Crossword：一种基于掩码的数据压缩语义方法

Mingxiao Li†、Rui Jin†、Liyao Xiang§、Kaiming Shen† 和 Shuguang Cui†
†科学与工程学院，FNii，香港中文大学（深圳），中国§上海交通大学，中国 电子邮箱：{mingxiaoli, ruijin}@link.cuhk.edu.cn;xiangliyao08@sjtu.edu.cn; shenkaiming@cuhk.edu.cn; shuguangcui@cuhk.edu.cn

arXiv:2304.01106v1 [cs.CL] 3 Apr 2023

摘要——传统的数据压缩方法通常基于符号级统计，将信息源建模为由独立同分布随机变量构成的长序列或随机过程，从而确立了无损压缩的基本极限为熵，有损压缩的基本极限为互信息。然而，现实世界中的信息源（包括文本、音乐和语音）往往难以进行明确的统计建模，因为它们与人类感知密切相关，因此模型驱动方法可能远非最优。本研究重点关注英文文本，并利用其语义方面进一步提高压缩效率。核心思想源自填字游戏：只要给出一些“关键”字母，隐藏的词语仍然可以被准确还原。本文提出的基于掩蔽的策略与上述游戏类似。简而言之，编码器根据语义损失评估每个词的语义重要性，然后对不重要的词进行掩码；解码器则借助 Transformer，根据语义上下文恢复掩码词。实验结果表明，所提出的语义方法在很大程度上保留目标文本含义的同时，能够实现远高于霍夫曼编码和 UTF-8 编码等传统方法的压缩效率。

索引词——语义信源编码、数据压缩、词语掩蔽、余弦距离、Transformer。

I. 引言

数据压缩，或通信系统中的信源编码，通常被视为一种涉及人类感知的巧妙技术，尤其适用于文学、音乐、语音等信源。事实上，Shannon 在其开创性论文中断言 [1]: “通信的这些语义方面与工程问题无关”，其前提是

（i）信源可以建模为随机过程，甚至可以建模为独立同分布随机变量序列；（ii）信源包含无限多个符号。

然而，上述假设在实践中都未必成立。本文提出了一种英语文本压缩的语义方法，该方法将词的含义纳入考虑，优于传统的基于模型的方法。其核心思想是掩蔽全文中语义上不太重要的词语，从而在保留文本整体含义的同时提高压缩效率。

所提出的语义数据压缩掩蔽策略受广受欢迎的文字谜题 Crossword 启发——其目标是在空白方格中填入字母，从而组成词语或短语，使其符合给定的方格布局以及已经填入字母的固定方格。事实证明，上述文字谜题与数据压缩存在类比关系。编码器可以被视为谜题设计者，旨在尽可能多地掩蔽词语（即最大化空白方格所占的比例），以增加谜题的难度。同时，解码器则充当玩家，目标是成功恢复所有词语。我们寻求能够实现均衡的最优词语掩蔽方案，以同时满足谜题设计者和玩家的需求。此外，作为这一谜题的主要特征，词语彼此交叉；可以认为，位于这些交叉方格中的字母通常更容易猜出。同样地，对于英文文本压缩而言，如果移除某些特定词语不会造成太大的语义损失，那么这些词语就可以被视为“交叉方格”类型，从而能够在不削弱解码器恢复文本能力的情况下被掩蔽。需要注意的是，对这类词语的选择通常取决于语境。例如，正如我们的实验所观察到的，如果样本文本取自欧洲议会拥有的一份政治文件，那么词语“affairs”在语义上很可能并不重要。诸如双向编码器表示（BERT） [2],[3] 和 Transformer [4] 之类的尖端人工智能工具，是感知背景知识、检测语义上不重要的词语以及保留语义信息的关键促成因素。

语义压缩的研究可追溯至 [5] 20世纪90年代，当时有人提出一个简单的想法：用同义词词典中更短的同义词替换每个词。 [5] 的作者进一步探讨了如何以生成方式模仿特定体裁（例如海明威的作品）。另一项早期尝试见于 [6] ，其重点是可扩展标记语言（XML）。其核心思想是根据重要性对 XML 数据进行分类，然后降低那些能够容忍较高损失的数据的精度。此外， [7] 建议将语义熵定义为词模型与句法消息之间的互信息，并据此衡量语义压缩的极限。另一类研究 [8], [9] 利用知识库中的事实之间的关联，从语义层面减少冗余。近年来，句子-BERT 工具 [3] 已广泛应用于各种语义通信任务中的词嵌入，包括信源编码

