DiffuSum: Generation Enhanced Extractive Summarization with Diffusion

Haopeng Zhang*, **Xiao Liu***, **Jiawei Zhang**
IFM Lab, Department of Computer Science, University of California, Davis, CA, USA
haopeng,xiao,jiawei@ifmlab.org

Abstract

Extractive summarization aims to form a summary by directly extracting sentences from the source document. Existing works mostly formulate it as a sequence labeling problem by making individual sentence label predictions. This paper proposes DiffuSum, a novel paradigm for extractive summarization, by directly generating the desired summary sentence representations with diffusion models and extracting sentences based on sentence representation matching. In addition, DiffuSum jointly optimizes a contrastive sentence encoder with a matching loss for sentence representation alignment and a multi-class contrastive loss for representation diversity. Experimental results show that DiffuSum achieves the new state-of-the-art extractive results on CNN/DailyMail with ROUGE scores of $44.83/22.56/40.56$. Experiments on the other two datasets with different summary lengths also demonstrate the effectiveness of DiffuSum. The strong performance of our framework shows the great potential of adapting generative models for extractive summarization. To encourage more following work in the future, we have released our codes at https://github.com/hpzhang94/DiffuSum

1 Introduction

Document summarization aims to compress text material while keeping its most salient information. It plays a critical role with the growing amount of publicly available text data. Automatic text summarization approaches can be divided into two streams: abstractive and extractive summarization. Although abstractive methods (Nallapati et al., 2016; Gupta and Gupta, 2019; Bae et al., 2019; Li et al., 2020) produce flexible and less redundant summaries, they suffer from problems of generating ungrammatical or even nonfactual contents (Kryściński et al., 2019; Zhang et al., 2022b). In contrast, extractive summarization forms a summary by directly extracting sentences from the source document. Thus, the extracted summaries are grammatically accurate and faithful.

We focus on extractive summarization in this work. Extractive summarization is commonly formulated as a sequence labeling problem, which predicts a $0/1$ label for each sentence, indicating whether the sentence should be included in summary (Nallapati et al., 2017; Zhou et al., 2018; Liu and Lapata, 2019). Compared to individual sentence label prediction in the sequence labeling setting, generative models offer increased flexibility and attend to the entirety of input context. Recent works have also successfully applied generative models to wide-ranging token-level sequence labeling tasks (Athiwaratkun et al., 2020; Du et al., 2021; Yan et al., 2021). Nonetheless, how to apply generative models for sentence-level tasks like extractive summarization has not been explored.

Recently, continuous diffusion models have achieved great success in vision and audio domains (Ho et al., 2020; Kong et al., 2020; Yang et al., 2022; Rombach et al., 2022; Ho et al., 2022). Researchers have also attempted to apply diffusion models for text generation by converting the discrete token to continuous embeddings and mapping from embedding space to words with a rounding method (Li et al., 2022; Yuan et al., 2022; Strudel et al., 2022; Gong et al., 2022). However, these approaches are not applicable for sentence-level tasks like summarization: **(1)** Summarization has a relatively longer input context and larger generation length (around 3-6 sentences), while the above token-level diffusion-LM models are only applicable to short generation tasks like text simplification and question generation. Their performance tends to drop by a large margin when generating longer sequences; **(2)** The word embeddings generated by these models could be indistinguishable, resulting in ambiguous and hallucinated generation; **(3)**

* equal contribution

Findings of the Association for Computational Linguistics: ACL 2023, pages 13089–13100

The rounding step in those existing diffusion models is less efficient and slows down the inference dramatically.

To address the above issues, we propose a novel extractive summarization paradigm, **DiffuSum**, which generates the desired summary sentence representations with transformer-based diffusion models and extracts summaries based on sentence representation matching. Instead of generating word by word, DiffuSum directly generates the desired continuous representations for each summary sentence and thus could process much longer text. DiffuSum is a summary-level framework since the transformer-based diffusion architecture generates all summary sentence representations simultaneously. Moreover, DiffuSum incorporates a contrastive sentence encoding module with a matching loss for sentence representation alignment and a multi-class contrastive loss (Khosla et al., 2020) for representation diversity. DiffuSum jointly optimizes the *sentence encoding module* and the *diffusion generation module*, and extracts sentences by representation matching without any rounding step. We validate DiffuSum by extensive experiments on three benchmark datasets and experimental results demonstrate that DiffuSum achieves a comparable or even better performance than state-of-the-art systems that rely on pre-trained language models. DiffuSum also shows a strong adaptation ability based on cross-dataset evaluation results.

