Tsetlin 机　　嵌入：使用逻辑表达式表示词语　　s

Bimal Bhattarai 、**Ole-Christoffer Granmo**、**Lei Jiao** 、**Rohan Yadav** 和**Jivitesh Sharma**人工智能研究中心（CAIR）、阿格德尔大学、挪威{bimal.bhattarai、ole.granmo、lei.jiao、rohan.yadav、jivitesh.sharma}@uia.no

arXiv:2301.00709v1 [cs.CL] 2 Jan 2023

摘要

将词语嵌入向量空间是当前最先进自然语言处理（NLP）中的基础第一步。典型的 NLP 解决方案采用预定义的向量表示，通过将相似词语置于向量空间中的相近位置来提升泛化能力。例如，Word2Vec 是一种自监督预测模型，利用神经网络捕捉词语的上下文。类似地，GloVe 是一种流行的无监督模型，融合了整个语料库的词共现统计。这类词嵌入显著推动了包括情感分析、文档分类和机器翻译在内的重要 NLP 任务。然而，这些嵌入是稠密的浮点向量，计算成本高且难以解释。本文则提出使用少量通过命题逻辑关联的定义性词语来表示词语的语义。为生成这种逻辑嵌入，我们引入了一种基于 Tsetlin 机的自编码器，以自监督方式学习逻辑子句。这些子句由“black”“cup”和“hot”等上下文词组成，用于定义“coffee”等其他词语，因此人类可以理解。我们在多个内在和外在基准测试上评估了这种嵌入方法，在六项分类任务中超过了 GloVe。此外，我们利用训练过程中获得的逻辑表示，研究了嵌入的可解释性。我们还将向量空间中的词簇可视化，展示逻辑嵌入如何将相似词语置于相近位置。[1]

1 引言

自然语言处理（NLP）的成功有赖于词语、句子和文档表示方面的进步。通过捕捉词语的语义和相似性，这类表示能够提升下游任务的性能 [Borgeaud 等人， 2022], 包括聚类、主题建模 [Angelov，2020], 搜索和文本挖掘 [Huang 等人， 2020]。

虽然传统的词袋编码简单直接，但它不会考虑文档中词语的位置、语义和上下文。分布式词语表示 [Bengio 等人, 2000; Bojanowski等人，2017] 通过将词语编码为低维向量（称为嵌入）弥补了这一不足。其目的是在向量空间中让相似或上下文相关的词语彼此靠近。学习词嵌入的算法有很多。当代的自监督技术，如 Word2Vec [Mikolov等人， 2013]，FastText [Bojanowski 等人， 2017]，和 GloVe [Pennington等人，2014] 展示了如何利用海量训练数据，根据词共现构建嵌入。引入依赖上下文的嵌入后，更复杂的语言模型 BERT [Devlin 等人， 2019] 和 ELMo [Peters 等人，2018] 如今在下游任务中表现出色 [Reimers和Gurevych，2019]。然而，它们需要大量计算能力 [Schwartz等人，2020]。

上述方法将单词表示为稠密的浮点向量。例如，Word2Vec 通常会为每个单词构建一个 300 维向量。这些向量的规模和稠密性使其计算成本高昂，也难以解释。以单词“queen”为例，用 300 个浮点数来表示它，似乎效率不高；相比之下，《牛津英语词典》对这个词的定义是：“独立国家的女性统治者，尤其是凭出生权继承这一职位的人。”从这个角度看，直接根据单词而不是任意浮点数创建嵌入似乎更有优势。这样的可解释嵌入可以用少数几个定义性词语来捕捉一个单词的多重含义，从而简化计算和解释。

本文提出了一种基于 Tsetlin 机器 (TM)[Granmo, 2018]的自编码器，用于创建可解释的嵌入。该自编码器利用上下文词构建命题逻辑表达式，以识别每个目标词。例如，术语“咖啡”可以用“一个”“热”“杯子”“桌子”和“黑色”来表示。通过这种方式，TM 从海量文本语料库中构建上下文表示，以刻画每个词的语义。与基于神经网络的嵌入不同，逻辑 TM 嵌入是稀疏的。嵌入空间由例如 500 个真值组成，

[1] Tsetlin机器自编码器和逻辑词嵌入的实现可在此处获取: https://github.com/cair/tmu。

其中每个真值都是一个关于词语的逻辑表达式。在上下文表示中，每个目标词关联的此类表达式不到百分之十。尽管这种表示具有稀疏性且边界明确，但其性能仍可与基于神经网络的嵌入相媲美。

我们的工作贡献总结如下：

- 我们提出了首个基于TM的自编码器，以自监督的方式学习高效编码。
- 我们提出了基于TM的词嵌入，能够从未标注数据中构建人类可理解的上下文表示。
- 我们在多个内在和外在基准测试上，将我们的嵌入与最先进的方法进行比较；在六项下游分类任务中，我们的表现优于GloVe。

2 相关工作

大多数自监督嵌入方法都基于分布假说生成稠密的词表示 [Harris, 1954], 该假说指出，在相同上下文中出现的词很可能具有相似的含义。Word2Vec [Mikolov 等人, 2013] 是最著名的模型之一。它利用神经网络，根据词共现构建嵌入，并采用隐藏层输出权重。另一方面，GloVe [Pennington 等人, 2014], 通过对词共现矩阵进行因子分解来生成嵌入。类似地，Dhillon 等人, [使用典型相关分析（CCA）来嵌入词，以最大化上下文相关性。在Levy 等人, [2015], 展示了精确的、基于因子分解的奇异值分解（SVD）如何与神经嵌入相媲美。然而，所有这些方法都难以训练，因为它们需要针对特定应用调整算法和超参数 [Lample 等人, 2016],这限制了它们更广泛的适用性。

在词嵌入的基础上，若干研究聚焦于句子嵌入 [Arora 等, 2017; Logeswaran和Lee, 2018]。句子嵌入领域的最新进展包括监督数据推断 [Reimers和Gurevych, 2019], 多任务学习 [Cer 等, 2018], 对比学习 [Zhang 等, 2020], 以及预训练大型语言模型 [Li 等, 2020]。然而，大多数句子嵌入技术忽视了相似度任务等内在评估，而主要关注涉及下游性能的外在评估。最新的嵌入构建模块源自 Transformer 方法 [Vaswani 等, 2017