通信任务，涵盖从信源编码 [10], [11]到联合信源信道编码 [12], [13]。与旨在逐词进行数据压缩的 [10]不同，我们采用 sentence-BERT 在更高层次上压缩数据，以涵盖整体语境。具体而言，随着神经网络的复兴，该领域最新的发展趋势是采用基于深度学习的方法，例如考虑资源分配的 [12]、考虑抵御语义噪声的鲁棒传输的 [14]，以及考虑多用户语义通信的 [15]。

II. 系统模型

考虑一篇英文文本 T，其总共包含 M 个句子：

$$T = (S_1, S_2, \ldots, S_M), \tag{1}$$

其中，S_i、$i = 1$、.....、M是第 i个句子。假设这些句子的长度可能不同。为实现数据压缩，T 通过编码函数 $f(\cdot)$被转换为有限长度比特串集合 $\{0,1\}^*$ 中的 $\boldsymbol{x}$，即

$$\boldsymbol{x} = f(T) \in \{0,1\}^*. \tag{2}$$

相反，解码函数 $g(\cdot)$旨在从比特串 $\boldsymbol{x}$ 中恢复文本T，即

$$\widehat{T} = g(\boldsymbol{x}). \tag{3}$$

此外，记 S_i 为恢复出的第 i个句子，我们可以将解码后的文本拆分为

$$\widehat{T} = (\widehat{S}_1, \widehat{S}_2, \ldots, \widehat{S}_M). \tag{4}$$

令 $\delta(T,T) \geq 0$ 表示用解码文本 T表示真实值 T 所产生的失真代价。作为数据压缩问题的一种典型表述，我们寻求使 $\boldsymbol{x}$的长度最小的最优编码器和解码器对（f，g），记为$\text{len}(\boldsymbol{x})$，同时满足失真约束 ϵ，即

$$\underset{f(\cdot),\, g(\cdot)}{\text{minimize}} \quad \text{len}(\boldsymbol{x}) \tag{5a}$$

$$\text{subject to} \quad \delta(T, \widehat{T}) \leq \epsilon. \tag{5b}$$

请注意，在上述问题中，T并非先验固定，因此我们的目标是对 $f(\cdot)$和 $g(\cdot)$进行通用设计，使其适用于所有可能的文本。

还需要指出的是，在理论的传统研究中，通常按符号逐一比较T 和 $\widehat{T}$；英文文本中的符号可以是字母或标点符号。给定来自 T 的第 j个符号 ℓ_j 以及来自T的第 j 个符号 $\hat{\ell}_j$，传统方法会计算符号级失真$d(\ell_j, \hat{\ell}_j)$，汉明失真就是一个常见的例子：

$$d(\ell_j, \hat{\ell}_j) = \begin{cases} 0 & \text{if } \ell_j = \hat{\ell}_j, \\ 1 & \text{otherwise.} \end{cases} \tag{6}$$

因此，总体文本失真 $\delta(T,T)$ 等于所有 j 的符号失真 $d(\ell_j, \hat{\ell}_j)$ 之和。原则上，

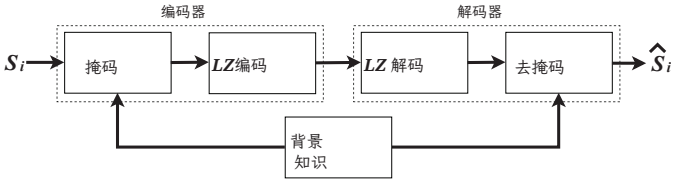

图 1. 所提出的基于掩码的语义数据压缩范式。

上述策略将每个符号视为随机变量单位，因此把文本视为随机变量序列。由此得到的数据压缩算法只能利用符号层面的语言统计特性。

相反，本研究提出以词间和句间语境为基础来评估失真，从而捕捉目标文本的语义方面。考虑一对句子 S_i 和 S_i。首先，利用 sentence-BERT [3], 将它们转换为 $\mathbb{R}^{384}$ 中的两个欧几里得向量，即：

$$\boldsymbol{\mu}_i = \texttt{BERT}(S_i) \quad \text{and} \quad \hat{\boldsymbol{\mu}}_i = \texttt{BERT}(\widehat{S}_i). \tag{7}$$

