arXiv:2303.07196v2 [cs.CL] 2 Mar 2024

摘要

基于向量的词表示有助于众多自然语言处理（NLP）任务捕捉语言的语义和句法规律。本文介绍了现有词嵌入方法的特点，并针对多种分类任务对其进行分析。我们将这些方法分为两大类：传统方法主要通过矩阵分解生成词表示，难以很好地捕捉语言的语义和句法规律；而基于神经网络的方法能够捕捉语言中复杂的规律，并在生成的词表示中保留词语关系。我们报告了多项分类任务上的实验结果，并指出了某种方法优于其他方法的具体情形。

现有词嵌入方法的全面实证评估

ObaidullahZaland MuhammadAbulaish
obaidullahzaland1997@gmail.com abulaish@sau.int南亚大学 南亚大学

Mohd. , Fazil
mohdfazil.jmi@gmail.com南亚大学

2020年6月

1 引言

稠密实值词向量表示对于自然语言处理任务至关重要，例如问答 [1], [2], 语义角色标注 [3]–[5], 文本蕴含 [6] 和机器翻译 [7]。这些词表示也称为词嵌入，用于编码词语的语义和句法特征，因此可作为下游任务的输入特征，例如情感分析 [8], [9], 谣言检测 [10], 和假新闻检测 [11]。

自自然语言理解发展的早期以来，大量研究一直致力于从未标注文本语料库中获取这些词表示。研究人员使用了统计模型 [12], [13], 神经网络语言模型 [14], 对数双线性模型 [15], 以及基于上下文的神经模型 [16], [17], 等方法来构建这些词表示。然而，在特定场景下应采用哪种方法，这个问题仍未得到解答。要比较词嵌入方法，我们可以使用内在评估方法或外在评估方法[18]。内在评估方法利用词嵌入之间的语义和句法关系，直接衡量其质量[19],；外在评估方法则将这些词嵌入作为输入特征，用于下游任务，并据此衡量其质量。

含义更接近的词称为语义相关词项，而为此类词生成的嵌入应当在生成的嵌入空间中彼此更近。例如，自行车、脚踏车和单车 等词彼此相似且在语义上相关。句法相关词是指受某种自然语言（如英语）的句法规则约束的词。例如，简短 和简洁地 在句法上彼此相关，其中简洁地 是形容词简短的副词。此外，大 和更大 在句法上相关，后者是前者的比较级形式。

大多数研究都考察词嵌入模型的内在特征。尽管对这些模型进行内在评估有助于揭示这些词之间的语义和句法相似性，但它们并未明确呈现某个模型优于或逊于其他模型的情形。此外，内在评估方法需要额外的资源，即用于主观测试的预定义查询[20]。这些查询可以是词语对，例如形容词及其比较级形式，或国家及其首都；这类查询也称为查询清单[21]。

另一方面，对这些方法进行外在评估可能计算成本高昂且耗时，但它能更深入地揭示这些方法所提取的词表示的质量。要评估这些方法的实际质量，我们应根据它们在真实场景中的用途对其进行比较。窗口大小、数据预处理、数据集偏斜度和词嵌入维度等细微因素，有时会影响下游任务（例如分类）的结果 [22]。

然而，研究人员虽然探讨了内在评估与外在评估方法之间的关系，但相关研究并不全面[23]。此外，这些比较研究并未评估所有现有模型，因为大多数研究仅关注潜在语义分析、word2vec 和 GloVe [24]–[26]。而且，据我们所知，尚无涵盖所有最先进模型（GloVe、Word2vec、ELMo、BERT 和 fastText）的全面外在评估。

优秀的词表示方法应能够处理一些关键问题。

- 最高频的词，如the 或and ，不应影响词向量的质量。
- 稀有词也应具有高质量的词表示。
- 针对多种词义生成多种词嵌入。

除了词嵌入本身具备上述特性之外，这些模型提取的特征还应适用于不同场景；例如，在进行情感分析时，这些特征应能很好地适用于平衡数据和不平衡数据。本文采用使用主流词表示算法训练得到的预训练向量，以适用于不同的分类任务场景。我们还使用这些算法从头开始训练词嵌入，以衡量预训练对各算法在不同任务中的影响。此外，我们研究了模型参数（窗口大小和嵌入维度）对下游任务输出的影响。我们也研究了外部参数对这些模型输出的影响。这些外部参数包括用于预训练的语料库的正式程度、用于预训练的语料库中文本的数量，以及预处理。

本文其余部分安排如下：第 2 节介绍现有的词表示模型，包括传统模型和神经模型。

第 3 节介绍词嵌入模型的属性，并根据这些属性比较现有模型。相关结果已在第 4 节报告。我们对预训练和训练词嵌入都进行了全面比较。最后，我们在第 5 节总结研究结果。

2 先进的词嵌入方法

词嵌入是下游自然语言处理任务的基础。因此，多年来人们提出了许多训练这类嵌入的方法。为便于说明，我们将这些模型分为两类：传统模型和神经网络模型。

2.1 传统模型

传统模型基于分布语义学的理念，根据语料库中存在的统计信息构建词表示。它们以词项频率、词项共现频率 [27], 和词项-文档频率 [12]为基础，生成这些向量表示。最简单的模型——独热编码——使用大小为 $|V|$的向量，其中 $|V|$ 是词汇表的大小，用于表示每个词。生成的词向量在该词对应的特定位置上取值为 1，其余位置均为 0。例如，如果$V = \{have, a, great, day\}$，那么词 $have$ 可表示为 $\{1, 0, 0, 0\}$，词 a 可表示为 $\{0, 1, 0, 0\}$，以此类推。

潜在语义分析（LSA） [12] 是这一类别中影响力最大的模型。LSA 利用语料库中包含的统计信息构建词项-文档频率矩阵 X。随后，LSA 使用奇异值分解（SVD）对该矩阵进行低秩近似，以构建词向量表示。SVD 将共现矩阵 X 分解为三个矩阵： V、V^T和 Σ。

$$X = V\Sigma V^T \tag{1}$$

其中， Σ 是由该矩阵的奇异值组成的对角矩阵， V和 V^T 是由左、右奇异向量组成的正交矩阵。要将矩阵 X近似为秩为 j的低秩矩阵，我们选取最大的 j 个奇异值，以及V 和 V^T 中与这些奇异值对应的右、左奇异向量。

$$X_j = V_j\Sigma_j V_j^T \tag{2}$$

另一方面，超空间类比语言模型（HAL）[27] 使用词项频率构建共现矩阵 X。HAL 在文本上滑动一个“窗口”，窗口内的词被视为共现，其共现强度与它们之间的词项数量成反比，直到

窗口范围内。某个词左侧和右侧出现的词会分别记录。因此，形成了一个包含 n行和 $2n$ 列的矩阵，其中 n表示词汇表的大小。随着词汇表扩大，共现矩阵也会变得十分庞大，词嵌入的大小亦然。因此，从 $2n$ 列中选出方差较大的列，以降低词嵌入的维度。与the等最常见词相关的列频率较高，但对文本提供的信息很少。这些高频或高方差列会不成比例地影响生成的词表示向量之间的距离。

词汇语义相关共现类比（COALS）[28] 使用归一化过程，尽量降低最常见词对生成的词嵌入质量的影响。COALS 移除表示左侧和右侧上下文的不同列，并为每个词添加一列，使共现矩阵变为对称矩阵。此外，COALS 使用条件比率作为矩阵条目，而不是原始的词项共现计数；条件比率表示词a 与词 b 的共现程度是否高于或低于一般水平。词对之间的条件比率可以用皮尔逊相关系数计算。构建共现矩阵后，会从矩阵中移除负值，并将正值替换为其平方根。作者解释说，负相关词项在语义上可能彼此无关，也未在对应同一主题的文本中共同使用。他们认为，移除负相关词列造成的信息损失，比移除低方差词频列更少。