]。Transformer模型通过使用多层自注意力层来感知上下文。例如，BERT [Kenton和Toutanova, 2019], 采用Transformer架构进行大规模自监督训练，因此能够生成文本嵌入。其他嵌入模型则使用对比损失函数，对正负文本对进行监督微调 [Wang 等, 2021]。尽管文本嵌入模型种类繁多，但它们都有三个主要缺点：i）训练时计算开销大；ii）模型本身复杂，因为它们需要利用大量数据来调整海量参数进行训练；

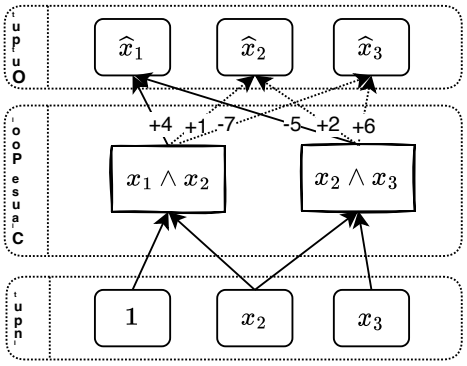

图 1：Tsetlin机器自编码器。在此示意图中，将 x_1 替换为值 1 进行掩码，以推断$\hat{x}_1$。

iii）这些模型生成的嵌入不容易被人类理解。

为提高可解释性，Faruqui等提出了“稀疏过完备词向量”（SPOWV），通过字典学习为词嵌入创建稀疏的非负投影 [Faruqui等，2015]。类似地，稀疏可解释神经嵌入（SPINE）采用 k-稀疏去噪自编码器生成稀疏嵌入 [Subramanian 等，2018]。然而，这些方法无法区分多个依赖上下文的词义。为解决这一问题，另一研究方向聚焦于组合来自 Word2Vec 和 GloVe 的稠密向量线性组合 [Arora 等，2018]。然而，线性假设并不适用于真实世界数据，因此产生的线性系数难以理解 [Mu等，2017]。

我们在此提出的逻辑嵌入方法与朴素贝叶斯词义归纳和主题建模最为相关 [Charniak等，2013；Lau 等，2014]。该方法通过将文档中目标词的每个实例都视为伪文档，从局部上下文中学习词语含义。然而，由于需要为每个目标词训练一个单独的主题，该方法无法扩展。相比之下，我们的方法具有可扩展性，并能构建捕捉词语组合的非线性（非朴素）逻辑嵌入。为构建逻辑嵌入，我们提出了一种基于 TM 的新型人类可解释算法，用逻辑规则描述上下文。近期，TM 在许多自然语言处理任务中的表现可与其他深度学习技术相媲美，包括新颖性检测 [Bhattarai 等，2022a], 情感分析 [Yadav 等, 2021], 以及假新闻检测 [Bhattarai 等，2022b]。此外，研究人员还通过直接操作逻辑规则，探索了 Tsetlin机器的局部和全局可解释性 [Blakely和Granmo, 2021]。

3 Tsetlin 机器自编码器

这里详细介绍基于 Coalesced TM [Glimsdal和Granmo 的 TM自动编码器 2021], , 该模型通过输入掩码和冻结被掩码变量进行了扩展。为便于说明，我们使用三个输入。增加更多输入也很容易。

3.1 架构

输入与输出。 如图 1 所示，TM自动编码器处理并输出命题值：x_1 x_2 x_3 $\in$ $(\{0,1\}^3 \rightarrow \widehat{x}_1\ \widehat{x}_2\ \widehat{x}_3 \in \{0,1\}^3$,,)

(,,) 。对我们的用途而言，命题变量 x_1、 x_2 和 x_3 各自代表一个词，例如“Brilliant”（精彩）、“Actor”（演员）和“Awful”（糟糕）。值 1 表示该词出现在输入文本中，而值 0 表示该词未出现。也就是说，我们将自然语言文本表示为一个词集。还请注意，输入变量分别对应输出变量 $\widehat{x}_1$、 $\widehat{x}_2$ 和 $\widehat{x}_3$。简言之，根据 x_2 和 x_3 预测 $\widehat{x}_1$，根据 x_1 和 x_3 预测 $\widehat{x}_2$，依此类推。继续我们的例子， $\widehat{x}_1$ 根据“Actor”和“Awful”的出现情况来预测“Brilliant”是否出现。

子句池。 由 n 个合取子句组成的池，表示为C_j, $j \in \{1,2,\ldots,n\}$，用于对输入进行编码，以预测输出。合取子句 C_j 只是针对给定输入子集 $L_j \subseteq \{x_1,x_2,x_3\}$ 的一个与表达式（我们的自编码器不使用输入否定 $\neg x_1$、$\neg x_2$ 和 $\neg x_3$）：

$$C_j(x_1,x_2,x_3) = \bigwedge_{x_k \in L_j} x_k. \tag{1}$$

例如，输入子集 $L_1 = \{x_1,x_2\}$ 生成图中的子句$C_1(x_1,\ x_2,\ x_3) = x_1 \wedge x_2$。当 x_1 和 x_2 都为 1 时，该子句与输入匹配。在我们的示例中，该子句因此编码了“杰出演员”这一概念。

权重。 一个整数权重矩阵 $\boldsymbol{W}$ 将每个子句n 连接到三个输出 $\widehat{x}_1$、$\widehat{x}_2$ 和 $\widehat{x}_3$:

$$\boldsymbol{W} = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ w_{21} & \cdots & w_{2n} \\ w_{31} & \cdots & w_{3n} \end{bmatrix} \in \mathbb{Z}^{3\times n}. \tag{2}$$

行索引是输出，而列索引是子句。例如，权重 w_{12} 将输出 $\widehat{x}_1$ 与子句 C_2 相连。在图 1 中，六个权重连接着两个子句和三个输出：

$$\begin{bmatrix} +4 & -5 \\ +1 & +2 \\ -7 & +6 \end{bmatrix}. \tag{3}$$

例如，图中输出 $\widehat{x}_1$ 的权重（$+4$, -5）。权重 $+4$ 表示子句 C_1（x_1, x_2, x_3）$=x_1 \wedge x_2$ 支持 $\widehat{x}_1$ 为 1，而子句 C_2（x_1, x_2, x_3）$=x_2 \wedge x_3$ 则反对这一结果。例如，“糟糕的演员”这一概念反对输出“杰出”。