We highlight our contributions in this paper as follows:

(i) We propose DiffuSum, a novel generation-augmented paradigm for extractive summarization with diffusion models. DiffuSum directly generates the desired summary sentence representations and then extracts sentences based on representation matching. To the best of our knowledge, this is the first attempt to apply diffusion models for the extractive summarization task.

(ii) We also introduce a contrastive sentence encoding module with a matching loss for representation alignment and a multi-class contrastive loss for representation diversity.

(iii) We conduct extensive experiments and analysis on three benchmark summarization datasets to validate the effectiveness of DiffuSum. DiffuSum achieves new extractive state-of-art results on CNN/DailyMail dataset with ROUGE scores of $44.83/22.56/40.56$.

2 Related Work

2.1 Extractive Summarization

Recent advances in deep neural networks have dramatically boosted the progress in extractive summarization systems. Existing extractive summarization systems span an extensive range of approaches. Most works formulate the task as a sequence classification problem and use sequential neural models with different encoders like recurrent neural networks (Cheng and Lapata, 2016; Nallapati et al., 2016) and pre-trained language models (Egonmwan and Chali, 2019; Liu and Lapata, 2019; Zhang et al., 2023). Another group of work formulates extractive summarization as a node classification problem and applies graph neural networks to model inter-sentence dependencies (Xu et al., 2019; Zhang and Zhang, 2020; Wang et al., 2020; Zhang et al., 2022a). These formulations are sentence-level methods that make individual predictions for each sentence. Recently, Zhong et al. (2020) observed that a summary consisting of sentences with the highest scores is not necessarily the best. As a result, summary-level formulation like text matching (Zhong et al., 2020; An et al., 2023) and reinforcement learning (Narayan et al., 2018b; Bae et al., 2019) are proposed. Our proposed framework DiffuSum is also a novel summary-level extractive system with generation augmentation. Instead of sequentially labeling sentences, DiffuSum directly generates the desired summary sentence representations with diffusion models and extracts sentences by representation matching.

2.2 Diffusion Models on Text

Continuous diffusion models are first introduced in (Sohl-Dickstein et al., 2015) and have achieved great success in continuous domain generations like image, video, and audio (Kong et al., 2020; Yang et al., 2022; Rombach et al., 2022; Ho et al., 2022). Nevertheless, few works have applied continuous diffusion model to text data due to its inherently discrete nature. Among the initial attempts, Diffusion-LM (Li et al., 2022) first adapts continuous diffusion models for text by adding an embedding step and a rounding step, and designing a training objective to learn the embedding. DiffuSeq (Gong et al., 2022) proposes a diffusion model designed for sequence-to-sequence (seq2seq) text generation tasks by adding partial noise during the forward process and conditional denoising during the reverse pro-

cess. CDCD (Dieleman et al., 2022) is proposed for text modeling and machine translation based on variance-exploding stochastic differential equations (SDEs) on token embeddings. SeqDiffuSeq (Yuan et al., 2022) also proposes an encoder-decoder diffusion model architecture for conditional generation by combining self-conditioning and adaptive noise schedule technique. However, these works only focus on generating token-level embeddings for short text generation (less than 128 tokens). In order to adapt diffusion models to longer sequences like summaries, our DiffuSum directly generates summary sentence embeddings with a partial denoising framework. In addition, DiffuSum jointly optimizes the diffusion model with a contrastive sentence encoding module instead of using a static embedding matrix.

3 Preliminary

3.1 Continuous Diffusion Models

The continuous diffusion model (Ho et al., 2020) is a probabilistic model containing two Markov chains: the forward and the backward process.

Forward Process Given a data point sampled from a real-world data distribution $\mathbf{x}_0 \sim q(x)$, the forward process gradually corrupts $\mathbf{x}_0$ into a standard Gaussian distribution prior $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Each step of the forward process gradually interpolates Gaussian noise to the sample, represented as:

$$q(\mathbf{x}_{t+1}|\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t+1}; \sqrt{1-\beta_t}\mathbf{x}_t, \beta_t\mathbf{I}\right), \quad (1)$$

where $\beta_t \in (0, 1)$ adjusts the scale of the variance.