各种传统模型以不同方式利用语料库统计信息来构建共现矩阵。此外，除奇异值分解（SVD）外，其他变换方法（如海灵格主成分分析（HPCA）[29] ）也被用于学习这些词表示。尽管传统模型易于理解，但计算开销很大；由于共现矩阵会变得极其庞大且难以处理，因此在大型数据集[30], 上使用这些模型并不可行。

2.2 神经模型

与传统模型不同，神经模型随着时间推移经历了广泛的发展。这些模型的第一代是神经网络语言模型（NNLM）。尽管早期的神经网络语言模型 [14] 解决了统计模型中存在的维度灾难问题，但显著的提升是在引入循环神经网络语言模型（RNN-LM） [31], [32]之后才出现的。在自然语言文本中，语言建模会预测词序列的 $w_1 w_2 ... w_T$ 概率。

$$P(w_1^T) = \prod_{t=1}^{T} P(w_t | w_1^{t-1}) \tag{3}$$

前馈神经网络语言模型（FNNLM）完成相同的任务。不过，它并不考虑所有历史词，而是采用

基于 n 元语法的思路，只考虑当前词之前的 $n-1$ 个词。在前馈神经网络语言模型（FNNLM）中，

$$\prod_{t=1}^{T} P(w_t|w_1^{t-1}) \approx \prod_{t=1}^{T} P(w_t|w_{t-n+1}^{t-1}) \tag{4}$$

循环神经网络语言模型（RNNLM）略有不同，因为它们具有内部状态空间。这个内部状态空间可以充当内存，存储与此前所有序列相关的信息，并传递给下一个序列，使模型能够处理长度不确定的序列。

这类语言模型的主要问题在于，它们使用 1-of-V 编码，也称为独热编码（OHE）。在 OHE 中，每个词都用一个大小为 V（词汇表大小）的向量表示。词汇表大小增长非常快，可达数百万词。用如此高维的向量表示每个词，会使模型运行缓慢且效率低下。此外，训练集中未出现的词无法表示。这些缺陷促使人们采用稠密词表示，并提出了诸如 [33] 和 [34] 这样的模型。分布式表示（也称为词嵌入）是对词的一种实值表示，其维度低于大小为 V 的向量。嵌入中的每个维度都体现了该词的一种潜在特征 [35]。在接下来的小节中，我们将介绍用于获得这些词嵌入的、应用最广泛的模型。

2.2.1 word2vec

随着时间推移，神经网络语言模型变得越来越复杂。后来人们发现，像 word2vec 这样简单的浅层神经网络模型，效果可能优于这些过于复杂的语言模型。Mikolov 于 2013年 提出了两种基于窗口的神经模型变体：连续词袋（CBOW）和 Skip-gram[34], [36]。

连续词袋模型根据上下文预测中心词，因此每次迭代都会尝试最大化语料库中每个词的以下概率。

$$\sum_{-c\leq j\leq c, j\neq 0} \log p(w_i|w_{i+j}) \tag{5}$$

这里， w_i 是中心词，而 w_{i+j} 是与中心词相距j 的上下文词。另一方面，Skip-gram模型根据中心词预测上下文词。它在每次迭代中都会最大化以下概率。

$$\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{i+j}|w_i) \tag{6}$$

为了计算词与词之间的概率函数，这两种模型都使用 Softmax 函数。

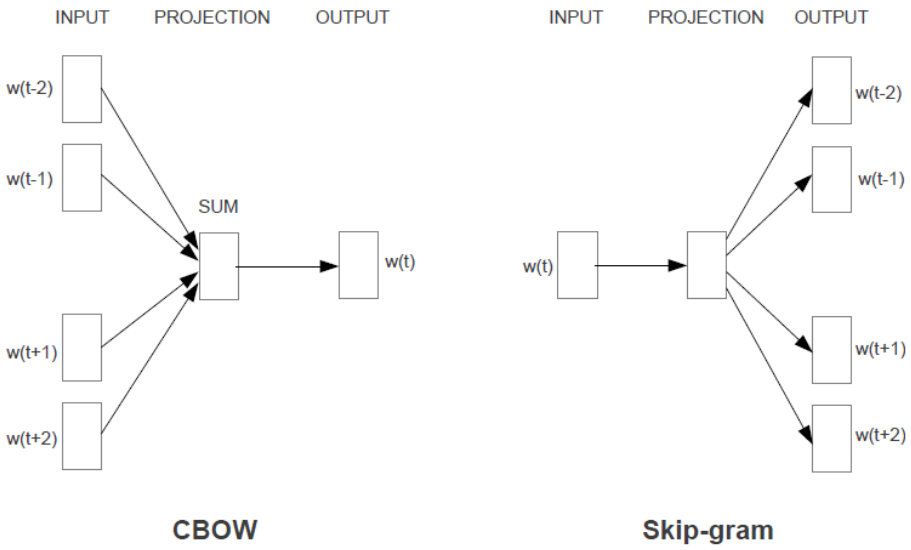

图 1：连续词袋模型使用上下文词预测中心词，而 Skip-gram 使用中心词预测上下文词——图片摘自 Mikolov 的原始 word2vec 论文

$$p(w_c|w_o) = \frac{\exp\left(v_{w_c}^{'T} v_{w_o}\right)}{\sum_{w=1}^{W} exp(v_w^{'T} v_{w_o})} \tag{7}$$

这里，v_w 和 $v_w^{'}$ 是词 w的输入 和输出 向量表示。在这两种模型中，每个词都有两种不同的表示。例如，如果使用跳元模型，当该词是中心词时，会使用输入表示；当该词是上下文词时，则会使用输出表示。最后，我们可以将输入表示和输出表示拼接起来，或者取二者的平均值，以构建最终的词嵌入。

Mikolov 在他的第二篇论文中提出了一些技巧，以改进模型。首先，Softmax 函数效率低下，因为它需要对词汇表中的所有词求和。这样一来，模型处理语料库中的每个词元时都会变得非常缓慢。负采样解决了上述问题。我们不再遍历整个词汇表，而是最大化中心词与上下文窗口中的词（正例）之间的相似度，并从词汇表中随机选择 k 个其他词，最小化中心词与

这些 k 个负面例子之间的相似度。

$$p(w_c|w_o) = \log \sigma(v'^T_{w_c} v_{w_o}) + \sum_{i=1}^{k} \left[\log \sigma(-v'^T_{w_i} v_{w_o}) \right] \tag{8}$$

负样本从幂次分布中抽取，word2vec 论文建议，对于小型和大型训练数据集，k 值分别取 5-20 和 2-5。第二项改进是对高频词进行子采样。word2vec 的作者还指出，有些词（例如 "in"、"the" 和 "or"）比其他词出现得更频繁，却提供较少甚至不提供信息。作者采用了子采样技术来解决这一问题：每个词 w_i 都有一定概率 $P(w_i)$被丢弃。

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{9}$$

这里， $f(w_i)$表示词 w_i 在语料库中的频率，而 t 是选定的阈值。

连续词袋模型在语义任务上的表现不太理想，而 Skip-gram 则能很好地处理语义和句法任务。除了这些任务，word2vec 还引入了一项新的词类比任务。该任务用于检验生成的词嵌入能否保留词与词之间的关系。例如，王子之于公主，正如国王之于王后，这种关系应当在词嵌入中保留下来，表示为 $vec(prince) - vec(princess) = vec(king) - vec(queen)$。

这些模型仅凭一个投影层和简单的架构，就在众多任务中取得了最先进的性能。该模型的问题在于，由于它是基于窗口的模型，因此未能充分利用语料库的全局统计信息。此外，词表示不依赖上下文（不同词义只有一个向量）。2014年推出的全局向量（GloVe）解决了第一个问题。