3.2 推理

我们先来看看 $\widehat{x}_1$ 的预测。自编码器根据子句和权重预测 $\widehat{x}_1$：

$$\widehat{x}_1 = 0 \leq \sum_{j=1}^{n} w_{j1} C_j(1,x_2,x_3). \tag{4}$$

也就是说，每个子句 C_j 都会乘以其权重 w_{j1}，用于计算输出 $\widehat{x}_1$。随后将各项结果相加，以确定输出。如果总和大于或等于零，输出为 $\widehat{x}_1 = 1$；否则，输出为 $\widehat{x}_1 = 0$。具有正

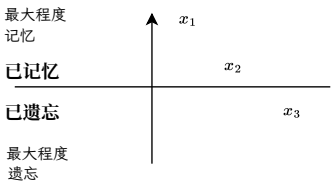

图 2：单个子句的 Tsetlin 机器记忆。

权重的子句会促使输出为 $\widehat{x}_1 = 1$，而具有负权重的子句则会促使输出为 $\widehat{x}_1 = 0$。请注意， x_1 会通过替换为值 1 来进行屏蔽。因此，自编码器会根据其余输入 x_2 和 x_3推断输出 $\widehat{x}_1$。

相应地， $\widehat{x}_2$ 和 $\widehat{x}_3$ 分别通过对 x_2 和 x_3进行掩码计算得出：

$$\widehat{x}_2 = 0 \leq \sum_{j=1}^{n} w_{j2} C_j(x_1,1,x_3), \tag{5}$$

$$\widehat{x}_3 = 0 \leq \sum_{j=1}^{n} w_{j3} C_j(x_1,x_2,1). \tag{6}$$

示例。 假设输入始终为 (1,1, 0) 或 (0,1,1)。例如，输入 (1,1,0) 可以表示“杰出演员”，而 (0,1,1) 可以表示“糟糕演员”。接着请注意，图 1 中的式 (4) 如何针对输入 (1,1, 0) 和 (0, 1, 1)，都正确地确定了带有输出 $\widehat{x}_1$ 的掩码输入 x_1。

3.3 学习

接下来，我们将讨论如何学习子句 $C_j, j \in \{1,2,\ldots,n\}$ 的变量子集 L_j，以及如何确定权重矩阵 W 的权重 w_{ji}。

子句记忆。 每个子句 C_j 都有一个分级记忆，其中包含输入变量，如图 2 所示。分级记忆支持根据数据对变量子集进行增量式 学习。可以看到，每个变量都位于四个记忆位置之一（记忆位置的数量是用户可配置参数）。位置 $1-2$ 表示已遗忘。位置 $3-4$ 表示已记忆。已记忆的变量参与子句，而已遗忘的变量不参与。因此，图 2 中的记忆表示子句 $C_j(x_1,x_2,x_3) = x_1 \wedge x_2$。

学习步骤。 TM自动编码器通过三种记忆和权重更新方式进行增量学习：类型 Ia、类型 Ib 和类型 II。每个训练样本的形式为$[k,(x_1,x_2,x_3)\ x_k], 1 \leq k \leq 3$。第一个元素是一个索引，用于标识要屏蔽哪个输入以及要预测哪个输出。第二个元素是一个输入向量（x_1,x_2,x_3），第三个元素是输出 $\widehat{x}_k$ 的目标值，即x_k。下面我们将针对索引为 1 的样本（输出 $\widehat{x}_1$ 预测）逐步介绍更新过程。 $\widehat{x}_2$ 和 $\widehat{x}_3$ 的更新过程则可直接类推。

子句更新概率。 首先，我们根据公式 (4) 计算 $\widehat{x}_1$ 的加权子句和：$v_1 = \sum_{j=1}^{n} w_{j1} C_j(1,\ x_2,\ x_3)$。然后将该和与边际值 T（超参数）进行比较，以计算求和误差 ϵ。误差取决于 x_1值：

$$\epsilon = \begin{cases} T - \mathrm{clip}(v_1,-T,T), & x_1 = 1, \\ T + \mathrm{clip}(v_1,-T,T), & x_1 = 0. \end{cases} \tag{7}$$

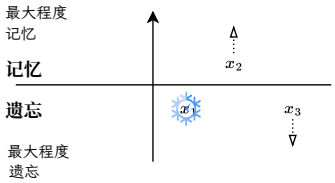

图3：针对输入 (1，1，0) 的 Ia型（识别）反馈。掩码变量 x_1 被冻结。

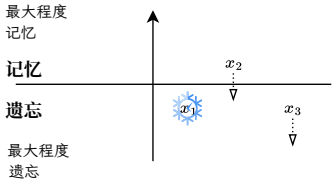

图4：针对输入 (0，0，1) 的 Ib型（擦除）反馈。掩码变量 x_1 被冻结。

也就是说，当 x_1值为 1 时，加权子句和应变为 T；当 x_1值为 0 时，该和应变为 $-T$。因此，学习的目标是让所有输入（x_1，x_2，x_3）都达到边际值，从而确保公式 (4) 得到正确的输出。为实现这一目标，每一轮都会以概率 $\frac{e}{2T}$ 随机更新各个子句 C_j。换言之，更新概率会随误差趋近于零而降低。

更新类型。更新类型取决于 x_1、$C_j(1, x_2, x_3)$ 和 w_{j1} 的值。我们首先考虑权重为正的子句 $w_{j1} \geq 0$。根据公式 4，它们要识别 $x_1 = 1$ 的模式。请注意，在以下所有更新中，掩码变量 x_1 都被冻结，不受更新影响。

- **Ia型（识别）反馈** 发生于 x_1 =1 且 $C_j(1, x_2, x_3) = 1$ 时。此时可以说，$C_j(1, x_2, x_3) = 1$ 是一个真正例，因为它正确预测了掩码 x_1值。Ia 型反馈通过更新 C_j 的记忆，进一步使其模仿输入，从而强化这一成功匹配（见图3）。也就是说，值为 1 的变量会以概率 1。0 在记忆中向上移动一步。[2] 相反，值为 0 的输入会向下移动一步，但这是随机发生的，概率为 $\frac{1}{s}$。这里，s 是一个称为特异性的超参数，意味着 s 越大，子句就越具体。通过将子句的权重 w_{j1} 加 1，进一步强化整个子句。

- **Ib型（擦除）反馈** 发生于 $x_1 = 1$ and$C_j(1, x_2, x_3) = 0$时。此时，我们称 $C_j(1, x_2, x_3) = 0$为假负例，因为它未能促成 $x_1 = 1$。在这种情况下，所有输入都会随机地在记忆中向下移动一步（见图4）。同样，每次向下移动发生的概率为 $\frac{1}{s}$。这样做的目的是通过从子句中擦除变量，消除假负例结果。