Reverse Process The reverse process starts from $\mathbf{x}_T \sim \mathcal{N}(0, I)$ and learns a parametric distribution $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to invert the diffusion process of Eq. 1 gradually. Each step of the reverse process is defined as:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_\theta^2(t)\mathbf{I}\right), \quad (2)$$

where $\mu_\theta(\mathbf{x}_t, t)$ and $\sigma_\theta^2(t)$ are learnable means and variances predicted by neural networks.

While there exists a tractable variational lower-bound (VLB) on $\log p_\theta(\mathbf{x}_0)$, Ho et al. (2020) simplifies the loss function of continuous diffusion to:

$$\mathcal{L}_{\text{simple}} = \sum_{t=1}^{T} \left\| \mathbf{x}_0 - \tilde{f}_\theta(\mathbf{x}_t, t) \right\|^2, \quad (3)$$

where $\tilde{f}_\theta(\mathbf{x}_t, t)$ is the reconstructed $\mathbf{x}_0$ at step t.

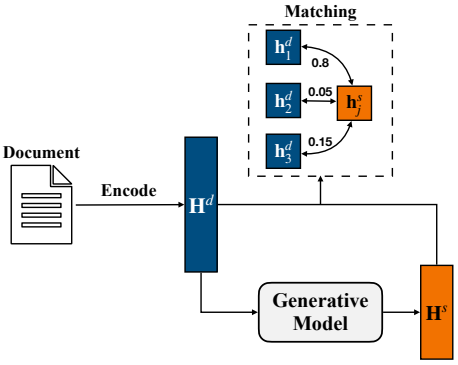

Figure 1: The proposed generation-enhanced extractive summarization framework. The model first conditionally generates desired summary embeddings and then extracts sentences based on representation matching.

3.2 Problem Formulation

Given a document with n sentences as $D = \{s_1^d, s_2^d, ..., s_n^d\}$, extractive summarization system aims to form a $m(m \ll n)$ sentences summary $S = \{s_1^s, s_2^s, ..., s_m^s\}$ by directly extracting sentences from the source document. Most existing work formulates it as sequence labeling and gives each sentence a $\{0, 1\}$ label, where label 1 indicates that the sentence will be included in summary S. Since extractive ground-truth labels (ORACLE) are not available for human-written gold summary, it is common to use a greedy algorithm to generate an ORACLE consisting of multiple sentences which maximize the ROUGE-2 score against the gold summary following (Nallapati et al., 2017).

In contrast, we propose a summary-level framework with generative model augmentation as shown in Figure 1. Formally, we train a diffusion model with the *reverse process* $p_\theta(\tilde{\mathbf{H}}_{t-1}^s|\tilde{\mathbf{H}}_t^s, \mathbf{H}^d)$ to directly generate the desired summary sentence representations $\tilde{\mathbf{H}}_{t-1}^s = [\tilde{\mathbf{h}}_1^s, \tilde{\mathbf{h}}_2^s, ..., \tilde{\mathbf{h}}_m^s] \in \mathbb{R}^{m \times h}$, where $\tilde{\mathbf{h}}_j^s$ is the vector representing the j-th summary sentence at diffusion step $t-1$. The model then extracts summary sentences based on the matching between the generated summary sentence representations after T reverse steps $\tilde{\mathbf{H}}_0^s = [\tilde{\mathbf{h}}_1^s, \tilde{\mathbf{h}}_2^s, ..., \tilde{\mathbf{h}}_m^s]$ and the document sentence embeddings $\mathbf{H}^d = [\mathbf{h}_1^d, \mathbf{h}_2^d, ..., \mathbf{h}_n^d]$. The matching score for the j-th sentence in the output s_j^s with the document is defined as:

$$\tilde{\mathbf{y}}_j = \text{softmax}(\tilde{\mathbf{h}}_j^s \cdot \mathbf{H}^{d^T}). \quad (4)$$

Here we use dot product as similarity measurement and then extract the sentence with the highest matching score for each generated summary sentence.