2.2.2 全局向量（GloVe）

GloVe[37], 在许多方面与传统模型相似。与潜在语义分析（LSA）一样，它会根据文本语料库创建共现矩阵，但有两个主要区别。首先，它创建的是词项-词项共现矩阵，而非 LSA 中的词项-文档共现矩阵；其次，它考虑的不是文档中的共现次数，而是特定范围或窗口内的共现情况。因此，它兼具语料库的全局统计信息和窗口的局部信息。GloVe 的这一特性使其适用于词相似度和词类比任务。

GloVe 还引入了一种新的最小二乘损失函数。

$$J = \sum_{p,q=1}^{V} f(X_{pq})(w_p^T \tilde{w}_q + b_p + \tilde{b}_q - \log X_{pq})^2 \tag{10}$$

其中， V 是词汇表大小， w_p 是中心词（或中心词）的向量， $\tilde{w}_q$ 是˜上下文词的向量， b_p 是目标词偏置， b_q 是上下文词

偏置， X_{pq} 是 w_p 与 w_q 共同出现的次数， f（x）是一个加权函数。加权函数的定义如下：

$$f(x) = \begin{cases} (x/x_{max})^{\alpha} & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases} \tag{11}$$

α 和 x_{max} 是超参数，在论文中分别固定为 0.75 和 100。这种加权函数具有一些理想特性，例如：

a $f(0) = 0$，随着共现次数增多，数据会变得稀疏，大多数实体的值会变为零。因此， f 应当连续，且应定义 $f(0)$， $f(0) = 0$。

b 它不应过度加权较大的共现次数，并且对于较大的数值，其权重应相对较小。

通过选择合适的损失函数并利用全局和局部信息，GloVe 成为一种更适合对低频词和高频词建模的算法。然而，该算法无法为未见过的词生成嵌入，而且仍未考虑词的上下文。书 book 只有单一的词表示，而没有分别为名词 book 和动词 book 提供不同的词表示。2017 年推出的 Fasttext[38], 解决了第一个问题。

2.2.3 fastText

有效地建模罕见词和此前未见过的词颇具挑战性。此外，为每种动词形式分别建立向量表示可能效率低下，尤其是在形态丰富的语言中。fastText 通过改进基础 Skip-gram 模型并融入子词信息，解决了这些问题。在 fastText 中，每个词都表示为其字符 n 元语法的总和，同时考虑了词的形态。尽管 [39]等其他模型也学习了语言的形态规律，但并未在很大程度上利用子词信息。fastText 模型的一个扩展 [40]也解决了拼写错误的词这一问题，因为它会用最接近的正确词替换拼写错误的词。不过，在本文的比较中，我们将考虑基础 fastText 模型。

由于 fastText 对 Skip-gram 模型进行了改进，因此它们的目标函数相似。fastText 以 Skip-gram 目标函数为基础，通过纳入子词信息加以改进。采用负采样的 Skip-gram 目标函数为：

$$\sum_{t=1}^{T} \left[\sum_{c \in C_t} \ell(v_{w_t}^T v_{w_c}^{'}) + \sum_{i=1}^{k} \ell(-v_{w_t}^T v_{w_i}^{'}) \right] \tag{12}$$

v_w 和 $v_w^{'}$ 是相同的输入和输出词向量，而 $\ell(x)$是逻辑损失函数。

fastText 将子词信息引入函数中，并且不再使用公式中的目标词向量，而是使用其 n 元语法表示。以单词“jargon”和 $n = 3$为例，其中 n 是 n 元语法的长度；我们将得到以下 n 元语法：

<ja, jar, arg, rgo, gon, on>

以及特殊序列

<jargon>

词表示通过将其所有字符 n 元语法的表示相加得出。例如，如果 $\mathcal{M} \subset \{1, ..., M\}$ 是一个词的字符 n 元语法集合， z_m 是 n 元语法 m 的向量表示，那么，

$$v_w = \sum_{m \in \mathcal{M}_w} z_m \tag{13}$$

类似地，现在 fastText 的目标函数变为：

$$\sum_{t=1}^{T} \Big[\sum_{c \in C_t} \ell \Big(\sum_{m \in \mathcal{M}_w} (z_m^T v'_{w_c}) \Big) + \sum_{i=1}^{k} \ell \Big(- \sum_{m \in \mathcal{M}_w} (z_m^T v'_{w_i}) \Big) \Big] \tag{14}$$

简单来说，该函数会针对语料库 T中的每个词 t ，最大化中心词的 n 元语法表示与上下文词表示之间的相似度，同时最小化中心词的 n 元语法表示与 k 负样本之间的相似度。

fastText 在句法任务上的表现显著提升，尤其是在意大利语等形态变化丰富的语言上，而在语义任务上的表现则保持不变。不过，使用 n 元语法表示词有助于更好地表示稀有词以及训练期间未出现过的词，但这些表示仍然与上下文无关。

2.2.4 语言模型嵌入（ELMo）

在自然语言中，词的语言上下文决定了词的含义。同一个词在不同语境下可能有不同的含义，这种现象称为多义性。此前已有方法（例如 [41], [42], ）提出模型，通过为每个词学习多个嵌入来解决词表示学习中的多义性问题。然而，这些模型的架构较浅，并且需要预先定义词义类别。ELMo 是一种深层上下文化模型，用于从未标注文本中学习词嵌入。ELMo 表示具有深层和上下文化特性，这是因为它们是双向模型内部层的函数，并且取决于词所处的上下文。ELMo 词嵌入是整个句子的函数。

前向语言模型接收词元（t_1，.....， t_{i-1}），并对 t_i 的概率进行建模，以预测词元序列的可能性，

$$p(t_1, t_2, ..., t_N) = \prod_{i=1}^{N} p(t_i | t_1, ..., t_{i-1}) \tag{15}$$

该模型接收与上下文无关的词元嵌入，并将其输入L 层前向长短期记忆网络（长短期记忆网络）。每一层j 都会输出词元 t_i $\overrightarrow{h}_{i,\ j}^{LM}$ [43] [44]的上下文相关表示。最后一层的长短期记忆网络输出可用于预测下一个词元t_{i+1}。而反向语言模型则使用后续词元（t_{i+1}， t_{i+2}，.....， t_N）对当前词元（t_i）的可能性进行建模

$$p(t_1, t_2, ..., t_N) = \prod_{i=1}^{N} p(t_i | t_{i+1}, t_{i+2}, ..., t_N) \tag{16}$$

与上文相同，实现了 L 层反向长短期记忆网络，并且每一层 j 都会计算词元 t_i的上下文相关隐含表示
$\overleftarrow{h}_i^{LM}$
$_{,\ j}$。

双向语言模型将前向语言模型和反向语言模型结合起来，而 ELMo 会同时最大化双向语言模型的对数似然：

$$\begin{aligned} \sum_{i=1}^{N} (&\log p(t_i | t_1, ..., t_{i-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) \\ &+ \log p(t_i | t_1, ..., t_{i-1}; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s)) \end{aligned} \tag{17}$$

Θ_x和 Θ_s是词元表示和 softmax层的参数。$\overrightarrow{\Theta}_{LSTM}$和 $\overleftarrow{\Theta}_{LSTM}$是前向和反向长短期记忆网络的参数。

模型会为每一层前向和反向长短期记忆网络的每个词元 t_i计算 $2L + 1$ 参数、词元表示 x_i以及隐含表示$h_{i,\ j}^{L}M$。如果对于每个双向长短期记忆网络层，$x_i^{LM} = h_{i\ 0}^{LM}$，且 $h_{i\ ,\ j}^{LM} = [\overrightarrow{h}_{i,j}^{LM};\ \overleftarrow{h}_{i}^{LM,\ j}]$，那么

$$R_i = \{h_{i,j}^{LM} | j = 0, ..., L\} \tag{18}$$

在最简单的情况下，ELMo 会选择顶层 $E(R_i) = h_{i\ L}^{LM}$，如 [45],中所示；更一般而言，它会针对当前任务对所有双向语言模型层进行加权。