然后，为了量化用 S_i表示 S_i 的效果，我们使用它们各自的 sentence-BERT 向量之间的余弦相似度

$$\eta(S_i, \widehat{S}_i) = \frac{\boldsymbol{\mu}_i^\top \hat{\boldsymbol{\mu}}_i}{\|\boldsymbol{\mu}_i\|_2 \cdot \|\hat{\boldsymbol{\mu}}_i\|_2} \tag{8}$$

并进一步考虑余弦距离

$$\lambda(S_i, \widehat{S}_i) = 1 - \eta(S_i, \widehat{S}_i). \tag{9}$$

因此，从语义视角出发，我们提出如下 δ的定义（T，$\widehat{T}$）：

$$\delta(T, \widehat{T}) = \sum_{i=1}^{M} \alpha_i \lambda(S_i, \widehat{S}_i), \tag{10}$$

其中，正权重 α_i 反映了 S_i 相对于整体文本 T的语义重要性。我们设定 $\alpha_i = N_i$，因为我们认为语义信息与句子长度成正比。

III. 主要结果

图 1 展示了所提出的数据压缩语义方法的整体范式。下文将分别说明编码部分和解码部分，随后讨论传统熵度量为何不适用于这一范式。

A. 语义编码

所提出的语义编码器由两个部分组成：掩码处理，随后是比特串转换。假设句子S_i 有 N_i 个词。用 W_{in}表示第 n个词，则句子 S_i 可表示为

$$S_i = (W_{i1}, W_{i2}, \ldots, W_{iN_i}). \tag{11}$$

算法 1 语义数据压缩（编码器部分）

输入：英文文本 T 和掩码比例 $\rho \in [0,1)$

初始化：词表 $\mathcal{K} = \emptyset$
对每个句子 S_i 在 T 中执行
  计算 $\boldsymbol{\mu}_i$ =BERT(S_i)
  对每个词 W_{in} in S_i **do**
    更新 $\mathcal{K} = \mathcal{K} \cup \{W_{in}\}$
    计算 $\hat{\boldsymbol{\mu}}_i$ =BERT(S_{in}^-)
    计算 $\sigma_{in} = 1 - \boldsymbol{\mu}_i^\top \hat{\boldsymbol{\mu}}_i / (\|\boldsymbol{\mu}_i\|_2 \cdot \|\hat{\boldsymbol{\mu}}_i\|_2)$
  结束循环
  根据（13）对每个 σ_{in} 进行归一化，作为 $\bar{\sigma}_{in}$
结束循环
对每个词 $V_k \in \mathcal{K}$ 执行
  根据 (15) 计算权重 μ_k
结束循环
按照 μ_k的升序对 $\mathcal{K}$ 中的词语排序
将全文中排名靠前的 $\lfloor \rho|\mathcal{K}| \rfloor$ 个词替换为 # T
使用 LZ编码将掩码文本转换为比特串 $\boldsymbol{x}$
输出：比特串 $\boldsymbol{x}$

对于每个 $n = 1$，……，N_i，我们每次将句子 S_i中的当前 W_{in} 替换为 #；得到的新句子记为S_{in}^-。我们将相应的语义损失计算为

$$\sigma_{in} = \lambda(S_i, S_{in}^-). \tag{12}$$

在获取句子 S_i 中的所有 σ_{in} 后，我们进一步对语义损失进行归一化：

$$\bar{\sigma}_{in} = \frac{\sigma_{in}}{\sum_{n'=1}^{N_i} \sigma_{in'}}. \tag{13}$$

直观地说，$\bar{\sigma}_{in}$ 表示仅由 W_{in} 承载的 S_i 总语义信息的比例。上述过程对每个句子 S_i 都会执行。

随后，我们收集文本中出现过的所有词语并将其排序（例如，按字典序），列入列表

$$\mathcal{K} = \{V_1, V_2 \ldots, V_{|\mathcal{K}|}\}, \tag{14}$$

其中，V_k 是来自 T 的不同词语。我们使用集合 $\mathcal{Q}_{ik}$ 记录词语 V_k 在句子 S_i 中所有出现位置；请注意，同一个词语可能在一个句子中出现多次。特别地，$\mathcal{Q}_{ik} = \emptyset$，如果 S_i 不包含 V_k。对于列表中的每个词语 V_k，其在当前文本中的语义重要性通过权重

$$\mu_k = \frac{1}{\sum_{i=1}^{M} |\mathcal{Q}_{ik}|} \cdot \sum_{i=1}^{M} \left(\alpha_i \sum_{n \in \mathcal{Q}_{ik}} \bar{\sigma}_{in} \right). \tag{15}$$

