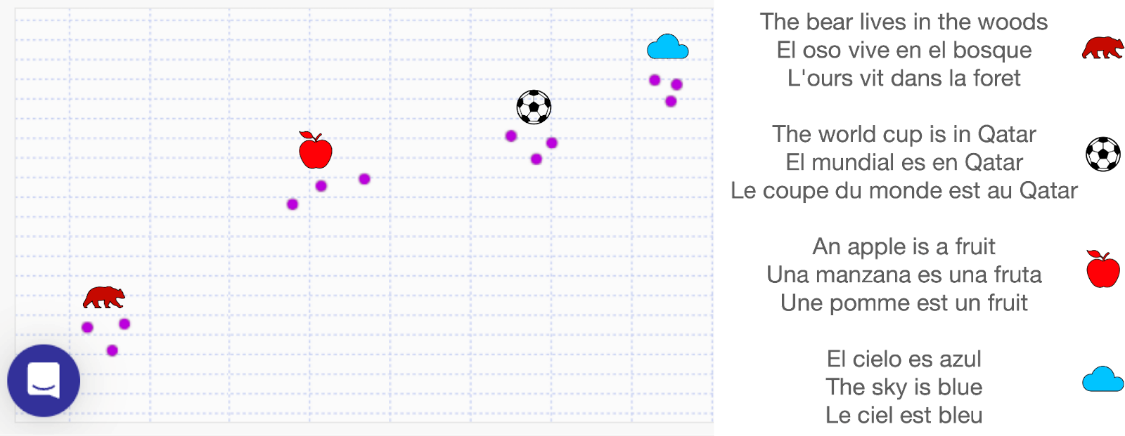


图 11- 3

文本嵌入在LLM中的角色

词和句子嵌入是LLM的基石。它们是大多数语言模型的基本构建块，因为它们将人类的语言（词语）转化为计算机语言（数字），同时捕获词语之间的关系、语言的语义和细微差别。句子嵌入可以扩展为语言嵌入，其中每个句子的数字都与语言无关。这些模型对于翻译以及在不同语言中搜索和理解文本非常有用。

对于LLM，判断两个词或两个句子是否相似或不同是至关重要的。幸运的是，词和句子嵌入为此提供了有力的工具。

### 11.1.3 点积和余弦相似性

我们将探讨几种不同的相似性概念，并为开发者展示如何在LLM应用中有效地使用这些概念。

对于每个LLM，知道两个词或两个不同的句子是否相似或不同是非常关键的。词和句子嵌入为此提供了有力的工具。简而言之，词嵌入将每个词与一组数字（向量）关联起来，这样词的语义属性就可以转化为数字的数学属性。句子嵌入则更为强大，因为它们将每个句子与一组数字关联起来，这些数字也携带了句子的重要属性。例如，OpenAI平台的OpenAI嵌入为每个句子分配一个长度为4096的向量。

了解嵌入的基础后，我们可以使用它们来查找相似性。一旦得到了文本的嵌入，我们可以计算它们之间的相似性。本文将定义两种相似性：点积相似性和余弦相似性。这两种都是确定两个词（或句子）是否相似的有用方法。点积相似性和余弦相似性是两种常用的计算方法。

点积相似性

为了简化问题，考虑一个只有4个句子的数据集，每个句子都被分配了两个数字。例如，电影标题“Rush Hour”和“Rush Hour 2”被分配了相似的数字，因为它们在某种程度上是相似的。

You’ve Got Mail: [0, 5]  
Rush Hour: [6, 5]  
Rush Hour 2: [7, 4]  
Taken: [7, 0]

点积是一种创建相似性分数的方法。在这个方法中，如果两部电影的得分匹配，那么乘以两部电影的行动分数，然后乘以两部电影的喜剧分数，并加它们，这个数字就会很高。例如Rush Hour: [6, 5]，6代表行动分数6分，5代表喜剧分数是5分。计算过程为：

[You’ve got mail, Taken] = 0\*7 + 5\*0 = 0

[Rush Hour, Rush Hour 2] = 6\*7 + 5\*4 = 62

电影标题“Rush Hour”和“Rush Hour 2”被分配了相似的数字，计算的结果是62，数字就很高，而“You’ve got mail”和“Taken”的计算结果是0。这个例子直观地反映了两个向量之间的相似性。对于相似的句子，它们的嵌入向量的点积会很大；而对于不相似的句子，点积则相对较小。