ELMo 解决了自然语言理解中的许多问题，包括多义性。与 word2vec 和 GloVe 等较早的词表示模型相比，它采用了深层架构，生成的表示也能够感知上下文。在 ELMo 之后，又出现了许多类似的深度预训练模型，例如：
Open AI GPT[46], Open AI GPT2[47], 和 BERT[17],。

2.2.5 来自Transformer的双向编码器表示（BERT）

BERT 是一种基于多层双向 Transformer 编码器 [48]的语言表示模型。它使用掩码语言模型（MLM）进行编码

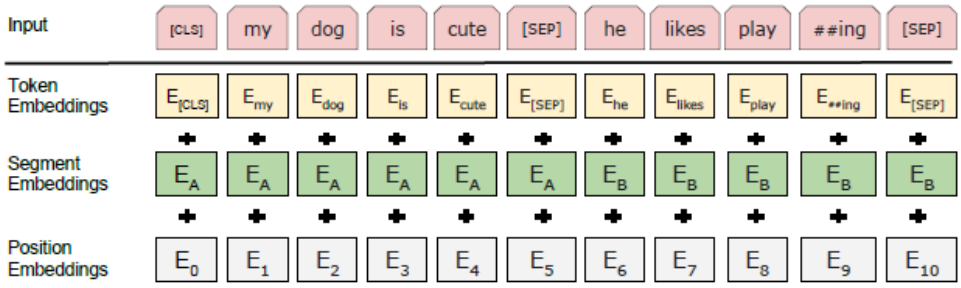

图 2：BERT 的输入表示由词元嵌入、位置嵌入和分段嵌入相加得到——图片引自 Devlin 等人的 BERT 原始论文，

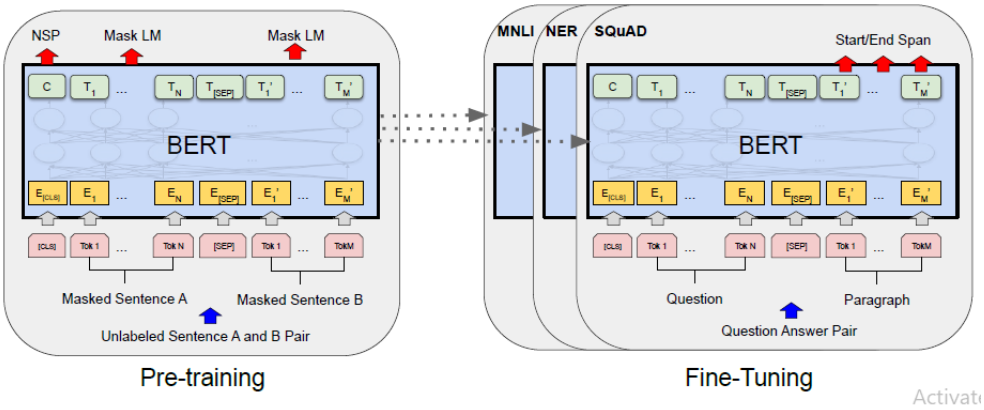

图 3：BERT 的训练阶段。除输出层外，预训练和微调采用相同的架构。微调期间，模型使用预训练模型参数进行初始化，然后针对当前任务进行微调。图片引自 BERT 原始论文。

左右两侧的上下文，并将其转换为词元表示。掩码语言模型会随机屏蔽部分输入词元，然后预测被屏蔽词元对应的词汇表 ID。实现 BERT 包括两个步骤：预训练和微调。在预训练阶段，模型使用大量未标注文本构成的语料库进行训练。在微调阶段，使用预训练阶段学到的参数初始化模型，随后针对特定下游任务进行微调。

BERT 凭借灵活的架构处理各种下游任务。它可以表示一个句子，也可以表示两个配对的句子 $(SentenceA,\ SentenceB)$。为了表示两个句子，BERT 会添加一个独特的词元 $[SEP]$。此外，每个序列都以一个特定的分类词元 $[CLS]$ 开头。我们将 $[CLS]$ 词元的最终状态用作分类任务的序列表示。

方法密度		多义性语境	感知	微调	OOV word 向量	资源意图-sity
独热编码	×	×	×	×	×	3
LSA	✓	×	×	×	×	3
word2vec	✓	×	×	×	×	1
GloVe	✓	×	×	×	×	1
fastText	✓	×	×	×	✓	2
ELMo	✓	✓	✓	✓	✓	4
BERT	✓	✓	✓	✓	✓	5

表 1：该表展示了不同词嵌入方法中某些属性的有无。对于内存和训练时间需求较低的模型，其资源消耗较低。

词元的输入表示由相应的词元嵌入、分段嵌入和位置嵌入相加得到。对于词元嵌入，BERT 使用 WordPiece嵌入 [49], 词元的分段嵌入表示该词元属于句子A还是句子B，而位置嵌入则包含该词元在输入序列中的位置。BERT 在语言建模方面非常实用，在几乎所有语言建模任务中的表现都优于其他模型。

3 现有词嵌入方法的属性

在进行外部评估之前，我们首先通过比较现有词表示方法的内在属性来评估这些方法。这些属性会直接或间接地影响这些方法在外在任务上的表现。我们从基本的词表示方法（例如，独热编码）入手，并考虑现有的最先进的词嵌入模型。我们研究这些模型在词表示的密度、多义性、上下文感知、微调、资源消耗以及表示词汇表外（OOV）词等方面的属性。

词嵌入模型生成的词表示密度较高，因此模型具有较高的内存效率。它还使模型能够编码与词对应的潜在特征，而不是编码单纯的统计信息。早期模型（如独热编码）会生成稀疏表示；其他模型（如潜在语义分析）则会生成稀疏表示，再将其投影到较低维度。另一方面，现代方法从学习低维稠密词向量开始。生成低维词向量使词嵌入模型能够编码词的隐含

属性，从而保留语料库中存在的语义和句法规律。例如，word2vec 会最大化彼此距离较近的词之间的对数概率。因此，具有相同上下文的词往往具有相似或更接近的词表示。

多义性使词嵌入模型能够针对不同词义，甚至同一词义，学习不同的词向量。一个词在不同语境中使用时，可能会表达不同的含义。词嵌入模型应能够追踪词语的上下文，为多义词生成词向量。传统模型（如潜在语义分析和独热编码）会为每个词提供单一、静态的词表示，而不考虑其词义。GloVe、word2vec 和 fastText 也会提供类似的静态词表示，不考虑多义性。另一方面，ELMo 和 BERT 会针对每种词义提供不同的词表示。这是可行的，因为这些模型会追踪词语的上下文。依赖上下文的词表示还能进一步提升 ELMo 和 BERT 在下游任务中的表现，因为词语的含义会影响其表示。以词语 书为例。它可以用作名词 或动词。前一种情况下，该词的表示应更接近笔记本 和钢笔 等词；后一种情况下，该词的表示则应更接近票等其他词。因此，多种依赖上下文的表示使 BERT 和 ELMo 等模型能够编码并有效利用某些词语的含义。

微调使模型能够针对任务调整其参数。词嵌入模型会生成词表示，可将其用作底层分类任务的特征。有些模型（如 GloVe 和 word2vec）会生成固定的词向量，而另一些模型（如 BERT 和 ELMo）则允许通过微调生成更适合任务的词表示。

