词嵌入技术原理与应用实践

黄京

May 12, 2025

自然语言处理（NLP）的核心挑战在于语言的非结构化特性与语义复杂性。传统方法如独热编码（One-Hot Encoding）和词袋模型（Bag of Words）仅能捕捉表面统计信息，无法处理同义词、一词多义等语义关联。例如，独热编码将每个词映射为高维稀疏向量，导致「猫」与「犬」的向量距离和「猫」与「汽车」的向量距离相同，显然违背语义直觉。

词嵌入（Embeddings）技术通过将词汇映射到低维稠密向量空间，实现了从符号表示到分布式表示的范式跃迁。这一技术革命性地解决了语义相似性与上下文关联性的建模问题。例如，在向量空间中，「国王」-「男性」+「女性」≈「女王」的向量关系，直观展示了词嵌入对语义关系的几何表达。

1　词嵌入技术原理

1.1　基础概念

词嵌入的核心目标是将词汇从高维稀疏向量（如独热编码的维度等于词表大小）映射到低维稠密向量（通常为 50-300 维）。这种映射使得语义相似的词在向量空间中距离相近。例如，「快乐」与「愉快」的余弦相似度应显著高于「快乐」与「悲伤」。实现这一目标的关键在于上下文关联性：通过分析词汇在语料中的共现模式，模型能够学习到词汇的分布式表示。

1.2　经典模型解析

Word2Vec 是词嵌入领域的里程碑模型，其包含两种架构：Skip-Gram 与 CBOW。Skip-Gram 通过中心词预测上下文词，适合处理低频词；而 CBOW 通过上下文词预测中心词，训练效率更高。两者的损失函数均基于极大似然估计：

$$L = -\sum_{c \in \text{Context}} \log p(w_c|w_t)$$

其中 w_t 为中心词，w_c 为上下文词。为降低计算复杂度，Word2Vec 引入负采样（Negative Sampling）技术，将多分类问题转化为二分类问题。例如，对于正样本（中心词与真实上下文词对），模型输出概率应接近 1；对于随机采样的负样本，输出概率应接近 0。

GloVe（Global Vectors）则从全局词共现矩阵出发，通过优化目标函数直接学习词向量：

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

其中 X_{ij} 表示词 i 与词 j 的共现次数，$f(X_{ij})$ 为加权函数，用于抑制高频词的影响。

1.3 上下文感知的嵌入技术

传统词嵌入模型生成静态向量，无法处理一词多义问题。例如，「苹果」在「吃苹果」与「苹果手机」中的语义差异无法通过单一向量表达。**ELMo**（Embeddings from Language Models）通过双向 LSTM 生成动态嵌入，结合不同网络层的表示，捕捉词汇的多层次语义。而 **BERT**（Bidirectional Encoder Representations from Transformers）基于 Transformer 的注意力机制，通过掩码语言模型（Masked Language Model）预训练，生成上下文相关的词向量。例如：

```
from transformers import BertTokenizer, BertModel
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
inputs = tokenizer("bank␣of␣the␣river", return_tensors="pt")
outputs = model(**inputs)
word_embeddings = outputs.last_hidden_state
```

此代码加载预训练 BERT 模型，对句子「bank of the river」进行编码。`last_hidden_state` 输出包含每个 token 的上下文相关向量，其中「bank」的向量会根据「river」的上下文动态调整，从而区别于「bank account」中的「bank」。

2 词嵌入的应用实践

2.1 基础 NLP 任务

在文本分类任务中，可通过简单平均所有词向量得到句子表示，再输入全连接网络进行分类。例如，使用 Gensim 训练 Word2Vec 模型：

```
from gensim.models import Word2Vec
sentences = [["自然", "语言", "处理", "是", "人工智能", "的", "核心"], ["词嵌入", "技术",
    ↪ "推动", "了", "NLP", "发展"]]
model = Word2Vec(sentences, vector_size=100, window=5, min_count=1, workers=4)
sentence_vector = np.mean([model.wv[word] for word in ["词嵌入", "技术", "NLP"]], axis
    ↪ =0)
```

此处 `vector_size` 定义词向量维度，`window` 控制上下文窗口大小。通过 `np.mean` 对词向量取平均，得到句子级表示。

2.2 高级应用场景

在语义搜索场景中，可通过余弦相似度匹配用户查询与文档向量。例如，将用户查询「智能语音助手」与文档库中的向量进行相似度排序，返回最相关结果。对于跨语言任务，**Facebook MUSE** 项目通过对抗训练对齐不同语言的向量空间，使得「dog」的向量与「犬」的向量在映射后接近。

2.3 实战代码示例

使用 t-SNE 对词向量降维可视化：

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

words = ["国王", "女王", "男人", "女人", "巴黎", "法国"]
vectors = [model.wv[word] for word in words]
tsne = TSNE(n_components=2, random_state=0)
projections = tsne.fit_transform(vectors)

plt.figure(figsize=(10, 6))
for i, word in enumerate(words):
   plt.scatter(projections[i, 0], projections[i, 1])
   plt.annotate(word, xy=(projections[i, 0], projections[i, 1]))
plt.show()
```

此代码将高维词向量投影到二维平面，n_components=2 指定输出维度为 2。可视化结果可清晰展示「国王-女王-男人-女人」的性别语义轴与「巴黎-法国」的地理关联。

3 挑战与优化方向

静态词嵌入的核心局限在于无法处理一词多义与领域迁移问题。例如，「细胞」在生物学与计算机领域分别指向「生物单元」与「电子元件」。动态嵌入模型如 BERT 虽能缓解此问题，但计算成本较高。优化策略包括领域自适应（Domain Adaptation）：在目标领域数据上微调预训练模型，使其适应特定术语分布。例如，在医疗文本上微调 BERT：

```
from transformers import BertForMaskedLM, Trainer, TrainingArguments
model = BertForMaskedLM.from_pretrained('bert-base-uncased')
training_args = TrainingArguments(
   output_dir='./med_bert',
   overwrite_output_dir=True,
   num_train_epochs=3,
   per_device_train_batch_size=16,
)
trainer = Trainer(
   model=model,
   args=training_args,
   train_dataset=medical_dataset
)
```

```
trainer.train()
```

通过 3 轮训练，模型能够学习医疗领域特有的语义模式，提升在该领域的下游任务表现。

4 未来展望

多模态嵌入将文本、图像、语音的表示统一到同一空间，例如 OpenAI 的 CLIP 模型，可将「狗」的文本描述与狗的图像映射到相近向量。在可解释性方向，**Embedding Projector** 等工具允许用户交互式探索高维向量空间，分析模型语义捕获能力。轻量化技术如模型蒸馏（Distillation）可将 BERT 压缩为 TinyBERT，在保持 90% 性能的同时减少 70% 参数量，推动词嵌入技术在移动端的落地。