[2]最初，增量概率为 $\frac{s-1}{s}$，可以将其提高到1.0，以增强对真正例模式 [Granmo, 2018] 的学习。

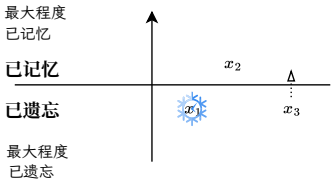

图5：对输入 (0,1,0) 的 II型（拒绝）反馈。掩码变量 x_1 被冻结。

- **II型（拒绝）反馈** 发生于 $x_1 = 0$ and$C_j(1, x_2, x_3) = 1$时。此时，可以说 $C_j(1, x_2) = 1$是假阳性，因为它促成了 $x_1 = 1$，而事实上我们有 $x_1 = 0$。随后，所有已遗忘的、值为 0 的输入都会在记忆中向上移动一步。其目的是通过向子句中注入值为 0 的变量，最终消除当前的假阳性结果。通过将权重 w_{j1} 减 1，子句会进一步减弱。请注意，后一次递减可能会使权重从正数变为负数。实际上，此时子句的作用也会改变，转而训练识别 $x_1 = 0$。

带有负权重的子句 C_j $w_{j1} < 0$也以相同的方式更新。不过，它们要识别的是$x_1 = 0$的模式。为此，在更新记忆时，$x_1 = 0$ 被视为 $x_1 = 1$，而$x_1 = 1$ 被视为 $x_1 = 0$。此外，权重更新的方向也相反：递增变为递减，反之亦然。

算法 1TM词嵌入

要求： 词汇表 $\mathcal{V}$；文档 $\mathcal{D} \in \mathcal{G}, \mathcal{D} \subseteq \mathcal{V}$；累积-量 u；子句 n；边际值 T；特异性 s；轮数 r

TM创建(n, T, s) ▷创建具有 n 个子句的 Tsetlin机器。
对于 r 轮次 **执行**
  对于 $word_k \in \mathcal{V}$ **执行** ▷为每个单词创建一个示例。
    $q_k \leftarrow$ 选择$(\{0, 1\})$ ▷随机目标值。
    如果 $q_k = 1$ **那么**
      $\mathcal{G}_k \leftarrow \{\mathcal{D} | word_k \in \mathcal{D}, \mathcal{D} \in \mathcal{G}\}$ ▷ 含有文档 $word_k$。
    else
      $\mathcal{G}_k \leftarrow \{\mathcal{D} | word_k \notin \mathcal{D}, \mathcal{D} \in \mathcal{G}\}$ ▷ 文档不含有 $word_k$。
    $\mathcal{S}_k \leftarrow$ 选择N项$(\mathcal{G}_k,\ u)$ ▷ 大小为 u的随机子集。
    $\mathcal{U}_k \leftarrow \bigcup_{\mathcal{D} \in \mathcal{S}_k} \mathcal{D}$ ▷ 选定文档的并集。
    $\boldsymbol{x}_k \leftarrow (x_1, x_2, \ldots, x_m)$ $x_i = \begin{cases} 1, & word_i \in \mathcal{U}^k \\ 0, & word_i \notin \mathcal{U}^k \end{cases}$
    TMUpdate$(k, \boldsymbol{x}_k, q_k)$ ▷为输出索引 k、输入 $\boldsymbol{x}_k$ 和目标值 $\hat{x}_k = q_k$ 更新 TM 自动编码器。
$\mathcal{C}, \boldsymbol{W} \leftarrow$ TMGetState() ▷带权重 $\boldsymbol{W}$ 的子句 $C_j \in \mathcal{C}$。
$\boldsymbol{E} \leftarrow$ clip$(\boldsymbol{W}, 0, T)$ ▷对负值进行逐元素裁剪生成加权逻辑词嵌入。
$\boldsymbol{B} \leftarrow (\boldsymbol{W} > 0)$ ▷与零逐元素比较会生成纯逻辑词嵌入。
返回 $\mathcal{C}, \boldsymbol{E}, \boldsymbol{B}$

4 逻辑嵌入流程

现在，我们使用 TM自动编码器来构建逻辑嵌入。设 $\mathcal{V} = \{$单词 $_1$,单词 $_2$，……，单词 $_m\}$为由 m 个不重复单词组成的目标词汇表。

预处理。 第一步是对文档语料库进行预处理。为此，将每篇文档表示为一个词的子集 $\mathcal{D} \subseteq \mathcal{V}$。例如，文档“The ac- tor was brilliant”会变成集合 $\mathcal{D} =$ {“actor”、“ brilliant”、“the”、“was”}。而集合 $\mathcal{G}$则包含所有文档，$\mathcal{D} \in \mathcal{G}$。最后，词集 $\mathcal{D}$的命题向量形式为：

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_t), x_i = \begin{cases} 1, & word_i \in D, \\ 0, & word_i \notin D. \end{cases} \quad (8)$$

嵌入。 算法 1 指定了使用 n 个子句C_j，$1 \leq j \leq n$将 $\mathcal{V}$ 中的 m 词汇表词嵌入，从而形成一个子句集 $\mathcal{C}$。每轮训练都会为 $\mathcal{V}$中的每个词$_k$ 生成一个训练样本 $[k, (x_1, x_2, \ldots, x_m)\ q_k]$。首先，将该词的目标值 q_k 随机设为 0 或 1。这种随机选择可使数据集保持平衡。如果 q_k 变为 1，我们就随机选取 u 篇包含该词$_k$ 的文档，并将其分配到集合 $\mathcal{S}_k$（正例）中。否则，我们随机选取 u 篇不包含该词的文档（负例）。接下来，将随机选定的文档通过 OR 运算合并，得到统一文档 $\mathcal{U}_k$。对多篇文档进行 OR 运算，旨在提高罕见上下文词的出现频率。这样就更容易挑选出具有代表性的词。之后，得到 $\mathcal{U}_k$的命题向量形式（x_1，x_2，…，x_m）。最后，按照第 3 节中的训练流程，使用 $[k, (x_1, x_2, \ldots, x_m)\ q_k]$ 更新 TM 自动编码器。