未登录词是指模型在训练过程中未见过的词。如果模型以某种方式理解词的形态学，就能为未见过的词生成词表示。传统模型 word2vec 和 GloVe 不会利用子词信息，而是将每个词视为不可分割的整体；现代方法（如 fastText、ELMo 和 BERT）则能利用子词信息。fastText 将每个词视为多个 n 元语法的组合，因此能够表示未登录词。构成该词所需的 n 元语法可能已经存在，而该词的表示就是其所有 n 元语法表示之和。ELMo 使用字符嵌入计算词级嵌入，因此能够处理未登录词的表示。BERT 使用 WordPiece 算法将一个词拆分为 片段 ，并用嵌入表示每个 片段 。

虽然大多数研究都关注不同词嵌入方法在各类任务中的表现，但了解这些方法的资源消耗也很有必要。有时，与另一种模型相比，某个模型会消耗更多的时间和资源，但它们在某项任务上的表现可能并无显著差异。表1展示了不同词表示模型在内在属性方面的比较。

4 实验结果

4.1 数据集

我们针对4种不同的分类任务评估上述模型：垃圾邮件检测、激进语言检测、辱骂性语言检测，以及区分辱骂性语言和仇恨性语言。垃圾邮件检测和激进语言检测数据集各包含20万条推文（10万条负面推文和10万条正面推文）；辱骂性语言检测数据集包含2.5万条推文（1.25万条负面推文和1.25万条正面推文）；辱骂性语言与仇恨性语言数据集包含5000条推文（2500条辱骂性语言推文和2500条仇恨性语言推文）。数据集的详细信息见表2。

我们针对垃圾邮件检测和激进语言检测数据集，在多种设置下开展了不同的实验。例如，我们使用这些数据集的不同子集（25K、50K、100K、150K 和 200K 条推文）评估了模型。我们还通过采用正面类别和负面类别的不同占比来比较模型，以考察数据偏斜对各个模型的影响。

数据集	说明
Spam-N	垃圾邮件检测数据集的一个子集，包含 N 条推文。它可能是平衡的（1:1），也可能是不平衡的（n:1），其中每条正面垃圾推文对应 n 条负面普通推文。每条正面垃圾推文。
Radical-N	激进语言数据集的一个子集，包含 N 条推文。它可以是均衡的（1:1），也可以是不均衡的（n:1），其中每条每条正面激进推文。
辱骂性语言	该数据集包含 2.5 万条推文。该数据集在正面推文之间均衡分布和负面推文之间均衡分布。
辱骂性与仇恨性	该数据集包含 5K 条推文。数据集分布均衡，辱骂性语言推文与仇恨性语言推文数量相同。

表2：各数据集的简要说明。

4.2 实验模型设置

我们的实验模型由一个输入嵌入层、两个双向长短期记忆层（分别包含 32 个和 64 个单元）、一个包含 128 个单元的全连接层，以及一个 Sigmoid 输出层组成。

我们使用 TensorFlow 框架 [50]训练模型。我们使用 Gensim 库 [51] 在语料库上训练 word2vec 和 fastText 向量。我们还使用 GloVe 的 TensorFlow 实现来训练向量，而没有使用以 C 语言 编写的原始代码。对于 ELMO 和 BERT，我们使用 tensorflow-hub 获取依赖上下文的预训练向量。在实验中，我们使用了 BERT 模型的 BERT-base（L=12，H=768）实现。

4.3 评估训练后的词嵌入

在语料库上训练词向量时，需要设置许多超参数，例如窗口大小和向量维度。为超参数选择合适的值，会显著影响计算得到的词向量。我们已在语料库上训练了 word2vec（Skip-gram 和连续词袋模型）、GloVe 和 fastText 的词向量。随后，我们将计算得到的词向量用于分类任务，并报告结果。由于语料库规模较小，相较于使用预训练向量，从头开始训练 BERT 和 ELMo 的向量效率较低，因此我们没有从头开始训练它们的向量。此外，这种做法还会大量消耗资源和时间；正如我们将在下一节看到的，使用它们的预训练向量更好。

4.3.1 窗口大小：

窗口大小是指训练词嵌入时，我们考虑中心词两侧的词数。在 Skip-gram、连续词袋模型和 fastText 中，窗口大小用于最大化中心词与上下文词（窗口中出现的词）之间的对数概率；而在 GloVe 中，我们使用窗口大小来构建词共现矩阵。我们选择了三种不同的窗口大小，并在表 3 中报告了每个数据集对应的结果。

虽然 GloVe 在较大的窗口大小下始终表现更好，但其他模型的表现则往往起伏不定。尽管所有模型在不同窗口大小下的性能差异并不显著，但我们可以放心地说，较大的窗口大小能让模型捕获更多信息。我们应谨记，较大的窗口大小需要更多计算资源和内存资源。此外，窗口大小过大时，模型可能会捕获无关信息。例如，一个词可能与 20 个词之后出现的另一个词无关，也不受其影响。

	Skipgram			CBOW			GloVe			fasttext		
Datasets	w=2	w=5	w=10	w=2	w=5	w=10	w=2	w=5	w=10	w=2	w=5	w=10
Spam-100K	0.823	0.820	**0.828**	0.828	0.84	**0.841**	0.849	0.848	**0.85**	**0.854**	0.844	0.849
Spam-100K (10:1)	0.510	0.517	**0.529**	**0.57**	0.56	0.569	0.597	0.597	**0.60**	**0.593**	0.567	0.56
Spam-30K	0.808	**0.821**	0.818	**0.814**	0.804	0.798	0.82	0.82	**0.823**	0.809	**0.818**	0.80
Radical-100K	**0.999**	0.998	0.998	**0.999**	0.998	0.998	0.995	0.995	**0.997**	0.997	0.997	**0.998**
Radical-30K	0.997	0.997	0.997	**0.998**	0.996	0.996	0,994	0.995	**0.996**	0.997	0.996	**0.998**
Abusive language	0.993	**0.995**	0.994	**0.992**	0.991	0.991	0.980	0.982	**0.985**	0.993	0.993	**0.994**
Abusive vs. hateful	0.733	0.741	**0.767**	0.718	0.709	**0.720**	0.768	0.767	**0.769**	0.745	**0.771**	0.741

Table 3: The table shows the performance of different models, trained on different window sizes, on external classification tasks. All models have been trained with 300 dimension vectors on the same task's training data. We have reported f1 score for the positive class for every task. All datasets, except the ones we have mentioned the ratio with, are balanced.

-	Skipgram			CBOW			GloVe			fasttext		
Datasets	50d	200d	300d	50d	200d	300d	50d	200d	300d	50d	200d	300d
Spam-200K	0.853	0.859	0.856	0.859	0.860	0.862	0.821	0.855	0.86	0.863	0.866	0.866
Spam-100K	0.833	0.835	0.820	0.84	0.84	0.84	0.843	0.847	0.848	0.854	0.856	0.844
Spam-30K	0.805	0.80	0.824	0.811	0.81	0.804	0.811	0.814	0.82	0.808	0.818	0.818
Radical-100K	0.997	0.997	0.998	0.998	0.998	0.998	0.995	0.994	0.995	0.996	0.996	0.997
Radical-30K	0.995	0.995	0.997	0.996	0.997	0.996	0.995	0.994	0.995	0.997	0.996	0.996
Abusive language	0.995	0.995	0.995	0.990	0.990	0.991	0.978	0.979	0.982	0.991	0.992	0.993
Abusive vs. hateful	0.750	0735	0.741	0.721	0.739	0.709	0.722	0.755	0.767	0.763	0.78	0.771

Table 4: The table shows the effect of trained word vector dimension on extrinsic classification tasks. All models have been trained with a window size of 5.