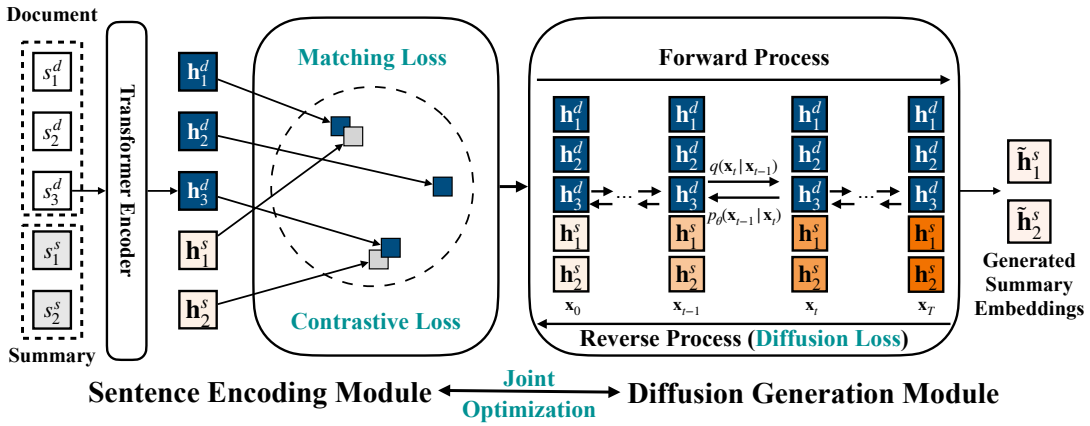

Figure 2: The overall architecture of DiffuSum. The input document is passed to the sentence encoding module and the diffusion generation module. DiffuSum will generate the desired summary sentence representations for inference.

Our framework operates on the summary level by generating all summary sentence representations simultaneously and adopts continuous diffusion models here for sentence embedding generation.

4 Method

In this section, we introduce the detailed design of DiffuSum. DiffuSum consists of two major modules: a sentence encoding module and a diffusion module, which will be introduced in Section 4.1 and Section 4.2, respectively. After that, we explain how we optimize our model and conduct inference in Section 4.3. The overall model architecture of DiffuSum is also illustarted in Figure 2.

4.1 Sentence Encoding Module

In order to generate desired summary sentence embeddings, we first build a contrastive sentence encoding module to transfer discrete text inputs $D = \{s_1^d, s_2^d, ..., s_n^d\}$ into continuous vector representations $\mathbf{H}^d = [\mathbf{h}_1^d, \mathbf{h}_2^d, ..., \mathbf{h}_n^d] \in \mathbb{R}^{n \times h}$, where h is the dimension of the encoded sentence representations.

Specifically, we first obtain the initial representations of sentences $\mathbf{E}^d = [\mathbf{e}_1^d, \mathbf{e}_2^d, ..., \mathbf{e}_n^d]$ with Sentence-BERT (Reimers and Gurevych, 2019). Note that the Sentence-BERT is only used for initial sentence embedding, but is not updated during training. The initial representations are then fed into a stacked transformer layer followed by a projection layer to obtain contextualized sentence representations $\mathbf{h}_i^d$:

$$\mathbf{h}_i^d = \text{MLP}(\text{Transformer}(\mathbf{e}_i^d)). \quad (5)$$

The same encoding process is applied to the summary sentences $S = \{s_1^s, s_2^s, ..., s_m^s\}$ to obtain encoded summary sentence representations $\mathbf{H}^s = [\mathbf{h}_1^s, \mathbf{h}_2^s, ..., \mathbf{h}_m^s] \in \mathbb{R}^{m \times h}$. The encoded document sentence representations $\mathbf{H}^d$ and summary sentence representations $\mathbf{H}^s$ are then concatenated as $\mathbf{H}^{in} = \mathbf{H}^d || \mathbf{H}^s \in \mathbb{R}^{(n+m) \times h}$ and will be passed to the diffusion generation module.

To ensure the sentence encoding module produces accurate and distinguishable representations, we introduce a matching loss $\mathcal{L}_{\text{match}}$ and a multi-class supervised contrastive loss $\mathcal{L}_{\text{contra}}$ to optimize the module, which are defined as follows.:

Matching Loss We first introduce a matching loss to ensure an accurate matching between the encoded document and summary sentence representations. Formally, for the j-th encoded summary sentence representation $\mathbf{h}_j^s$, we generate its encoding matching scores $\hat{\mathbf{y}}_j$ by computing the dot product with document representations followed by a softmax function:

$$\hat{\mathbf{y}}_j = \text{softmax}(\mathbf{h}_j^s \cdot \mathbf{H}^{d^{\mathrm{T}}}). \quad (6)$$