向量空间表示。 现在可以从矩阵 $\boldsymbol{E}$ （由算法 1 返回）的第 k 行获取《 word 》$_k \in \mathcal{V}$ 的加权逻辑嵌入，而纯逻辑嵌入则位于矩阵 $\boldsymbol{B}$ 的第 k 行。设 $\boldsymbol{e}_k$ 表示 $\boldsymbol{E}$ 的第 k 行，设 $\boldsymbol{e}_l$ 表示第 l 行。于是，我们可以使用两个词《word 》$_k$和《word》$_l$ 各自的 $\boldsymbol{E}$嵌入之间的余弦相似度（CS），来比较它们的相似程度：

$$CS(word_k, word_l) = \frac{\boldsymbol{e}_k \cdot \boldsymbol{e}_l}{||\boldsymbol{e}_k||\ ||\boldsymbol{e}_l||}. \quad (9)$$

5 实证评估

在此，我们评估所提出的逻辑嵌入方案，并将其与神经网络方法进行比较。

5.1 数据集与设置

我们首先对逻辑嵌入进行内在评估，随后使用分类任务进行外在评估。

内在评估。 我们使用词相似度和分类任务基准测试进行内在评估。也就是说，我们考察所采用的方法在多大程度上保留了语义词关系。为此，我们使用六个数据集来衡量语义关系在向量空间中的体现：SimLex-999、WordSim-353、MEN、MTurk-287、Mturk-771 和 RG-65。每个数据集都包含人工评分词对，这些词对将与相应的向量空间相似度进行比较。分类任务评估我们仅根据词的嵌入将其归入不同词类别的能力。我们在此使用三个数据集：AP、BLESS 和 ESS-LLI。[3] 作为基线，我们选择了应用广泛的 Word2Vec、GloVe 和 FastText。

外在评估。 在外在评估中，我们考察逻辑嵌入对下游自然语言处理分类任务的支持效果。将词嵌入用作特征向量，通过监督分类模型的性能来了解嵌入质量。我们采用 SentEval 中的六个标准文本分类数据集 [Conneau 和 Kiela， 2018]：R8、R52、TREC、SUBJ、SST-2 和 SST-5。对于监督学习，我们使用标准的基于注意力机制的双向长短期记忆网络模型，以及 Adam 优化器和交叉熵损失函数。这样，我们便可以直接比较 GloVe 嵌入与 Tsetlin机器方法。

嵌入数据集。 在使用 BiL-TM 进行外在评估时，我们采用标准的 300 维 GloVe 嵌入，这些嵌入是在维基百科 *2014* + *Gigaword 5* 数据集（60亿个词元）上预训练的。[4] 我们的目的是将 TM 嵌入在下游任务中的表现，与广泛使用且表现出色的 GloVe 嵌入进行比较。为了直接比较 Word2Vec、GloVe、FastText 和 TM 嵌入的内在属性，我们还使用十亿词数据集从头开始训练它们 [Chelba 等， 2014]。训练 TM 时，我们使用算法 1 进行 $r = 2000$ 轮训练；通过为每个示例累计 $u = 25$ 个上下文，生成每个词 2000 个示例。我们使用以下超参数：包含 $n = 600$ 个子句的池、边际值 $T = 1200$和特异性 $s = 5.0$。[5] Word2Vec Skip-Gram 在数据上训练 10 轮次，并为输入和输出上下文使用独立的嵌入。窗口大小为 5，每个示例使用五个负样本。类似地，GloVe 训练 30 轮次，窗口大小为 10，学习率为 0。05。[6]

5.2 结果与讨论

如第 5.1 节所述，我们采用两种评估方式：内在评估和外在评估。表 1 汇总了六项词语相似度任务的内在评估结果。我们针对每个数据集，计算人工标注的相似度分数与预测相似度分数之间的斯皮尔曼相关系数、肯德尔系数和余弦相似度。从斯皮尔曼和肯德尔得分来看，Word2Vec 和 GloVe 略优于表现相近的 FastText 和 TM 嵌入。然而，正如 [Rastogi 等人所报告的， 2015],

[3]为获得分类准确率，我们使用 sklearn 中的 KMeans 聚类对词嵌入进行聚类，并通过计算（https://github.com/purity）中的纯度分数来检验聚类质量。[4]预训练 GloVe 词嵌入可在此处获取：https://nlp.stanford.edu/projects/glove/[5]TM自动编码器和逻辑词嵌入的实现可在此处获取：https://github.com/cair/tmu。[6] Word2Vec 和 FastText 使用标准 gensim 库进行训练（ https://github.com/RaRe-Technologies/gensim/tree/develop/gensi）。GloVe 使用 https://github.com/maciejkula/glove-python 进行训练。

数据集	W2V			FastText			TM			GloVe		
	斯皮尔曼	肯德尔	余弦	斯皮尔曼	肯德尔	余弦	斯皮尔曼	肯德尔	余弦	斯皮尔曼	肯德尔	余弦
WS-353	0.53	0.37	0.87	0.46	0.32	0.79	0.45	0.31	0.90	0.41	0.28	0.90
SIM999	0.26	0.18	0.79	0.23	0.16	0.79	0.14	0.10	0.76	0.25	0.17	0.80
MEN	0.71	0.50	0.91	0.71	0.51	0.94	0.64	0.45	0.94	0.73	0.53	0.95
MTURK287	0.66	0.47	0.77	0.63	0.44	0.93	0.63	0.44	0.92	0.66	0.47	0.86
MTURK717	0.57	0.39	0.86	0.52	0.36	0.93	0.48	0.32	0.91	0.58	0.40	0.94
RG65	0.72	0.58	0.89	0.67	0.49	0.88	0.75	0.63	0.92	0.78	0.62	0.93
平均值	0.58	0.42	0.85	0.54	0.38	0.88	0.52	0.38	0.89	0.57	0.42	0.90