4.3.2 嵌入维度

我们还训练了不同维度的词向量，以考察其对内在分类任务的影响。对于大多数此类任务，高维向量和低维向量的表现相近；但我们发现，对于数据量较大的任务（例如 Spam-200K），高维向量表现更好。高维向量能够编码更多数据，并以更稀疏的方式表示词语，因此在大多数外在任务中表现更佳。相反，如果数据较少，在某些情况下（例如辱骂性与仇恨性），较低维度的向量能够更好地编码信息。

总的来说，高维向量在几乎所有任务中的表现可能相近，甚至更好，但训练它们也需要更多内存和时间。多个数据集的 F1 分数已列于表 4。

4.4 预训练词嵌入的评估

我们还将预训练向量用于外部分类任务，并报告了结果。除了 word2vec、GloVe 和 fastText 外，本节的评估还纳入了 BERT 和 ELMo。word2vec、GloVe 和 fastText 的预训练词向量是静态的，因为它们与上下文无关。与此同时，BERT 和 ELMo 提供依赖上下文的词向量，因此在训练分类模型时，必须针对每个句子提取向量。我们将先比较这些模型在若干分类任务上的表现，随后考察一些因素对这些模型的影响。在实验中，我们使用了 BERT-base，其向量维度为 768；ELMo 的向量维度为 1024；其他所有模型的词表示维度均为 300。

表5展示了所有模型在各类分类任务上的 F1 分数。可以看出，所有模型在相对简单的数据集（例如激进语言和辱骂性语言）上表现都很好，而许多模型在其他数据集上的表现则不尽如人意。区分辱骂性推文和仇恨推文是一项艰巨的任务，因为两者的语言结构相似；即便如此，与其他任何模型相比，BERT 仍将该数据集的 F1 分数提高了近 5%。

数据集	word2vec	GloVe	fastText	ELMo	BERT
Spam-100K	0.822	0.818	0.836	0.969	0.979
Radical-100K	0.995	0.995	0.998	0.999	1.0
辱骂性语言	0.982	0.982	0.993	0.997	1.0
辱骂性与仇恨性	0.771	0.782	0.783	0.761	0.833

表5：此表展示了各种预训练词嵌入模型在不同分类任务上的表现。

4.4.1 数据集规模

在本部分中，我们评估数据规模对预训练词嵌入性能的影响。我们从数据集中选取多个子集来评估模型，并考察其分类 F1 分数。尽管数据越多、性能越好这一点或许显而易见，但在某些情况下，可能无法获得海量数据。此外，了解分类数据增加对模型性能的影响程度也至关重要。

所有模型均使用数十亿个词进行了预训练。BERT 和 ELMo 具备针对特定任务进行微调的能力，而其他模型则不具备。这两项特性——微调和上下文依赖性——的影响十分显著，因为这两个模型在外部分类任务上的表现异常出色。此外，分类数据集较小时，BERT 几乎不受影响，而其他模型的性能损失则相对较大。word2vec 受到的影响最大：在垃圾信息分类任务中，当数据集包含 25K 条推文而非 200K 条时，其 F1 分数下降了 6% 以上。

分类结果进一步证明，这两个数据集的文本结构截然不同。激进推文结构清晰，书写也更加正式，因此所有模型在对其进行分类时都表现出色。此外，激进推文包含一些特定词语，这些词语在普通推文中很少出现，进一步使其文本结构与普通推文有所区别。另一方面，垃圾推文与普通推文更为相似，而且其中大多数包含俚语和缩写，而许多模型都无法表示这些内容。

另一个进一步助益于 BERT 的因素，是其核心机制：利用 WordPiece 算法将一个词拆分成多个片段。这种拆分有助于 BERT 为这些词提供依赖上下文的向量，否则它们可能属于未登录词。表 6 展示了结果。

数据集	Spam				激进			
	50K	100K	150K	200K	50K	100K	150K	200K
word2vec	0.81	0.82	0.83	0.85	0.99	0.99	0.99	1.0
GloVe	0.81	0.82	0.83	0.84	0.99	0.99	0.99	1.0
fastText	0.81	0.84	0.84	0.85	1.0	1.0	1.0	1.0
ELMo	0.96	0.97	0.97	0.98	1.0	1.0	1.0	1.0
BERT	0.98	0.98	0.98	0.98	1.0	1.0	1.0	1.0

表 6：此表展示了数据集规模增加或减少对词嵌入模型性能的影响。BERT向量有 768 个维度，ELMo 使用 1024 维度向量，而其他所有模型均使用 300 维预训练向量。

4.4.2 不平衡数据

深度学习模型通常容易过拟合更普遍的类别。本节根据各词嵌入模型在不平衡数据上的表现对其进行比较。尽管所有模型在偏斜数据上的表现都较差，但 BERT 和 ELMo 在最复杂的任务中仍保持着高于平均水平的表现。我们再次观察到，对于较简单的数据集，数据偏斜的影响微乎其微；而对于复杂任务（例如垃圾信息分类），其影响则更加显著。表 7 展示了各数据集的分类 F1 分数。

数据集	Spam				激进			
	1:1	2:1	5:1	10:1	1:1	2:1	5:1	10:1
word2vec	0.822	0.759	0.646	0.539	0.995	0.994	0.990	0.986
GloVe	0.818	0.763	0.647	0.543	0.995	0.994	0.992	0.990
fastText	0.836	0.771	0.666	0.613	0.998	0.997	0.996	0.996
ELMo	0.969	0946	0.945	0.859	0.999	0.998	0.997	0.994
BERT	0.979	0.965	0.942	0.922	1.0	1.0	1.0	1.0

表 7：平衡和不平衡数据集上的 F1 分数。

4.4.3 分类数据预处理

虽然预处理是训练深度学习分类模型不可或缺的一部分，但我们希望考察对分类数据进行完全清理和部分清理对分类结果的影响。

在第一种情况下，我们会彻底清理数据，从推文中删除网址、话题标签、符号、提及和停用词，并对词进行词形还原。而在后一种情况下，我们只从数据中删除网址和话题标签，不对词进行词形还原。

数据集	Spam-100K		Radical-100K	
	F-Cl	P-Cl	F-Cl	P-Cl
word2vec	0.791	0.822	0.988	0.995
GloVe	0.792	0.818	0.989	0.995
fastText	0.788	0.836	0.988	0.998
ELMo	0.801	0.969	0.993	0.999
BERT	0.822	0.979	0.995	1.0

表 8：分类时部分清洗和完全清洗数据的 F1 分数。F-Cl 表示完全清洗，P-Cl 表示部分清洗。

部分清理后的数据在所有情况下都能取得更好的结果。这表明停用词和实际动词形式会影响分类结果。我们还观察到

在部分清理数据时，fastText、ELMo 和 BERT 的性能提升更为显著。这种性能提升的原因在于这些模型使用了子词信息，使其能够表示未登录词并利用词的形态学特征。结果如表 8 所示

4.5 多类别分类

到目前为止，我们已将推文分为两个类别：正面和负面。在某些情况下，我们需要将数据分为多个类别。本文评估了预训练词嵌入在多类别分类中的表现。

我们构建了一个包含五个类别的新数据集，每个类别有 2500 条推文。我们还调整了分类模型，将 Sigmoid 输出层替换为包含五个单元的 softmax 输出层，以适应多类别分类。完整的对比结果见表 9。

我们给出了完全清理和部分清理后的数据的结果。使用完全清理后的数据时，BERT 和 GloVe 之间略有差异；而使用部分清理后的数据时，差异则明显得多。这进一步印证了我们此前的观点：完全清理数据会导致信息丢失。

数据集	多类别	
	F-Cl	P-Cl
word2vec	0.730	0.779
GloVe	0.757	0.798
fastText	0.714	0.766
ELMo	0.745	0.856
BERT	0.777	0.873