按 μ_k 的升序对列表中的所有词进行排序。对于给定的掩码比例 $0 \le \rho < 1$，将文本 T 中排名前$\lfloor \rho|\mathcal{K}| \rfloor$ 的词替换为特殊符号“#”，即进行掩码。作为编码的最后一步，我们采用标准方法将掩码后的文本转换为比特串，例如使用 Lempel-Ziv（LZ）编码 [16]。算法 1 总结了上述过程

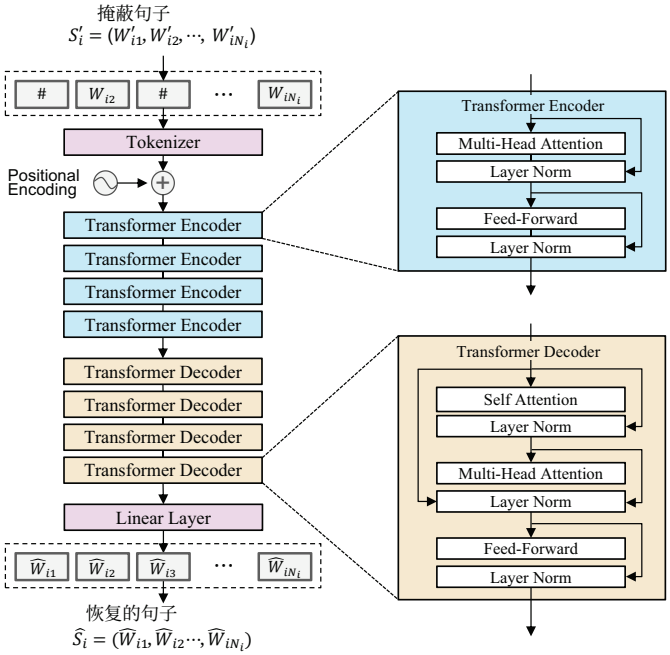

图 2. 语义解码器流程图示意。

编码过程。

B. 语义解码

现在，我们来考虑如何从压缩比特串 $\boldsymbol{x}$ 中恢复文本。首先，我们通过 LZ 解码将 $\boldsymbol{x}$ 转换为词串 T'，该词串由一系列原始句子（S_1'、S_2'、……、S_M'）组成。请注意，每个 S_i' 都是解码器端掩码后的 S_i 的对应项。为恢复每个 S_i' 中被掩码的词语，我们设计了一个基于 Transformer 的去掩码模块 [4]，如图 2 所示。

具体而言，每个原始句子 S_i' 被分解为多个词（其中一些已被遮蔽），然后输入分词器，生成维度为词典大小的独热嵌入向量。如果句子少于 30 个词，则使用零填充。随后，将独热嵌入向量转换为维度为$d = 128$的词嵌入，再将以下位置编码（PE）序列逐项添加到词嵌入中：

$$\mathrm{PE}_{\mathrm{pos},z} = \begin{cases} \sin\left(10^{-\frac{4z}{d}}\mathrm{pos}\right) & \text{if } z \equiv 0 \mod 2 \\ \cos\left(10^{-\frac{4(z-1)}{d}}\mathrm{pos}\right) & \text{if } z \equiv 1 \mod 2. \end{cases}, \tag{16}$$

对于 $z = 1$、2、……、d，其中 pos 是当前句子 S_i' 中每个词的位置索引。每个原始句子 S_i' 的所有词都转换为添加位置编码后的嵌入向量，随后输入 Transformer 编码器。在我们的场景中，Transformer 编码器的多头注意力机制非常理想，因为它能够捕捉目标词与其相邻词之间的交互。因此，从每个词中提取的特征可以从三个方面加以识别：位置编码、前文词嵌入和后文

算法 2 语义数据压缩（解码器部分）

输入：比特串 $\boldsymbol{x}$

1：通过 LZ 解码将 $\boldsymbol{x}$ 转换为掩码原始文本 T'