表 1：TM 嵌入与基线算法在相似度任务上的性能比较。

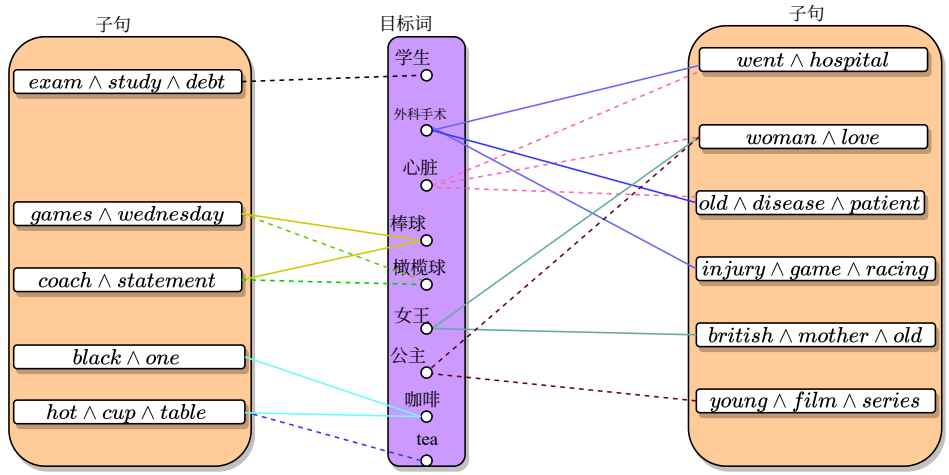

图 6：TM 嵌入中捕捉目标词不同含义的子句的可解释性

.

数据集	W2V	FastText	TM	GloVe
AP	0.50	0.35	0.41	0.41
BLESS	0.64	0.66	0.62	0.66
ESSLI	0.63	0.60	0.57	0.56
平均值	0.59	0.54	0.53	0.54

表 2：Tsetlin机器嵌入与基线嵌入在分类任务上的性能比较。

数据集	GloVe		TM		Tsetlin机器hybrid	
	Acc.	F1	Acc.	F1	Acc.	F1
R8	96.31	0.88	96.10	0.88	97.80	0.94
TREC	95.20	0.95	96.40	0.96	96.80	0.96
R52	90.34	0.58	91.23	0.62	94.23	0.68
SUBJ	86.20	0.86	85.80	0.85	86.70	0.87
SST-2	76.38	0.75	75.61	0.74	79.30	0.78
SST-5	47.47	0.46	47.80	0.43	49.75	0.44

表3：我们提出的嵌入与标准 GloVe 嵌入在分类任务上的性能比较。

对于较小的数据集，基于相关性的指标差异不一定具有统计显著性。因此，为了更稳健地评估性能，我们还使用余弦相似度，将预测的词语相似度与人工标注的相似度进行比较。就余弦得分而言，我们的模型在大多数数据集上的表现优于 Word2Vec 和 FastText，而表现与 GloVe 不相上下。这意味着人工设定的相似度与 GloVe/TM 预测的相似度之间的角度相当接近。最后，表 2展示了词语分类任务的结果。如表所示，所选嵌入技术的性能相当，Word2Vec 略胜一筹。

先前研究表明，内在词语相似度表现与下游自然语言处理性能之间的相关性极低，甚至呈负相关 [Wang 等，2021]。因此，我们还通过六项下游分类任务进行了外在评估。为避免过拟合并稳健地评估下游性能，我们沿用上述实验设置。表 3 汇报了评估结果，其中嵌入被输入基于注意力机制的双向长短期记忆网络模型。第一种配置（GloVe）使用来自维基百科 *2014+ Gigaword 5* 数据集的预训练 GloVe 词嵌入。第二种配置采用我们基于十亿词的纯逻辑 TM 嵌入（嵌入 $\boldsymbol{B}$ 来自算法 1）。十亿词数据集的规模只有前者的五分之一，因此只能提供分类任务所需词汇表的大约 80%。剩余 20% 的词语则使用随机嵌入。因此，TM 方法在评估中可能处于劣势。第三种配置（TM$_{\text{hybrid}}$）将 20% 的随机嵌入替换为相应的 GloVe 嵌入（约 80% TM + 20% GloVe）。我们注意到，双向长短期记忆网络在 TM 和 GloVe 上的下游准确率相近。具体而言，TM 嵌入

在 TREC、R52 和 SST-5 上略微优于 GloVe。另一方面，混合嵌入明显优于另外两种嵌入。尤其是在 R52、SST-2 和 SST-5 上，混合嵌入以约 $2-4\%$ 的显著边际值超越了 GloVe。鉴于这些数据集并非完全均衡，我们还计算了宏平均 F1 分数。我们再次发现，TM 嵌入要么优于 GloVe，要么与之相当。在 R8 和 R52 上，混合嵌入以较大优势超越 GloVe，分别约为 6% 和 10%。基于这些结果，我们推测逻辑 TM 嵌入能够成功替代神经网络嵌入。即使缺少 20% 的词汇表，且训练数据只有原来的五分之一，逻辑嵌入的表现仍可与 GloVe 相媲美。有意思的是，混合方法的表现甚至更好。性能提升的一种可能解释是，更大的词汇表带来了额外信息。此外，神经表示与逻辑表示之间可能存在协同作用，并体现在混合方法中。

5.3 可解释性与可视化

在本节中，我们将更详细地探究 Tsetlin 机器嵌入的性质，重点关注可解释性。我们的嵌入由正子句权重 $\boldsymbol{E}$ 构成，或者也可以由命题版本 $\boldsymbol{B}$ 表示，该版本由子句集合 $\mathcal{C}$ 解释。如图 6 所示，$\mathcal{C}$ 中的每个子句都捕捉了上下文的一个方面。图中的虚线展示了目标词与矩阵 $\boldsymbol{B}$ 中相应子句之间的联系（因此也展示了与 $\boldsymbol{E}$ 之间的联系）。每个目标词都有自己的颜色，以便更容易辨别这些联系。在图中，我们从 $\boldsymbol{B}$ 中摘取了 18 个联系，其中涉及 8 个目标词，以及针对这些词触发次数最多的 11 个子句。例如，考虑目标词 外科手术 和 心脏。这两个目标词共享两个子句：[前往 $\wedge$ 医院] 和 [古老的 $\wedge$ 疾病 $\wedge$ $patient$]。这两个子句捕捉了两个共同上下文，且都与健康有关。因此，这些子句体现了目标词之间的共性，为词语的一种特定含义提供了信息。

这两个目标词在语义上也有所不同。这些差异体现在它们各自拥有、彼此不共享的子句中。例如，目标词心脏还与含义[女性 $\wedge$ 爱情], 相关，而外科手术 则不然。另一方面，外科手术与 [受伤 $\wedge$ 比赛 $\wedge$ 赛车]相关。通过这种方式，词语之间独特的含义和关系得以通过共享逻辑表达式来表示。因此，可以用简洁的逻辑表达式捕捉广泛的潜在上下文表示。由此，逻辑嵌入为词语及其关系提供了稀疏表示。事实上，在我们的实验中，与每个词相连的子句最多占 10%。正如上一小节中的内在评估所示，这些上下文表示能够有效地衡量词相似度并进行词语分类。同样，我们发现逻辑嵌入能够提升下游自然语言处理分类任务的表现。