表 9：多类别分类的宏平均 F1 分数。该数据集包含 5 个类别，每个类别有 2500 条推文。F-Cl 表示完全清洗，P-Cl 表示部分清洗。

4.6 预训练词嵌入与训练词嵌入的比较

我们评估了与上下文无关的训练词嵌入模型。我们还在多个外部分类任务上比较了所有主要的预训练词嵌入模型。现在，我们将比较由这些与上下文无关模型生成的预训练词嵌入，以及我们

	word2vec		GloVe		fastText	
数据集	Pre-训练	训练	Pre-训练	训练	Pre-训练	训练
Spam-200K	0.849	0.856	0.844	0.86	0.854	0.866
Spam-100K	0.822	0.82	0.818	0.848	0.836	0.844
Spam-30K	0.79	0.821	0.788	0.82	0.816	0.818
Radical-100k	0.995	0.998	0.995	0.995	0.998	0.997
Radical-30K	0.995	0.997	0.995	0.995	0.998	0.996
辱骂性语言	0.982	0.995	0.982	0.982	0.993	0.993
辱骂性与仇恨性	0.771	0.741	0.782	0.767	0.783	0.771

表 10：不同数据集上预训练词嵌入和训练词嵌入的 F1 分数。

在语料库上训练的词嵌入。我们使用了尽可能相近的超参数值。我们使用 300 维向量，上下文窗口大小为 5；对于 word2vec，我们使用 Skip-gram模型，负采样值为 15。

对于大多数任务，我们从头开始训练得到的词表示优于预训练词嵌入。这可能是因为用于训练这些词嵌入的语料库与当前任务并不相关。另一个原因可能是，语料库中包含一些在预训练嵌入中没有对应嵌入的词。不过，如果分类数据集较小（例如，辱骂性与仇恨性），使用预训练向量会更好，因为小型文本语料库无法生成高质量的词表示。考察垃圾信息数据集的不同子集后，数据规模的影响就更加明显。预训练向量与训练得到的向量之间的性能差异，在较小的子集（3 万条推文）中更大，而在较大的子集（10 万和 20 万条推文）中则显著缩小。详情见表 10。

5 结论

本文评估了现有词嵌入算法在外部分类任务上的表现。我们还介绍了这些模型各自的工作原理，并阐述了它们如何编码词语之间的关系。我们说明了优秀词嵌入方法应具备的特性，并讨论了某些模型中这些特性的存在与缺失。我们还展示了这些特性如何影响词嵌入模型生成的词嵌入。此外，我们还展示了窗口大小、嵌入维度等参数对词嵌入质量的影响。我们还比较了预训练词嵌入和训练词嵌入，并分析了它们对分类任务的影响。此外，我们还探究了哪种算法在多类别分类中表现更好。

尽管在一篇论文中研究所有词嵌入算法及其特性

很困难，但我们涵盖了从不同类别中选出的最重要方法。为简化比较，我们将模型分为两组：传统模型和神经模型，并发现神经词嵌入模型相比传统方法具有诸多优势。

我们观察到，尽管 BERT 在几乎所有任务中的表现都优于其他词嵌入方法，但在某些分类任务（例如“辱骂性与仇恨性”）中，差异微乎其微。因此，考虑到 BERT 对资源的需求较高，简单的模型（例如 word2vec 和 GloVe）反而能取得更好的结果。另一方面，ELMo 在大多数任务中的表现与 BERT 相近，因为它会针对具体任务调整参数。

我们还观察到，底层分类数据的结构起着至关重要的作用。有些数据集（例如激进主义数据集）可能很容易分类，因为其中的文本组合差异很大；而在其他数据集（例如“辱骂性与仇恨性”）中，两个类别遵循的文本模式几乎相同，因此难以区分。在选择词嵌入模型之前，我们应分析底层分类数据，并考察不同文本类别的文本结构。当各类别遵循不同的文本模式时，简单模型（例如 word2vec 和 GloVe）的表现与复杂模型（例如 BERT）相近。

数据预处理虽然是分类任务中不可或缺的一部分，但如果处理过度，可能会对结果产生负面影响。我们观察到，文本原始的形态学特征提供了关键信息；如果预处理不够谨慎，这些信息可能会丢失。上述观点进一步凸显了使用子词信息的重要性。将词视为不可分割实体的方法可能会遇到未登录词，进而导致未知词向量的数量增加。所有这些未知词向量都会造成信息损失。另一方面，使用子词信息或字符级信息的方法能够处理未登录词，并有效地表示词的实际形态。因此，BERT、ELMo 和 fastText 等模型比其前代模型表现更好。

参考文献

[1] Xiaodong Liu、Yelong Shen、Kevin Duh 等人。《用于机器阅读理解的随机答案网络》。载于：*arXiv* 预印本 *arXiv:1712.03556*（2017）。[2] Caiming Xiong、Victor Zhong 和 Richard Socher。《用于问答的动态协同注意力网络》。载于：*arXiv* 预印本 *arXiv:1611.01604*（2016）。[3] Jie Zhou 和 Wei Xu。《使用循环神经网络端到端学习语义角色标注》。载于：计算语言学协会第*53*届年会暨自然语言处理国际联合会议第*7*届会议论文集（第*1*卷：长篇论文）。2015年，第1127–1137页。

[4] Luheng He、Kenton Lee、Mike Lewis 等人。《深度语义角色标注：哪些方法有效，下一步是什么》。载于：计算语言学协会第*55*届年会论文集（第*1*卷：长篇论文）。2017年，第473–483页。[5] William Foland 和 James H. Martin。《使用卷积神经网络进行基于依存关系的语义角色标注》。载于：第四届词汇语义学与计算语义学联合会议论文集。2015年，第279–288页。[6] Qian Chen、Xiaodan Zhu、Zhenhua Ling 等人。《用于自然语言推理的增强型长短期记忆网络》。载于：*arXiv* 预印本 *arXiv:1609.06038*（2016）。[7] Biao Zhang、Deyi Xiong、Jinsong Su 等人。《用于神经机器翻译的上下文感知循环编码器》。载于：*IEEE/ACM*音频、语音与语言处理汇刊 25.12（2017），第2424–2432页。[8] Aliaksei Severyn 和 Alessandro Moschitti。《使用深度卷积神经网络进行 Twitter 情感分析》。载于：第*38*届国际 *ACM SIGIR* 信息检索研究与发展会议论文集。2015年，第959–962页。[9] Chi Sun、Luyao Huang 和 Xipeng Qiu。《通过构造辅助句，利用 BERT 进行基于方面的情感分析》。载于：*arXiv* 预印本 *arXiv:1903.09588*（2019）。[10] Yue Geng、Zheng Lin、Peng Fu 等人。《社交媒体上的谣言检测：一种采用自注意力机制的多视图模型》。载于：国际计算科学会议。施普林格。2019年，第339–352页。[11] Yunfei Long、Qin Lu、Rong Xiang 等人。《通过多视角说话人画像检测假新闻》。载于：第八届自然语言处理国际联合会议论文集（第*2*卷：短篇论文）。2017年，第252–256页。[12] Thomas K Landauer、Peter W Foltz 和 Darrell Laham。《潜在语义分析简介》。载于：话语过程 25.2-3（1998），第259–284页。[13] David M Blei、Andrew Y Ng 和 Michael I Jordan。《潜在狄利克雷分配》。载于：机器学习研究杂志 3.Jan（2003），第993–1022页。[14] Yoshua Bengio、R´ejean Ducharme、Pascal Vincent 等人。《神经概率语言模型》。载于：机器学习研究杂志 3.Feb（2003），第1137–1155页。[15] Andriy Mnih 和 Koray Kavukcuoglu。《利用噪声对比估计高效学习词嵌入》。载于：神经信息处理系统进展。2013年，第2265–2273页。[16] Matthew E Peters、Mark Neumann、Mohit Iyyer 等人。《深度上下文化词表示》。载于：*arXiv* 预印本 *arXiv:1802.05365*（2018）。