余弦相似性

在开发LLM应用时，经常需要对句子或词语之间的相似度进行量化评估。其中，一种广泛使用的方法是余弦相似性。

余弦相似性基于向量间的夹角来衡量它们之间的相似度。这种方法特别适用于评估高维空间中的数据点之间的相似度，例如在LLM应用中的文本嵌入。

考虑一个简化的例子：在二维平面上，将电影嵌入为点，其中横坐标表示动作得分，纵坐标表示喜剧得分。电影的嵌入可能看起来像点在平面上。例如，“You’ve Got Mail”与“Taken”之间的距离很远，因为它们是非常不同的电影。而“Rush Hour”与“Rush Hour 2”非常接近，因为它们是相似的电影。

虽然欧几里得距离可以测量两点之间的距离，但它不总是能够很好地表示相似性。特别是当数据点在高维空间中非常接近时，角度测量更为合适。

这里，我们引入余弦相似性。余弦相似性衡量的是从原点出发到两句子所形成的两射线之间的夹角的余弦值。当两点非常接近时，这个角度会很小，其余弦值接近1，表示它们之间的相似度很高。

例如，在之前的电影示例中，“You’ve Got Mail”与“Taken”之间的角度为90度，其余弦值为0，表示它们之间的相似度为0。而“Rush Hour”与“Rush Hour 2”之间的角度为11.31度，其余弦值为0.98，表示它们之间的相似度非常高。

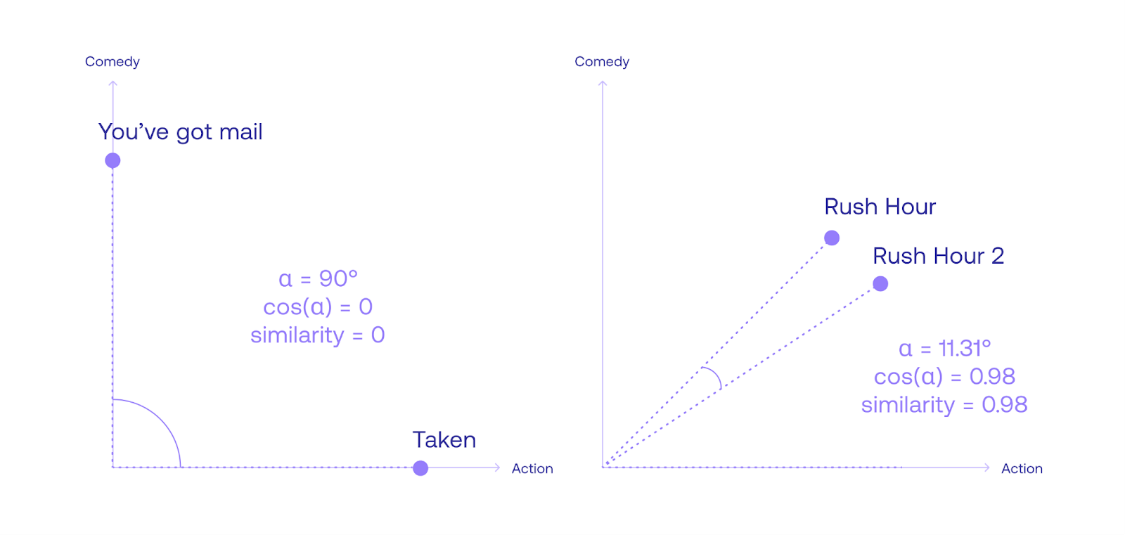


图 11- 4

### 11.1.3 注意力机制

在LLM应用的开发过程中，一个核心的技术挑战是如何准确处理多义词。为了有效解决这一问题，LLM引入了注意力机制。

在之前，我们已经了解了词嵌入和句子嵌入以及如何衡量词汇和句子之间的相似性。简而言之，词嵌入是一种将词与数字列表（向量）相关联的方法，使得相似的词产生相近的数字，而不同的词产生距离较远的数字。

但是，词嵌入面临一个重要的问题：如何处理具有多种定义的词。例如，单词“bank”可以指河岸或金融机构。传统的词嵌入为“bank”分配相同的向量，不考虑其上下文。为了解决这一问题，我们需要注意力机制。注意力机制可以根据上下文为单词提供特定的向量，从而为单词提供上下文信息。

为了理解注意力机制，考虑以下两个句子：

句子1：“The bank of the river.”