Then we have the encoding matching loss $\mathcal{L}_{\text{match}}$ as the cross-entropy between our encoding matching score $\hat{\mathbf{y}}_j$ and the ground truth extractive summarization label (ORACLE) $\mathbf{y}_j$:

$$\mathcal{L}_{\text{match}} = \sum_{j=1}^{m} \text{CrossEntropy}\left(\mathbf{y}_j, \hat{\mathbf{y}}_j\right). \quad (7)$$

Contrastive Loss The sentence encoding module also needs to ensure the encoded summary sentence embeddings $[\mathbf{h}_1^s, \mathbf{h}_2^s, ..., \mathbf{h}_m^s]$ are diverse and distinguishable. Thus, we introduce the multi-class supervised contrastive loss (Khosla et al., 2020) to push the summary sentence representation closer

to its corresponding document sentence representation while keeping it away from other sentence embeddings.

Given the sentence contextual representations $\mathbf{H}^{in} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_{n+m}] \in \mathbb{R}^{(n+m)\times h}$, the contrastive label $\mathbf{y}^c$ is defined as:

$$y_p^c = \begin{cases} q, \text{ if } p \leq n \text{ and } s_p^d = s_q^s \\ q, \text{ if } p = n + q \\ 0, \text{ otherwise} \end{cases}, \quad (8)$$

where $q \in \{1, 2, \cdots, m\}$ and y_p^c is the p-th element of $\mathbf{y}^c$. The contrastive loss $\mathcal{L}_{\text{contra}}$ is defined as:

$$\begin{aligned} \mathcal{L}_{\text{contra}} &= \frac{-1}{2N_{y_p^c} - 1} \sum_{p=1}^{n+m} \mathcal{L}_{\text{contra}}^p, \\ \mathcal{L}_{\text{contra}}^p &= \sum_{\substack{q=1;q\neq p;\\ y_q^c = y_p^c}}^{n+m} \log \frac{\exp\left(\mathbf{h}_p \cdot \mathbf{h}_q^T / \tau\right)}{\sum_{k=1;p\neq k}^{n+m} \exp\left(\mathbf{h}_p \cdot \mathbf{h}_k^T / \tau\right)}, \end{aligned} \quad (9)$$

where $N_{y_p^c}$ is the total number of sentences in the document that have the same label y_p^c ($N_{y_p^c} = 2$ in our case) and τ is a temperature hyperparameter.

The overall optimizing objective for the sentence encoding module $\mathcal{L}_{\text{se}}$ is defined as:

$$\mathcal{L}_{\text{se}} = \mathcal{L}_{\text{match}} + \gamma \mathcal{L}_{\text{contra}}, \quad (10)$$

where γ is a rescale factor that adjusts the diversity of the sentence representations.

4.2 Diffusion Generation Module

After obtaining the input encoding $\mathbf{H}^{in} = \mathbf{H}^d \| \mathbf{H}^s$, we adopt the continuous diffusion model to generate desired summary sentence embeddings conditionally. As described in Section 3.1, our diffusion generation module adds Gaussian noise gradually through the forward process and fits a stacked Transformer to invert the diffusion in the reverse process.

We first perform one-step Markov transition $q(\mathbf{x}_0|\mathbf{H}^{in}) = \mathcal{N}\left(\mathbf{H}^{in}, \beta_0 \mathbf{I}\right)$ for the starting state $\mathbf{x}_0 = \mathbf{x}_0^d \| \mathbf{x}_0^s$. Note that the initial Markov transition is applied to both document and summary sentence embeddings.

We then start the forward process by gradually injecting partial noise to summary embeddings $\mathbf{x}^s$ and leaving document embeddings unchanged $\mathbf{x}^d$ similar to (Gong et al., 2022). This enables the diffusion model to generate conditionally on the source document. At step t of the forward process $q\left(\mathbf{x}_t^s|\mathbf{x}_{t-1}\right)$, the noised representations is $\mathbf{x}_t$:

$$\mathbf{x}_t = \mathbf{x}_0^d || \mathcal{N}\left(\mathbf{x}_t^s; \sqrt{1-\beta_t}\mathbf{x}_{t-1}^s, \beta_t \mathbf{I}\right), \quad (11)$$

where $t \in \{1, 2, \cdots, T\}$ for a total of T diffusion steps and $\|$ represents concatenation.