为了进一步阐明 TM 嵌入方法，我们在图 7 中将 SimLex-999数据集中 400 个单词的嵌入可视化，并使用 t-SNE 绘制。图中显示

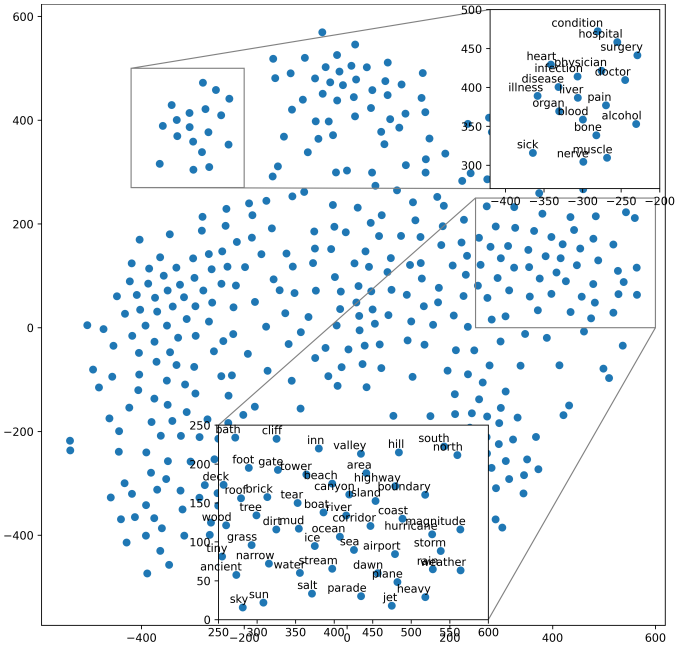

图 7：使用 t-SNE 绘制的 TM 嵌入可视化图。

我们能够在向量空间中聚类上下文相似的单词。为了仔细考察这些簇，我们放大其中两个。先看右上方的簇。注意簇中的单词与医院有关，例如心脏 和疾病。如图所示，这些词嵌入在向量空间中的位置彼此很近。同样，我们可以看到，与天气和地理相关的术语被归为底部的一个簇。从这两个例子来看，TM 嵌入显然捕捉到了单词之间的语义关系。

6 结论与未来工作

在本研究中，我们首先讨论了寻找计算上更简单且更具可解释性的词嵌入方法所面临的挑战及其必要性。随后，我们提出了一种高效的自监督方法，即基于TM的自编码器，用于生成稀疏且具有可解释性的逻辑词嵌入。我们在广泛的内在任务和外在任务上评估了该方法，结果表明，它与Word2Vec、GloVe和FastText等基于稠密神经网络的嵌入方案具有竞争力。此外，我们还通过可视化和案例研究考察了嵌入的可解释性。我们的研究结论是，逻辑嵌入能够利用逻辑表达式表示词语。这种结构使表示具有稀疏性，从而能够将每个词清晰地分解为若干语义概念集合。

未来的工作包括使用图形处理器扩展我们的实现，以支持从更多海量数据集中构建大规模词汇表。此外，我们打算研究如何利用子句创建句子级和文档级嵌入，例如将其应用于下游句子相似度任务。

参考文献

[Angelov, 2020] Dimo Angelov。Top2vec：主题的分布式表示。*arXiv* 预印本 *arXiv:2008.09470*，2020。

[Arora等， 2017] Sanjeev Arora、Yingyu Liang 和 Tengyu Ma。《一种简单却难以超越的句子嵌入基线》。载于国际学习表征会议，2017年。

[Arora等， 2018] Sanjeev Arora、Yuanzhi Li、Yingyu Liang、Tengyu Ma 和 Andrej Risteski。《词义的线性代数结构及其在多义性中的应用》。《语言学计算协会学报》，6:483–495，2018年。[Bengio 等， 2000] Yoshua Bengio、R´ejean Ducharme 和 Pascal Vincent。《神经概率语言模型》。《神经信息处理系统进展》，13，2000年。[Bhattarai等，2022a] Bimal Bhattarai、Ole-Christoffer Granmo 和 Lei Jiao。《用于通用异常值和新颖性检测的 Tsetlin 机器框架》。载于国际智能体与人工智能会议，第250–268页。Springer，2022年。[Bhattarai等，2022b] Bimal Bhattarai、Ole-Christoffer Granmo 和 Lei Jiao。《用于假新闻检测的可解释 Tsetlin 机器框架及可信度评分评估》。载于 *LREC*，2022年。[Blakely 和 Granmo， 2021] Christian D. Blakely 和 Ole-Christoffer Granmo。《Tsetlin 机器全局与局部解释的闭式表达式》。载于第*34*届工业、工程及其他应用智能系统国际会议（*IEA/AIE 2021*）。Springer，2021年。[Bojanowski 等， 2017] Piotr Bojanowski、Edouard Grave、Armand Joulin 和 Tomas Mikolov。《利用子词信息丰富词向量》。《语言学计算协会学报》，5:135–146，2017年。[Borgeaud 等， 2022] Sebastian Borgeaud、Arthur Mensch、Jordan Hoffmann、Trevor Cai、Eliza Rutherford、Katie Millican、George Bm Van Den Driessche、Jean-Baptiste Lespiau、Bogdan Damoc、Aidan Clark 等。《通过从数万亿个词元中检索来改进语言模型》。载于国际机器学习会议，第2206–2240页。PMLR，2022年。[Cer等， 2018] Daniel Cer、Yinfei Yang、Sheng-yi Kong、Nan Hua、Nicole Limtiaco、Rhomni St John、Noah Constant、Mario Guajardo-Cespedes、Steve Yuan、Chris Tar 等。《面向英语的通用句子编码器》。载于*2018*年自然语言处理实证方法会议论文集：系统演示，第169–174页，2018年。[Charniak 等， 2013] Eugene Charniak 等。《朴素贝叶斯词义归纳》。载于*2013*年自然语言处理实证方法会议论文集，第1433–1437页，2013年。