句子2：“Money in the bank.”

在这两个句子中，“bank”的含义完全不同。第一个句子中的“bank”指的是河岸，而第二个句子中的“bank”指的是金融机构。如何让计算机理解这两种不同的含义呢？

解决的关键是查看邻近的词。在第一个句子中，“river”这个词为我们提供了上下文，而在第二个句子中，“money”为我们提供了上下文。因此，为了理解“bank”的上下文，我们需要考虑其他单词。

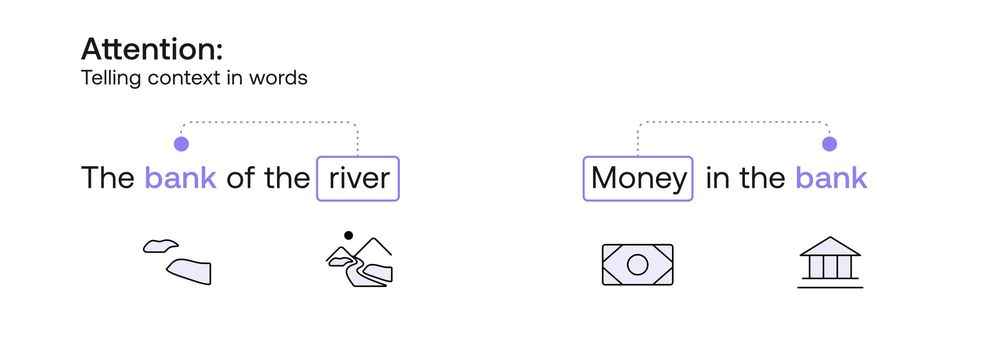


图 11- 5

这就是注意力机制的工作原理。它考虑了句子中的所有词，并为目标词（如“bank”）提供上下文信息。注意力机制可以为每个词提供一个与上下文相关的向量，从而使LLM能够更准确地理解每个词的含义。

为了在数学模型中表示词的上下文关系，我们可以通过调整词嵌入的向量来“移动”一个词更靠近另一个词。例如，为了使“bank”更接近“river”，我们可以将其向量与“river”向量的加权平均进行混合。权重可以基于两个词之间的相似性来决定。

假设我们有以下词向量：

River: [0,5]

Money: [8,0]

Bank: [6,6]

我们假设还有2个新的嵌入向量：

Bank1（与“river”更接近）: [5.4, 5.9]

Bank2（与“money”更接近）: [6.4, 4.8]

如你所见，“bank1”更接近“river”，而“bank2”更接近“money”。 把方括号内的两个数字看成平面上的坐标，其中第一个数字是水平坐标，第二个是垂直坐标。Bank1的纵坐标5.9更接近“river”的5。 Bank2的横坐标6.4更接近“money”的8。与通过生成上下文敏感的嵌入新的向量，我们能够更准确地捕捉到一个词在不同语境中的含义。在上面的例子中，通过创建两个上下文敏感的嵌入“Bank1”和“Bank2”，我们能够区分“bank”这个词在与“river”和“money”两种上下文中的不同含义。

当在实际的句子或段落中使用“bank”这个词时，我们会依赖于这些上下文敏感的嵌入来确定其真正的意思。例如，如果句子中提到了“river”，则更可能使用“Bank1”这个嵌入，因为它更接近“river”的上下文。相反，如果句子中涉及金融或存款，我们则可能使用“Bank2”这个嵌入，因为它与“money”的上下文更为接近。

这种方法使我们可以为多义词创建多个上下文相关的嵌入，从而为LLM应用提供更准确的表示。

## 11.2 Transformer模型

Transformer模型在机器学习领域中迅速崭露头角，特别是在处理文本上下文时表现出色。为了帮助开发者深入理解这一技术并在LLM应用中发挥其最大潜力，本章将详细探讨Transformer的架构及其工作原理。

Transformer模型是机器学习中最引人注目的新发展之一。这些模型能够撰写故事、随笔、诗歌，回答问题，进行语言翻译，与人类交流，甚至通过对人类来说困难的考试！但它们究竟是什么呢？幸运的是，Transformer模型的架构并不复杂，它只是一些非常有用组件的连接，每个组件都有其特定的功能。