2：对于每个原始句子 S'_i 在 T' 中执行

3：　将 S'_i 转换为多个独热嵌入向量

4：　加入位置编码

5：　通过 Transformer 编码器提取上下文特征

6：　通过 Transformer 解码器恢复掩码词

7：循环结束

8：拼接文本 $T = (\widehat{S}_1, \widehat{S}_2, \ldots, \widehat{S}_M)$

输出：恢复的文本 $\widehat{T}$

表 I 第 III-C 节示例中五种可能词语的平稳分布。

Word	V_1	V_2	V_3	V_4	V_5
平稳分布	0.15	0.15	0.20	0.25	0.25

词嵌入，所有这些信息都使后续的 Transformer 解码器能够猜测 S'_i 中的掩码词。上述 Transformer 网络基于训练数据集以端到端方式进行调优，从而最小化以下交叉熵:

$$\mathrm{CE}(S_i, \widehat{S}_i) = -\sum_{(n,k)} q_n(V_k) \cdot \log_2 p_n(V_k) - \sum_{(n,k)} (1 - q_n(V_k)) \cdot \log_2(1 - p_n(V_k)), \quad (17)$$

其中 $q_n(V_k) \in \{0, 1\}$ 是这样的真实标签：当 $W_{in} = V_k$ 时，$q_n(V_k) = 1$；否则，$q_n(V_k) = 0$；而 $p_n(W_{in}) \in [0, 1]$ 是软判决，反映第 n 个词是 V_k 的可能性。Transformer 训练完成后，我们按照算法 2 对比特串进行解码。

C. 熵与语义损失

现在，我们举例说明为何在式 (10) 中使用语义损失对词掩码方案至关重要。为便于讨论，考虑一个英文文本玩具模型，其中仅有五个可能的词 $\{V_1, \ldots, V_5\}$，它们构成一个不可约且非周期的五状态马尔可夫链——该链必定具有唯一的平稳分布。表 I 展示了上述模型的平稳分布。因此，若目标是实现无损数据压缩，则可通过霍夫曼编码达到最优结果（即表示文本所需的最小期望比特数）。图 3 展示了一种可能的霍夫曼树构造方式。

现在，假设我们希望通过选择$\{V_1, \ldots, V_5\}$ 中的两个词语进行合并，进一步降低数据压缩中的比特消耗。显然，根据数据处理不等式，这将使马尔可夫链各状态的熵降低。我们指出，词合并可以解释为用特殊符号“#”替换所选的两个词语，即进行词语掩蔽。如果继续追求

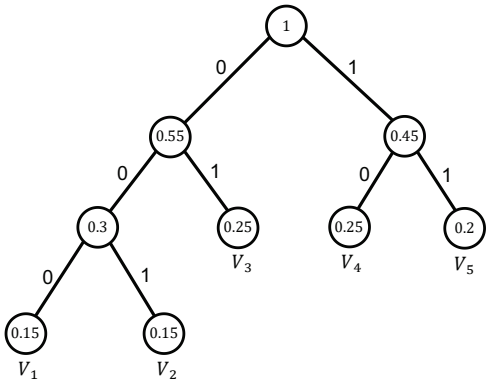

图 3. 无损数据压缩的霍夫曼树。

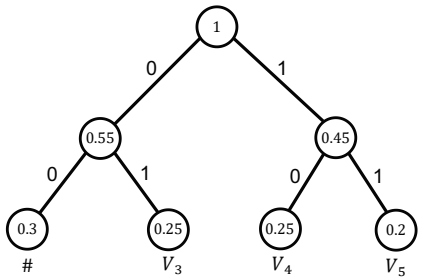

图 4. 词合并后的霍夫曼树。

最小熵，那么合并概率最低的两个词语 V_1 和 V_2是最优的，如图 4 所示。然而，从语义角度来看，通常应当掩蔽停用词（例如“a”“an”和“the”），或特定语境中频繁使用的词语（例如学术论文中机器学习领域的“learning”），因此，合并后的词语预计会具有较高的平稳概率。上述差异表明，传统的熵度量并不适用于基于掩码的数据压缩。

IV. 实验

A. 设置

数据集：实验中使用的样本文本取自欧洲议会 2005 年会议记录（英文版） [17]。该文本约由 80,000 个句子组成，总词数超过 150 万。每个句子的长度介于 4 至 30 个词之间。数据集分为两组：90% 用于训练，其余 10% 用于测试。