[17] Jacob Devlin、Ming-Wei Chang、Kenton Lee 等。“BERT：用于语言理解的深度双向 Transformer 预训练”。载于：*arXiv* 预印本 *arXiv:1810.04805* （2018）。[18] Amir Bakarov。“词嵌入评估方法综述”。载于：*arXiv* 预印本 *arXiv:1801.09536* （2018）。[19] Marco Baroni、Georgiana Dinu 和 Germ´an Kruszewski。“不要计数，要预测！对基于上下文计数与基于上下文预测的语义向量进行系统比较”。载于：计算语言学协会第*52*届年会论文集（第*1*卷：长篇论文）。2014年，第238–247页。[20] Bin Wang、Angela Wang、Fenxiao Chen 等。“词嵌入模型评估：方法与实验结果”。载于：《*APSIPA* 信号与信息处理汇刊》8（2019）。[21] Tobias Schnabel、Igor Labutov、David Mimno 等。“无监督词嵌入的评估方法”。载于：*2015*年自然语言处理实证方法会议论文集。2015年，第298–307页。[22] Xiao Yang、Craig Macdonald 和 Iadh Ounis。“在 Twitter 选举分类中使用词嵌入”。载于：《信息检索期刊》21.2-3（2018），第183–207页。[23] Billy Chiu、Anna Korhonen 和 Sampo Pyysalo。“词向量的内在评估无法预测外在表现”。载于：首届自然语言处理向量空间表示评估研讨会论文集。2016年，第1–6页。[24] Marwa Naili、Anja Habacha Chaibi 和 Henda Hajjami Ben Ghezala。“主题分割中词嵌入方法的比较研究”。载于：《计算机科学 *Procedia*》112（2017），第340–349页。[25] Bhuwan Dhingra、Hanxiao Liu、Ruslan Salakhutdinov 等。“用于阅读理解的词嵌入比较研究”。载于：*arXiv* 预印本 *arXiv:1703.00993* （2017）。[26] Dima Suleiman 和 Arafat Awajan。“词嵌入模型及其在阿拉伯语应用中的使用比较研究”。载于：*2018*年阿拉伯国际信息技术会议（*ACIT*）。IEEE。2018年，第1–7页。[27] Kevin Lund 和 Curt Burgess。“从词汇共现生成高维语义空间”。载于：《行为研究方法、仪器与计算机》28.2（1996），第203–208页。 ISSN: 1532-5970。 DOI: 10.3758/BF03204766。 URL: https://doi.org/10.3758/BF03204766。[28] Douglas LT Rohde、Laura M Gonnerman 和 David C Plaut。“基于词汇共现的改进型语义相似度模型”。载于：《美国计算机协会通讯》8.627-633（2006），第116页。

[29] R´emi Lebret 和 Ronan Collobert。“通过 Hellinger 主成分分析获得词嵌入”。载于：欧洲计算语言学协会分会第*14*届会议论文集。瑞典哥德堡：计算语言学协会，2014年4月，第482–490页。 DOI: 10.3115/v1/E14-1051。 URL: https://www.aclweb.org/anthology/ E14-1051。[30] Edgar Altszyler 、Mariano Sigman、Sidarta Ribeiro 等。“小型语料库中潜在语义分析与 word2vec 嵌入的比较研究：梦境数据库案例研究”。载于：*arXiv* 预印本 *arXiv:1610.01520* （2016）。[31] Tom´aˇs Mikolov、Martin Karafi´at、Luk´aˇs Burget 等。“基于循环神经网络的语言模型”。载于：国际语音通信协会第十一届年会。2010年。[32] Tom´aˇs Mikolov、Stefan Kombrink、Luk´aˇs Burget 等。“循环神经网络语言模型的扩展”。载于：*2011*年*IEEE*国际声学、语音与信号处理会议（*ICASSP*）。IEEE。2011年，第5528–5531页。[33] Ronan Collobert 和 Jason Weston。“自然语言处理的统一架构：采用多任务学习的深度神经网络”。载于：第*25*届国际机器学习会议论文集。2008年，第160–167页。[34] T Mikolov、K Chen、GS Corrado 等。“向量空间中词表示的高效表示”。载于：国际表征学习会议（*ICLR*）研讨会论文集。2013年。[35] Joseph Turian、Lev Ratinov 和 Yoshua Bengio。“词表示：一种简单通用的半监督学习方法”。载于：计算语言学协会第*48*届年会论文集。计算语言学协会。2010年，第384–394页。[36] Tomas Mikolov、Ilya Sutskever、Kai Chen 等。“词和短语的分布式表示及其组合性”。载于：神经信息处理系统进展。2013年，第3111–3119页。[37] Jeffrey Pennington、Richard Socher 和 Christopher D Manning。“GloVe：用于词表示的全局向量”。载于：*2014*年自然语言处理实证方法会议（*EMNLP*）论文集。2014年，第1532–1543页。[38] Piotr Bojanowski、Edouard Grave、Armand Joulin 等。“利用子词信息丰富词向量”。载于：计算语言学协会会刊5（2017），第135–146页。[39] Radu Soricut 和 Franz Josef Och。“使用词嵌入进行无监督形态归纳”。载于：*2015*年北美计算语言学协会分会会议论文集：人类语言技术。2015年，第1627–1637页。

[40] Aleksandra Piktus、Necati Bora Edizel、Piotr Bojanowski 等。“不受拼写错误影响的词嵌入”。载于：*2019*年北美计算语言学协会分会年会论文集：人类语言技术，第*1*卷（长篇与短篇论文）。2019年，第3226–3234页。[41] Arvind Neelakantan、Jeevan Shankar、Alexandre Passos 等。“在向量空间中对每个词的多个嵌入进行高效的非参数估计”。载于：*arXiv*预印本 *arXiv:1504.06654*（2015年）。[42] Eric H Huang、Richard Socher、Christopher D Manning 等。“通过全局上下文和多个词原型改进词表示”。载于：计算语言学协会第*50*届年会论文集：长篇论文第*1*卷。计算语言学协会。2012年，第873–882页。[43] Rafal Jozefowicz、Oriol Vinyals、Mike Schuster 等。“探索语言建模的极限”。载于：*arXiv*预印本 *arXiv:1602.02410*（2016年）。[44] G´abor Melis、Chris Dyer 和 Phil Blunsom。“神经语言模型评估的最新进展”。载于：*arXiv*预印本 *arXiv:1707.05589*（2017年）。[45] Matthew E Peters、Waleed Ammar、Chandra Bhagavatula 等。“使用双向语言模型进行半监督序列标注”。载于：*arXiv*预印本 *arXiv:1705.00108*（2017年）。[46] Alec Radford、Karthik Narasimhan、Tim Salimans 等。“通过生成式预训练改进语言理解”。载于：网址 *https://s3-us- west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*（2018年）。[47] Alec Radford、Jeffrey Wu、Rewon Child 等。“语言模型是无监督多任务学习器”。载于：*OpenAI* 博客1.8（2019年），第9页。[48] Ashish Vaswani、Noam Shazeer、Niki Parmar 等。“注意力即一切”。载于：神经信息处理系统进展。2017年，第5998–6008页。[49] Yonghui Wu、Mike Schuster、Zhifeng Chen 等。“谷歌的神经机器翻译系统：弥合人类与机器翻译之间的差距”。载于：*arXiv*预印本 *arXiv:1609.08144*（2016年）。[50] Martı — n Abadi、Paul Barham、Jianmin Chen 等。“TensorFlow：大规模机器学习系统”。载于：*OSDI*。第16卷。2016年。美国佐治亚州萨凡纳。2016年，第265–283页。[51] Radim ˇReh ˚u ˇrek 和 Petr Sojka。“用于大型语料库主题建模的软件框架”。英语。载于：*LREC 2010*年*NLP*框架新挑战研讨会论文集。http: / / is . muni . cz / publication/884893/en。马耳他瓦莱塔：ELRA，2010年5月，第45–50页。