简而言之，Transformer模型如何工作？当输入一个简单的句子，如“Hello, how are”，Transformer可以预测出最可能的下一个词，例如“you”。这是因为Transformer能够跟踪所写文本的上下文，从而使生成的文本有意义。

这种逐词构建文本的方法可能与人类形成句子和思考的方式不同，但这正是Transformer模型如此出色的原因：它们能够非常好地跟踪上下文，从而选择恰当的下一个词汇。

那么，Transformer模型是如何训练的呢？答案是使用大量的数据，事实上，是互联网上的所有数据。基于这些数据，当开发者输入“Hello, how are”时，Transformer知道基于所有互联网文本，下一个最佳词汇是“you”。

现在，让我们深入了解Transformer的架构。Transformer有四个主要部分：标记化、嵌入、位置编码、Transformer block（多个）、Softmax。

其中，Transformer block是最复杂的部分。多个这样的块可以被连接起来，每一个都包含两个主要部分：注意力和前向馈送组件。

1. 标记化(Tokenization) 标记化是文本处理的第一步。它涉及将每个单词、前缀、后缀和标点符号转换为一个已知的令牌。例如，句子“Write a story”将被转换为四个相应的令牌：<Write>, <a>, <story>和<.>。

2. 嵌入(Embedding) 经过标记化后，下一步是将这些令牌转换为数字，这就是嵌入的作用。它将每个令牌映射到一个数字向量，如果两个文本片段相似，它们对应的向量也会很相似。

3. 位置编码(Positional encoding) 为了确保句子中的每个单词在处理时能够保持其原始位置信息，引入了位置编码。它通过添加一系列预定义的向量到每个词的嵌入向量来实现。

4. Transformer block。Transformer的核心是由多个Transformer block组成的。每一个Transformer block都包含两个主要部分：注意力组件和前馈组件。

5. 注意力(Attention) 注意力机制是Transformer模型中的关键技术，它能够为每个单词提供上下文信息。例如，在句子“The bank of the river”和“Money in the bank”中，单词“bank”的含义在两个句子中是不同的。注意力机制通过分析句子中的其他单词（river和money）来为每个单词提供上下文，确保其在生成或处理文本时具有正确的含义。

为了更进一步增强这一机制的能力，引入了多头注意力(multi-head attention)技术，它使用多个嵌入来修改向量并为它们添加上下文。

总之，Transformer模型为LLM应用开发者提供了一个强大的工具，可以处理和生成高质量的文本。通过深入理解这些技术，开发者可以为最终用户创建更加高效和流畅的LLM应用。

### 11.2.1 Softmax层和后训练

在现代LLM应用开发中，Transformer模型已经成为了一个不可或缺的组件，为开发者提供了强大的文本处理和生成能力。以下是关于Transformer模型的深入技术分析，旨在为初学者和有意向使用LLM构建应用的开发者提供指导。

1. Softmax层 Transformer模型通过多层的Transformer block来构建，每一层都包含注意力和前馈层，从而形成了一个大型的神经网络，用于预测句子中的下一个单词。Transformer为所有单词输出分数，并为句子中最可能的下一个单词给出最高分数。

Softmax层的作用是将这些分数转化为概率值。例如，Transformer为单词“Once”给出了0.5的概率，而为“Somewhere”和“There”分别给出了0.3和0.2的概率。通过采样，选择概率最高的单词作为输出。

2. 后训练(Post Training) 。虽然了解了Transformer的基本工作原理，但为了使其在实际LLM应用中发挥出更好的效果，还需要进行后续的训练。例如，当询问Transformer“Algeria的首都是什么？”时，理想的回答是“Algiers”。但由于Transformer是基于整个互联网进行训练的，可能会给出不同的答案。

为了改善这种情况，可以进行后训练，即在整体训练完成后，再对模型进行特定任务的训练。这就像对人进行特定任务的培训一样。通过后训练，可以使Transformer在特定任务上，如回答问题、进行对话或编写代码上表现得更好。

Transformer模型为LLM应用开发带来了巨大的潜力。通过深入理解其工作原理和技术细节，开发者可以更有效地利用这一工具，为最终用户创造出更加强大、高效的LLM应用。