超参数：我们的 Transformer 模型训练 120 个 epoch，直到完全收敛。我们使用 Adam优化器[18] ，参数设置为 1×10^{-4} $\beta = (0.9, 0.98)\epsilon = 1 \times 10^{-8}$，权重衰减为 5×10^{-4}。

竞争算法：除所提出的数据压缩方法外，我们还在实验中纳入了传统的无损压缩方法：霍夫曼编码 [19]和 UTF-8编码 [20]。我们还纳入了一种近期提出的、基于深度学习的语义数据压缩方法，

接下来的项目 **是关于联合**辩论 **的**议程。 **主席**先生、各位委员、**女士们、先生们，我想重点谈谈**欧洲社会基金。 **我**投票 **赞成**修正案，**该修正案由**Trakatelli 的报告**提出，涉及**烤烟叶**的**法规。**接下来是关于**Jack Stewart Clark 爵士的报告，**他代表委员会，就**公民自由和内务**事务**提出了**联合**行动草案。

图 5. 欧洲议会 2005 年会议记录中的一段示例文本 [17]。粗体词语将由算法 1 掩码。

下一项是关于议程的联合辩论的续会内容。主席先生、各位委员、女士们、先生们，我想说的是欧洲社会基金。我投票赞成 Trakatelli 报告提出的关于烤烟叶法规的修正案。接下来是 Jack Stewart Clark Farage 先生代表委员会提交的关于公民自由和内政的行动草案报告。

图 6. 算法 2 对上述段落进行去掩码。恢复出的词语以粗体显示；差异以红色突出标示。

[11]。我们指出，所有方法都基于训练集进行参数化，例如，霍夫曼编码的码本是通过评估训练文本数据中每个词的出现频率构建的。此外，我们考虑所提出的算法的两种变体：

- 长句掩码：此方法将句子合并成长句，并计算式（10）中长句之间的余弦距离。在我们的案例中，原始短句每句最多包含 30 个词，而长句每句包含 227 至 256 个词。
- 基于频率的掩码：与使用式（15）中的注意力权重 μ_k 对词语排序不同，该方法采用频率指标，即对频率最高的词语进行掩码。这种新掩码方案背后的动机将在第 III-C 节讨论。

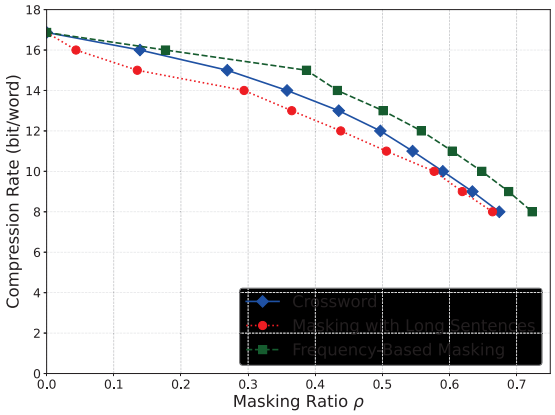

图 7. 不同掩码比例 ρ 下的三种掩码方法。

B. 测试结果

我们首先展示对一个示例段落进行语义压缩的结果，掩码比例为 $\rho = 0.674$，如图 5 和图 6 所示。可以看到，正如第 III-C 节所述，大部分掩码都应用于“of”“is”“to”“the”等停用词。值得注意的是，与语境密切相关的词语和短语，如“affairs”“ladies and gentlemen”和“proposed”，也被掩码了。从图 6 还可以看出，恢复效果相当不错，因为大多数缺失的词语都得以恢复。但仍存在一些差异，其中有些甚至会导致语义误解。例如，“Sir”被替换成了“Mr”，但前者实际上是一种敬称，而不是置于男性姓氏前的普通称谓。这个问题将在训练阶段通过补充背景知识来解决。

接下来，我们在图 7 中比较不同的数据压缩掩码方法。可以看出，随着掩码比例 ρ 增大，三种方法的压缩率都会下降。在相同的 ρ 下，图中显示，长句掩码的压缩效率最高。然而，这并不意味着长句掩码优于其他方法，因为我们尚未考虑它们的语义性能。