Once the partially noised representations are acquired, we conduct the backward process to remove the noise of summary representations given the condition of the sentence representations of the previous step:

$$p_\theta\left(\mathbf{x}_{t-1}^s|\mathbf{x}_t\right) = \mathcal{N}\left(\mathbf{x}_{t-1}^s; \mu_\theta\left(\mathbf{x}_t, t\right), \sigma_\theta^2\left(t\right)\mathbf{I}\right), \quad (12)$$

where $\mu_\theta(\cdot)$ and $\sigma_\theta^2(\cdot)$ are parameterized models (stacked Transformer in our case) to predict the mean and standard variation at diffusion step $t-1$. The final output of the diffusion module is the generated summary sentence representations after T reverse steps $\tilde{\mathbf{H}}_0^s = [\tilde{\mathbf{h}}_1^s, \tilde{\mathbf{h}}_2^s, ..., \tilde{\mathbf{h}}_m^s]$. We optimize the diffusion generation module with diffusion loss $\mathcal{L}_{\text{diffusion}}$ defined as:

$$\begin{aligned} \mathcal{L}_{\text{diffusion}} = \sum_{t=2}^{T} \left\| \mathbf{x}_0 - \tilde{f}_\theta\left(\mathbf{x}_t, t\right) \right\|^2 + \\ \left\| \mathbf{H}^{in} - \tilde{f}_\theta\left(\mathbf{x}_1, 1\right) \right\|^2 + \mathcal{R}\left(\mathbf{x}_0\right), \end{aligned} \quad (13)$$

where $\tilde{f}_\theta\left(\mathbf{x}_t, t\right)$ is the reconstructed $\mathbf{x}_0$ at step t and $\mathcal{R}\left(\mathbf{x}_0\right)$ is a L-2 regularization term.

4.3 Optimization and Inference

We jointly optimize the sentence encoding module and the diffusion generation module in an end-to-end manner. The overall training loss of DiffuSum can be represented as:

$$\mathcal{L} = \mathcal{L}_{\text{se}} + \eta \mathcal{L}_{\text{diffusion}} \quad (14)$$

where η is a balancing factor of sentence encoding module loss $\mathcal{L}_{\text{se}}$ and diffusion generation module loss $\mathcal{L}_{\text{diffusion}}$.

For inference, DiffuSum first obtains encoded document representations $\mathbf{H}^d$, followed by a one-step Markov transition $q(\mathbf{x}_0^d|\mathbf{H}^d)$. Then we random sample m Gaussian noise embeddings as initial summary sentence representations $\mathbf{x}_T^s \in \mathbb{R}^{m\times h}$ and concatenate it with document representations to get the input $\mathbf{x}_T^{in} = \mathbf{x}_0^d || \mathbf{x}_T^s$ for diffusion step T. Then DiffuSum applies the learned reverse process (generation process) to remove the Gaussian noise iteratively and get the output summary sentence representations $\tilde{\mathbf{H}}_0^s = [\tilde{\mathbf{h}}_1^s, \tilde{\mathbf{h}}_2^s, ..., \tilde{\mathbf{h}}_m^s]$.

DiffuSum then calculates the matching between the generated summary representation $\tilde{\mathbf{h}}_i^s$ and the document representation $\mathbf{H}^d$ to obtain prediction

Dataset	Domain	Doc #words	Sum #words	#Ext
CNN/DM	News	766.1	58.2	3
XSum	News	430.2	23.3	2
PubMed	Paper	444	209.5	6

Table 1: Statistics of the experimental datasets. Doc # words and Sum # words refer to the average word number in the source document and summary. # Ext refers to the number of sentences to extract.

label $\tilde{\mathbf{y}}_i^{pred}$ as in Eq. 4. We extract the sentence with the highest score for each generated summary sentence representation and form the summary.

5 Experiment

5.1 Experimental Setup

Datasets We conduct experiments on three benchmark summarization datasets: *CNN/DailyMail*, *XSum*, and *PubMed*. CNN/DailyMail (Hermann et al., 2015) is the most widely-adopted summarization dataset that contains news articles and corresponding human-written news highlights as summaries. We use the non-anonymized version in this work and follow the common training, validation, and testing splits (287,084/13,367/11,489). XSum (Narayan et al., 2018a) is a one-sentence news summarization dataset with all summaries professionally written by the original authors of the documents. We follow the common training, validation, and testing splits (204,045/11,332/11,334). PubMed (Cohan et al., 2018) is a scientific paper summarization dataset of long documents. We follow the setting in (Zhong et al., 2020) and use the introduction section as the article and the abstract section as the summary. The training/validation/testing split is (83,233/4,946/5,025). The detailed statistics of each dataset are shown in Table 1.