## 11.3 语义搜索

在LLM应用开发的世界中，语义搜索已经成为了一个核心技术。与传统的关键字搜索相比，语义搜索提供了更高的准确性和灵活性，使得开发者可以为用户提供更加丰富和准确的搜索体验。

语义搜索使用文本嵌入和相似度来构建一个查询模型。与此不同，传统的关键字搜索依赖于查询和响应之间共同词汇的数量。但是，这种方法往往无法捕捉到文本中的真正含义。例如，考虑以下查询和一组响应：

查询：世界杯在哪里？

响应：

世界杯在卡塔尔。

天空是蓝色的。

熊住在森林中。

苹果是一种水果。

传统的关键字搜索可能会选择与查询拥有最多共同词汇的响应，但这并不总是返回正确的答案。而语义搜索则会选择语义上与查询最匹配的响应。

文本嵌入是将每个文本片段（可以是一个单词或一个完整的文章）转换为一个数字向量的方法。这些向量可以使用各种算法（如OpenAI的嵌入模型）生成，并可以通过降维算法减少到更易于处理的尺寸。这些向量可以被绘制在平面上，使我们可以可视化查询和响应之间的距离。

尽管可以使用欧几里得距离来测量查询和响应之间的距离，但相似度通常提供了更好的结果。通过比较文本嵌入向量之间的相似度，可以确定哪些响应与给定查询最匹配。

在现代LLM应用开发中，语义搜索已经成为一个不可或缺的技术。这一技术的核心在于文本嵌入和相似度的计算，它们共同为开发者提供了一个强大的工具来增强用户的搜索体验。

在之前的章节中，我们简要介绍了相似度的两种计算方式：点积相似度和余弦相似度。这两种方法都可以用于判断两个文本片段是否相似。余弦相似度特别有趣，因为它的取值范围在0到1之间，可以直观地表示文本之间的相似性。

### 11.3.1 语义搜索的工作原理

首先：什么不是语义搜索？ 在我们学习语义搜索之前，让我们看看什么不是语义搜索。在语义搜索之前，最流行的搜索方式是关键字搜索。想象一下，你有很多句子的列表，这些句子是回应。当你提问（查询）时，关键字搜索会查找与查询中共有的单词数量最多的句子（回应）。例如，考虑以下查询和一组回应：

查询：世界杯在哪里？

回应：

世界杯在卡塔尔。

天空是蓝色的。

熊住在森林里。

苹果是一种水果。

通过关键字搜索，你可以注意到回应与查询有以下共同的单词数量：世界杯在卡塔尔。 （4个共同的词） 天空是蓝色的。 （2个共同的词） 熊住在森林里。 （2个共同的词） 苹果是一种水果。 （1个共同的词） 在这种情况下，获胜的回应是1号，“世界杯在卡塔尔”。幸运的是，这是正确的回应。但是，情况并非总是如此。想象一下，如果有另一个回应：

我杯中的咖啡在世界的哪个地方？

此回应与查询有5个共同的词，所以如果它在回应列表中，它就会获胜。这是不幸的，因为这不是正确的回应。

我们该怎么办？我们可以通过删除“the”、“and”、“is”等停用词来改进关键字搜索。我们还可以使用TF-IDF等方法来区分相关和非相关的词。然而，如你所想，总会有一些情况，由于语言的模糊性、同义词和其他障碍，关键字搜索将无法找到最佳的回应。所以我们转向下一个表现非常好的算法：语义搜索。

简而言之，语义搜索的工作原理如下：（1）它使用文本嵌入将单词转换为向量（数字列表）。（2）使用相似性来找到回应中与查询对应的向量最相似的向量。（3）输出与这个最相似的向量对应的回应。

要执行语义搜索，首先要计算查询和每个句子之间的相似度，然后返回相似度最高的句子。对于LLM应用开发者来说，这意味着可以通过简单的算法迅速找到与查询最相关的答案，从而为用户提供更精确的搜索结果。

执行语义搜索的常用算法是最近邻算法，这是一个简单且实用的算法，通常用于分类。在这个上下文中，最近邻算法会查找数据集中与给定点最近的点。然而，该算法在大型数据集中可能效率较低。为了提高效率，开发者可以使用近似最近邻算法或其他优化策略，如Inverted File Index和Hierarchical Navigable Small World。