现在，我们调整不同掩码方法的掩码比例，以便比较它们在相同压缩率下的语义性能。我们还将 [11] 中现有的语义压缩方法作为基准。对于传统的无损数据压缩方案，霍夫曼编码需要 24.4 比特/词，而 UTF-8 编码需要 47.2 比特/词。有损压缩方法的性能总结见表 II。请注意，当压缩率较高，也就是说，当我们的目标是无损压缩时，传统的基于频率的掩码表现最佳。不过，对于语义研究而言，这种情况意义有限，因为当 ρ 非常低时，主要考虑的是逐符号恢复。相比之下，当 ρ 上升并超过熵极限时，所提出的掩码方案开始占据优势。例如，当平均每个词允许使用 8 比特表示时，所提出的算法在余弦相似度方面比基于频率的掩码高出 27% 以上。还请注意，Crossword 优于长句掩码。一个可能的原因是，按每个短句计算语义损失，可以进一步考虑句子措辞；而将许多句子作为一个整体处理，则会丢失这一关键信息。此外，请注意，在相同的 8 比特/词压缩率下，Crossword 使 [11] 中现有的语义压缩方法性能显著提升了 73% 以上。

此外，我们通过刻画所提出的掩码方法在句子级的表现，进一步研究该方法。定义

表 II 相同压缩率下不同有损压缩方法的语义相似度表现。

句子相似度 比特/词 方法	8.0	9.0	10.0	11.0	12.0	13.0	14.0	15.0	16.0
[11]中的语义压缩方法	0.500	0.583	0.639	0.709	0.748	0.798	0.795	0.830	0.847
提出的方法“Crossword”	0.869	0.893	0.913	0.936	0.953	0.968	0.980	0.990	0.994
长句掩码	0.837	0.869	0.893	0.909	0.931	0.941	0.952	0.962	0.974
基于频率的掩码	0.684	0.732	0.768	0.849	0.883	0.921	0.958	0.983	0.997

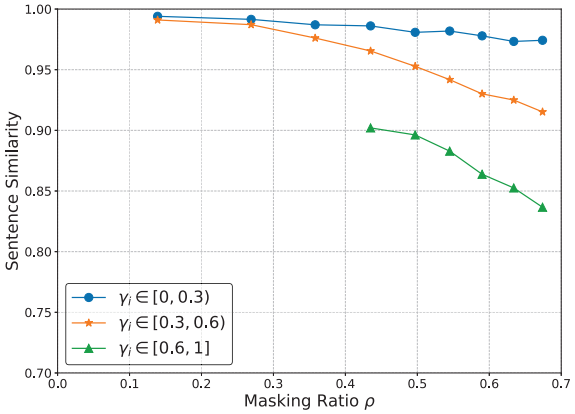

图 8. 不同句子组的余弦相似度 $\eta(S_i, \hat{S}_i)$与掩码比例 ρ 之间的权衡。

句子掩码率 $\gamma_i \in [0,1]$ 为句子 S_i中被掩码的词语所占的比例。我们根据句子掩码率将句子分为以下三组：$\gamma_i \in [0, 0.3)$、$\gamma_i \in [0.3, 0.6)$和$\gamma_i \in [0.6, 1]$。如图 8 所示，当 ρ 趋近于 1 以实现极高的压缩效率时，$\rho \in [0.6, 1]$ 的句子会遭受严重的语义损失，其与真实值的余弦相似度仅为 83%。但随着 ρ 降低，这种损失会迅速减轻。另一个重要发现是，当ρ 低于 0.42 时，没有句子落入 $\gamma_i \in [0.6, 1]$ 组。原因在于，如果 ρ 接近于零，语义压缩便会逐渐退化为传统的符号压缩，因此大多数句子的句子掩码率最终会 $\gamma_i \approx 0$。

V. 结论

本研究受填字游戏启发，提出了一种针对英文文本的数据压缩语义方法。其核心思路是根据整体语境，对语义上不太重要的词语进行掩码。我们使用 sentence-BERT 评估语义重要性，并使用 Transfomer 捕捉语义信息，从而恢复掩码词。实验表明，与传统的霍夫曼编码和 UTF-8 编码相比，所提出的方法在减少文本表示所需比特数方面具有显著优势，并且在保留文本含义方面也优于现有的语义压缩方法 [11] 。