[Chelba等人， 2014] Ciprian Chelba、Tomas Mikolov、Mike Schuster、Qi Ge、T. Brants、Phillip Todd Koehn和Tony Robinson。《用于衡量统计语言建模进展的十亿词基准》。发表于Interspeech，2014年。[Conneau和Kiela， 2018] Alexis Conneau和Douwe Kiela。《SentEval：通用句子表示评测工具包》。发表于第十一届语言资源与评测国际会议（LREC 2018）论文集，2018年。[Devlin等人， 2019] Jacob Devlin、Ming-Wei Chang、Kenton Lee和Kristina Toutanova。《BERT：用于语言理解的深度双向Transformer预训练》。发表于NAACL，2019年。[Dhillon等人， 2015] Paramveer S Dhillon、Dean P Foster和Lyle H Ungar。《特征词：谱词嵌入》。《机器学习研究期刊》，16：3035–3078，2015年。[Faruqui等， 2015] Manaal Faruqui、Yulia Tsvetkov、Dani Yogatama、Chris Dyer和Noah A Smith。《稀疏过完备词向量表示》。发表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七届会议论文集（第1卷：长篇论文），第1491–1500页，2015年。[Glimsdal和Granmo， 2021] Sondre Glimsdal和Ole-Christoffer Granmo。《具有子句共享的合并式多输出Tsetlin机器》。提交至ICML’23。可作为arXiv预印本获取，arXiv:2108.07594，2021年。[Granmo， 2018] Ole-Christoffer Granmo。《Tsetlin机器——一种由博弈论驱动的老虎机方法：利用命题逻辑实现最优模式识别》。arXiv预印本，arXiv:1804.01508，2018年。[Harris， 1954] Zellig S Harris。《分布结构》。《Word》，10(2-3)：146–162，1954年。[Huang等， 2020] Jui-Ting Huang、Ashish Sharma、Shuying Sun、Li Xia、David Zhang、Philip Pronin、Janani Padmanabhan、Giuseppe Ottaviano和Linjun Yang。《Facebook搜索中的基于嵌入的检索》。发表于第26届ACM SIGKDD知识发现与数据挖掘国际会议论文集，第2553–2561页，2020年。[Kenton和Toutanova， 2019] Jacob Devlin Ming-Wei Chang Kenton和Lee Kristina Toutanova。《BERT：用于语言理解的深度双向Transformer预训练》。发表于NAACL-HLT论文集，第4171–4186页，2019年。[Lample等人， 2016] Guillaume Lample、Miguel Ballesteros、Sandeep Subramanian、Kazuya Kawakami和Chris Dyer。《命名实体识别的神经网络架构》。发表于NAACL-HLT论文集，第260–270页，2016年。[Lau等， 2014] Jey Han Lau、Paul Cook、Diana McCarthy、Spandana Gella和Timothy Baldwin。《学习词义分布、检测未见证词义并识别新词义：使用主题模型》。发表于

计算语言学协会第*52*届年会论文集（第*1*卷：长篇论文），第259–270页，2014年。

[Levy等， 2015] Omer Levy、Yoav Goldberg 和 Ido Da- gan。《从词嵌入中汲取经验以改进分布相似度》。计算语言学协会汇刊，3：211–225，2015年。

[Li等，2020] Bohan Li、Hao Zhou、Junxian He、Mingx- uan Wang、Yiming Yang 和 Lei Li。《预训练语言模型的句子嵌入》。载于*2020*年自然语言处理实证方法会议（*EMNLP*）论文集，第 9119–9130 页，2020年。[Logeswaran和Lee， 2018] Lajanugen Logeswaran 和 Honglak Lee。《学习句子表示的高效框架》。载于国际学习表征会议，2018年。[Mikolov等， 2013] Tomas Mikolov、Kai Chen、Gre- gory S. Corrado 和 Jeffrey Dean。《在向量空间中高效估计词表示》。载于国际学习表征会议，2013年。[Mu等， 2017] Jiaqi Mu、Suma Bhat 和 Pramod Viswanath。《多义性的几何结构》。载于第*5*届国际学习表征会议（*ICLR 2017*），2017年。[Pennington等， 2014] 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，第 1532–1543 页，2014年。[Peters 等， 2018] Matthew E Peters、Mark Neu mann、Mo- hit Iyyer、Matt Gardner、Christopher Clark、Kenton Lee 和 Luke Zettlemoyer。《深度上下文化词表示》。*arXiv*预印本 *arXiv:1802.05365*，2018年。[Rastogi等， 2015] Pushpendre Rastogi、Benjamin Van Durme 和 Raman Arora。《多视图潜在语义分析：通过广义典型相关分析进行表征学习》。载于计算语言学协会北美分会：人类语言技术会议*2015*年论文集，第 556–566 页，2015年。[Reimers和Gurevych， 2019] Nils Reimers 和 Iryna Gurevych。《Sentence-BERT：使用孪生 BERT 网络进行句子嵌入》。载于*2019*年自然语言处理实证方法会议及第*9*届国际自然语言处理联合会议（*EMNLP-IJCNLP*）论文集，第 3982–3992 页，2019年。[Schwartz等， 2020] Roy Schwartz、Jesse Dodge、Noah Smith 和 Oren Etzioni。《绿色人工智能》。美国计算机协会通讯，63:54–63，2020年。[Subramanian等， 2018] Anant Subramanian、Danish Pruthi、Harsh Jhamtani、Taylor Berg-Kirkpatrick 和 Eduard Hovy。《SPINE：稀疏可解释神经嵌入》。载于*AAAI* 人工智能会议论文集，第32卷，2018年。

[Vaswani等， 2017] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。《Attention is all you need》。载于神经信息处理系统进展，第30卷，2017年。[Wang等， 2021] Kexin Wang、Nils Reimers 和 Iryna Gurevych。《Tsdae：使用基于 Transformer 的序列去噪自编码器进行无监督句子嵌入学习》。*arXiv*预印本 *arXiv:2104.06979*，2021年。[Yadav等， 2021] Rohan Yadav、Lei Jiao、Ole-Christoffer Granmo 和 Morten Goodwin。《面向基于方面的情感分析的人类水平可解释学习》。载于*AAAI* 论文集，2021年。[Zhang等， 2020] Yan Zhang、Ruidan He、Zuozhu Liu、Kwan Hui Lim 和 Lidong Bing。《一种通过互信息最大化实现的无监督句子嵌入方法》。载于*2020*年自然语言处理实证方法会议（*EMNLP*）论文集，第 1601–1610 页，2020年。