LLM应用开发者应当注意到，语义搜索的性能高度依赖于文本嵌入的质量。新的多语言嵌入模型为开发者提供了一个强大的工具，支持100多种语言的搜索。这意味着开发者可以使用任何一种语言的查询，并在所有其他语言中搜索答案。

尽管文本嵌入和相似度在语义搜索中发挥了关键作用，但它们并不总是能够提供最佳的搜索结果。例如，考虑一个查询：“世界杯在哪里？”。虽然正确的答案是“世界杯在卡塔尔”，但模型可能会返回与查询语义上更接近的其他响应，如“上届世界杯在俄罗斯”。

在进行相似性搜索时，我们的目标是找到文本中的语义意义，而不仅仅是基于表面上的词汇匹配。这需要对语言的深入理解和处理，而这正是NLP的核心。

## 11.4 NLP与机器学习基础

当我们执行相似性搜索时，我们追求的是文本的深层次含义，而不只是停留在字面上的词语对应。为了实现这一点，我们需要深入地理解和处理语言，这也是NLP的关键所在。

NLP是计算机科学、人工智能和语言学的交叉领域，致力于让计算机能够理解、解释和产生人类语言。

首先，我们将回顾NLP的历史背景及其在LLM应用中的各种实际应用。我们还将介绍处理自然语言文本所面临的各种挑战及相关技术。

随后，我们将专注于文本预处理的主题。为了确保LLM应用的高效性和准确性，我们需要对文本数据进行清洗和准备。这一过程涉及到如何将文本分解为单词或符号（tokenization），如何找到词的基本形式（stemming）和词形还原（lemmatization）。完成这些步骤后，我们将探讨如何将文本转化为向量，以便模型能够处理它们。

此外，为了使LLM应用更加智能，我们将深入研究构建和评估分类器的技术。分类器是能够自动将文本分类到不同类别或类的算法。在此，我们将涉及到特征工程、监督学习和评估指标等关键技术。

### 11.4.1 NLP的演变历程

NLP是一种允许计算机理解、解释和翻译人类语言的人工智能子领域，并能生成文本。其目的是通过处理非结构化文本数据，促进计算机与人类之间的更好沟通。

结合计算机科学和计算语言学的技术，NLP尝试缩小人与计算机之间的沟通鸿沟，使计算机能够读取文本、听取语音并理解信息含义。NLP广泛分为两大领域：自然语言理解和自然语言生成，这两者虽然都属于NLP，但具有明显的差异：

自然语言理解涉及分析非结构化文本的语义和语法。自然语言生成则使用结构化数据生成非结构化文本。

从第二次世界大战后，人们对于能够从一种语言翻译到另一种语言的机器产生了极大的需求。然而，当时使用的技术非常原始，主要依赖于字典查找和硬编码规则来排序单词。

在1950年代末，研究者们开始认识到开发NLP的某些问题。其中一个研究者是Noam Chomsky，他观察到NLP中使用的模型应该能够像人类一样识别语法正确的句子。他在1957年出版的《句法结构》一书中，革命性地介绍了如何在计算机中表示语言学概念。他声称语法是生成的，并为计算机中的语言引入了一个数学模型。

从1957年到1970年代，开发NLP的研究者之间出现了冲突观点，这导致了两组NLP的出现：基于语言学的NLP和基于统计的NLP。在这个时代，基于语言学的NLP取得了显著的进展，但结果仍然不尽如人意。

直到1980年代，带有NLP功能的计算机系统还在使用复杂的“硬编码”规则。计算机科学家们意识到，自然语言问题的孤立问题并不有效。这一挑战导致了其他科学领域的合并，包括语言学。因此，为了弥合计算机科学和语言学之间的差距，进行了更多的研究。

随着计算能力的增强，1980年代后期见证了机器学习算法的转变，决策树是最早采用的算法之一，它们能够使用概率和统计来处理NLP。

进入2000年后，研究者们开始使用深度学习算法来解决NLP中的问题。随着可用的文本数据的大幅增加、更快的CPU和更多的存储空间，训练LLM变得更加容易，这些模型使用神经网络提供更好的结果。研究者们使用卷积神经网络进行句子分类、情感分析和文本摘要等任务。