参考文献

[1] C. E. Shannon, 《通信的数学理论》, 贝尔系统技术期刊, 第 27 卷, 第 3 期, 第 379–423 页, 1948 年 3 月。[2] J. Devlin 、 M.-W. Chang 、 K. Lee 和 K. Toutanova, 《Bert：用于语言理解的深度双向变换器预训练》, 2018, [在线]。可访问：https://arxiv.org/abs/1810.04805。[3] N. Reimers 和 I. Gurevych, 《Sentence-BERT：使用孪生 BERT 网络的句子嵌入》, 2019, [在线]。可访问：https://arxiv.org/abs/1908.10084。[4] A. Vaswani、N. Shazeer、N. Parmar、J. Uszkoreit、L. Jones、A. N. Gomez、Ł. Kaiser 和 I. Polosukhin, 《注意力就是你所需要的一切》, 载于《神经信息处理系统》（NeurIPS）, 第 30 卷, 2017年。[5] I. H. Witten、T. C. Bell、A. Moffat、C. G. Nevill-Manning、T. C. Smith 和 H. Thimbleby, 《用于有损文本压缩的语义模型与生成模型》, 计算机杂志, 第 37 卷, 第 2 期, 第 83–87 页, 1994年2月。[6] V. P. B. ISI-CNR, 《XML 数据的语义有损压缩》, 2001, [在线]。可访问：http://CEUR-WS.org/Vol-/-cannataro.pdf。[7] P. Basu、J. Bao、M. Dean 和 J. Hendler, 《利用语义关系保持信息质量》, 普适与移动计算, 第 11 卷, 第 188–202 页, 2014年4月。[8] R. Wang、D. Sun、R. Wong、R. Ranjan 和 A. Y. Zomaya, 《SInC：关系数据压缩的语义方法与增强》, 知识型系统, 第 258 卷, 第 110 001–110 001 页, 2022 年 12 月。[9] J. Bao 、 P. Basu 、 M. Dean、C. Partridge、A. Swami、W. Leland 和 J. A. Hendler, 《迈向语义通信理论》, 载于 IEEE 网络科学研讨会, 2011年6月, 第 110–117 页。[10] L. Vial、B. Lecouteux 和 D. Schwab, 《利用 WordNet 语义知识进行词义词汇压缩，以实现神经词义消歧》, 2019, [在线]。可访问：https://arxiv.org/abs/1905.05677。[11] K. Niu、J. Dai、S. Yao、S. Wang、Z. Si、X. Qin 和 P. Zhang, 《迈向语义通信的范式转变》, IEEE 通信杂志, 第 60 卷, 第 11 期, 第 113–119 页, 2022年11月。[12] Z. Qin、X. Tao、J. Lu 和 G. Y. Li, 《语义通信：原理与挑战》, 2021, [在线]。可访问：https://arxiv.org/abs/2201.01389。[13] J. Dai、P. Zhang、K. Niu、S. Wang、Z. Si 和 X. Qin, 《超越比特传输的通信：语义引导的信源与信道编码》, IEEE 无线通信, 2022年8月。[14] Q. Hu、G. Zhang、Z. Qin、Y. Cai、G. Yu 和 G. Y. Li, 《抵御语义噪声的鲁棒语义通信》, 载于 2022 年第 96 届 IEEE 车辆技术会议（VTC 秋季会议）, 2022年9月。[15] H. Xie、Z. Qin 和 G. Y. Li, 《面向视觉问答的任务导向多用户语义通信》, IEEE 无线通信快报, 第 11 卷, 第 3 期, 第 553–557 页, 2021年3月。[16] A. Lempel 和 J. Ziv, 《论有限序列复杂度》, IEEE 信息论汇刊, 第 22 卷, 第 1 期, 第 75–81 页, 1976年1月。[17] P. Koehn, 《Europarl：用于统计机器翻译的平行语料库》, 载于机器翻译峰会会议录, 2005年9月, 第 79–86 页。[18] D. P. Kingma 和 J. Ba, 《Adam：一种随机优化方法》, 2014, [在线]。可访问：https://arxiv.org/abs/1412.6980。[19] D. A. Huffman, 《一种构造最小冗余码的方法》, IRE 会议录, 第 40 卷, 第 9 期, 第 1098–1101 页, 1952年9月。[20] F. Yergeau, 《UTF-8, 10646, 2003, : ISO 的转换格式》在线。可访问：https://www.rfc-editor.org/rfc/rfc .html。