目前，现代NLP中的Transformer也为其发展做出了贡献，解决了如问答系统、机器翻译等问题。NLP中的Transformer是一种新型的神经网络架构，专门处理序列到序列的任务。在深度学习应用中，它们有效地将输入序列转化为输出序列。

有了这么多的进步，神经网络的应用已经大大增长，现在你可以使用大型的预训练模型来创建NLP驱动的应用。

LLM应用开发中的传统机器学习方法

随着深度学习的出现和流行，NLP领域发生了巨大的变革。然而，在深度学习成为主流之前，研究者们使用了各种传统方法来构建NLP模型。这些方法在当今仍然有其应用场景，尤其是在某些特定的NLP任务中。

以下是几种用于执行NLP任务的流行机器学习方法及其简要描述：

N-Gram模型：这是一种基于训练的概率模型，可以估计文本中词序列的概率分布。

Logistic回归：这是一个在文本分类任务中非常流行的算法，它使用概率生成函数为任何给定的数字输入产生一个0到1之间的值。

贝叶斯：这是一个使用概率预测输出类别的可能性的监督算法。它使用贝叶斯定理，假设数据中的特征是相互独立的。

Markov模型：这些模型非常适合序列数据，因为它们可以预测随机变量序列的概率。

作为一名开发者，特别是对于那些希望使用LLM构建应用的初学者，了解这些传统方法对于整体理解NLP技术发展有很大帮助。尽管现代技术已经发展得很快，但这些传统方法在某些场景中仍然很有价值。

### 11.4.2 NLP文本预处理

在构建LLM应用时，开发者通常会发现，处理和清理数据是实现最佳模型性能的关键。在NLP领域，这一步骤通常被称为文本预处理。它不仅提高了数据的质量，还最终增强了模型的表现力。

文本预处理的主要目标是将嘈杂的文本转化为机器学习模型可以理解的形式。这一过程为进一步的分析和报告准备了清晰、有序的文本数据。

当开发者在NLP项目中收集文本数据时，通常首先从数据库或非正式设置（如博客帖子、社交媒体、电子商务网站、消息板、新闻组和维基）中提取数据。由于文本数据的非结构化特性，其格式和质量可能会有所不同，这可能会引入噪声，影响构建的NLP模型的性能。因此，学习如何有效地预处理这些数据是至关重要的。

文本数据清理

以下是一些用于清理文本数据的众所周知的方法：

Tokenization（标记化）：这是NLP中的一个基础步骤，它将文本分解为较小的块，如词、短语、符号或其他有意义的元素。

Normalization（规范化）：规范化是将具有相似含义的多个单词标准化的过程，将它们转化为单一的规范形式。

Stop Word Removal（停用词移除）：某些词在文本数据中频繁出现，但通常不增加数据的实际意义，这些词被称为停用词。

Stemming（词干提取）：通过移除词的前缀和后缀来清理文本数据。

Lemmatization（词形还原）：这是NLP中使用的一个过程，它将一个词的各种屈折形式转化为其基本形式或词典形式。

文本预处理是为LLM应用优化数据的关键一步。这一过程确保了文本数据的清晰性和一致性，为开发者提供了一个坚实的基础，以构建和训练高性能的NLP模型。

LLM应用开发：从文本到向量的转换

在LLM应用开发的旅程中，将文本转化为数字或向量形式是一个核心步骤。此过程被称为文本向量化，其目的是为各种机器学习算法提供数值输入。

机器学习和深度学习模型需要数字数据作为输入，因为它们不能像人类那样直接处理数据。以下是执行文本向量化的一些常用方法：

One Hot Encoding（一热编码）：此方法采用文本数据中的唯一单词并为每个单词生成向量。

Count Vectorizers（计数向量化器）：与一热编码类似，但它能够捕获单词在文本数据中的出现频率。

Bag of Words（词袋模型）：这是一种提取文本数据特征的方法，不考虑单词出现的顺序。

N-Grams（N元模型）：N-grams代表句子中彼此相邻的一系列单词或标记。

TF-IDF（词频-逆文档频率）：这是一个表示单词在文本中重要性的比率。

为开发者准备的LLM应用中，上述方法可以帮助将原始文本转化为模型可以理解的格式，从而实现更高的模型性能和准确性。

### 11.4.2 构建分类器

在机器学习领域，分类器的任务是为输入数据分配一个类别或类。开发者可以选择使用监督或非监督模型。本章将为初学者和希望使用LLM构建应用的开发者展示如何构建一个简单的监督分类模型。

数据加载与清洗

构建分类器第一步通常是数据收集和清洗。开发者需要确定数据来源是数据库、文件还是在线网页。根据数据来源，选择相应的脚本加载数据。

加载数据后，使用以下方法清洗文本数据以去除噪声并提高其质量：（1）通过分词将数据集中的句子拆分成单词。（2）使用停用词典，通过停用词移除过程过滤掉文本数据中的所有停用词、数字和标点符号。（3）检查所有带有前缀或后缀的剩余单词，并通过词干提取过程将这些单词转换为根词。

文本向量化

为了使机器学习算法能够处理数据，开发者需要将文档转换为数字表示。可以使用词袋算法来实现这一点。此算法将遍历所有数据中的文档。

在此步骤中，确保使用单词出现的频率作为创建向量的评分方法。同时，确保将目标标签转换为数字。

模型训练

为了构建监督分类器，首先确定要使用的算法，经常使用的是贝叶斯算法。对于语料库中的每个文档，现在都有其关联的向量表示和目标变量。

在训练模型之前，首先将数据分为两组：训练数据和测试数据。建议训练集占数据的70%，测试集占剩下的30%。

使用训练数据集训练贝叶斯算法，其中向量作为自变量，目标标签作为因变量。这将输出适合的模型。

分类器评估

对于希望利用LLM构建应用的开发者来说，评估模型的表现是至关重要的步骤。在分类任务中，我们的目标是预测输入数据所属的类别。例如，我们可能需要判断一封电子邮件是否为垃圾邮件。

为了确保分类器的有效性，以下是四个最常用的评估指标：

**准确度 (Accuracy)**：这是一个简单的指标，用于测量分类器的整体效果。但仅在目标类别平衡的情况下使用这个指标可能会产生误导。

**精确度 (Precision)**：这个指标衡量模型预测为正的结果中有多少是真正的正样本。对于垃圾邮件过滤，这意味着被标记为垃圾邮件的消息中有多少实际是垃圾邮件。

**召回率 (Recall)**：召回率量化了模型能正确识别的正样本数。在垃圾邮件过滤的例子中，召回率是指被发送到垃圾邮件文件夹的垃圾邮件与实际垃圾邮件总数的比例。

**F1得分 (F1-Score)**：这是一个用于评估二元分类器的指标，通过计算召回率和精确度的调和平均值得到。

此外，混淆矩阵是一个能够显示分类模型性能的表格，它比较了实际值和预测值。这四种结果（真阳性、假阳性、真阴性、假阴性）构成了混淆矩阵，它为不平衡的数据提供了很好的评估。

选择合适的指标是至关重要的，因为不同的场景可能需要对精确度和召回率进行权衡。此外，阈值的选择对于将概率值转化为目标类标签也非常关键。通常，提高一个指标可能会降低另一个指标。

在LLM应用开发中，了解如何评估分类器的表现是至关重要的。这些评估技术不仅帮助开发者跟踪模型的性能，还为开发者提供了调整和优化模型的手段。

### 11.5 本章小结

在这一章中，我们深入探讨了LLM的核心知识和基本概念。我们首先介绍了嵌入技术，解释了如何在高维空间中表示文本，并使用这些表示来测量文本之间的相似性。接着，我们详细探讨了注意力机制，这是一种让模型在处理数据时更加聚焦的技术。Transformer模型架构作为当今许多先进语言模型的核心，其对NLP领域的革命性影响也被详细讨论。此外，我们还介绍了如何利用LLM进行语义搜索，以及这种方法如何颠覆了搜索行业的传统方式。

在章节的后半部分，我们转向了NLP和ML的基础知识。我们概述了NLP的发展历程，为读者提供了这一领域的历史背景。为了有效地应用NLP技术，我们介绍了关于语言预处理的关键步骤和技术。然后，我们回顾了自NLP起源以来所使用的各种机器学习模型及其功能。最后，我们详细讨论了分类的概念，提供了一个关于如何训练、评估分类模型并在实际中应用它们的全面指南。

总的来说，这一章为读者提供了LLM、NLP和ML的综合和深入的知识，为开发良好的LLM应用程序打下了坚实的理论基础。