Baselines We compare DiffuSum with strong sentence-level baseline methods: the vanilla Transformer (Vaswani et al., 2017), hierarchical encoder model HIBERT (Zhang et al., 2019), PNBERT (Zhong et al., 2019) that combines LSTM-Pointer with pre-trained BERT, BERT-based extractive model BERTSum (Liu and Lapata, 2019), and BERTEXT (Bae et al., 2019) that augments BERT with reinforcement learning,.

We also compare DiffuSum with state-of-the-art summary-level approaches: contrastive Learning based re-ranking framework COLO (An et al., 2023) and summary-level two-stage text matching framework MATCHSUM (Zhong et al., 2020).

5.2 Implementation Details

We use Sentence-BERT (Reimers and Gurevych, 2019) checkpoint *all-mpnet-base-v2* for initial sentence representations. The dimension of the sentence representations h is set to 128. We use an 8-layer Transformer with 12 attention heads in our sentence encoding module and a 12-layer Transformer with 12 attention heads in the diffusion generation module. The hidden size of the model is set to 768, and temperature τ is set to 0.07. The scaling factors γ and η are set to 0.001 and 100, where γ is searched in the range of $[0.0001, 1]$ and η is searched within the range of $[10, 1000]$. We set the diffusion steps T to 500. Effects of hyperparameter T and h are discussed in section 6.2.

DiffuSum has a total of 13 million parameters and is optimized with AdamW optimizer (Loshchilov and Hutter, 2017) with a learning rate of $1e^{-5}$ and a dropout rate of 0.1. We train the model for 10 epochs and validate the performance by the average of ROUGE-1 and ROUGE-2 F-1 scores on the validation set.

Following the standard setting, we evaluate model performance with ROUGE[1] F-1 scores (Lin and Hovy, 2003). Specifically, ROUGE-1/2 scores measure summary informativeness, and the ROUGE-L score measures summary fluency. Single-run results are presented in the following sections with the default random seed of 101.

5.3 Experiment Results

Results on CNN/DailyMail Experimental results on CNN/DailyMail dataset are shown in Table 2. The first block in the table contains the extractive ground truth ORACLE (upper bound) and LEAD that selects the first few sentences as a summary. The second block includes recent strong one-stage extractive baseline methods and our proposed model DiffuSum. The third section includes two-stage baseline methods that pre-select salient sentences. We follow the same setting and show the results of DiffuSum with the same pre-selection for a fair comparison.

According to the results, DiffuSum achieves *new state-of-the-art* performance under both one-stage and two-stage settings, especially a large raise in the ROUGE-2 score. The supreme per-

[1]ROUGE: https://pypi.org/project/rouge-score/

Model	R-1	R-2	R-L
LEAD	40.43	17.62	36.67
ORACLE	52.59	31.23	48.87
One-stage Systems			
Transformer (2017)	40.90	18.02	37.17
HIBERT* (2019)	42.37	19.95	38.83
PNBERT* (2019)	42.69	19.60	38.85
BERTEXT* (2019)	42.76	19.87	39.11
BERTSum* (2019)	43.85	20.34	39.90
$\text{COLO}^{*}_{\text{Ext}}$ (2023)	44.58	21.25	40.65
DiffuSum (ours)	**44.62**	**22.51**	**40.34**
Two-stage Systems			
MATCHSUM*(BERT) (2020)	44.22	20.62	40.38
MATCHSUM*(Roberta)	44.41	20.86	40.55
DiffuSum (ours)	**44.83**	**22.56**	**40.56**

Table 2: Experimental results on CNN/DailyMail dataset. Models using pre-trained language models are marked with*.

Model	PubMed			XSum		
	R-1	R-2	R-L	R-1	R-2	R-L
ORACLE	45.12	20.33	40.19	25.62	7.62	18.72
LEAD	37.58	12.22	33.44	14.40	1.46	10.59
BERTSUM	41.05	14.88	36.57	22.86	4.48	17.16
MatchSUM	41.21	14.91	36.75	**24.86**	4.66	**18.41**
DiffuSum	**41.40**	**15.55**	**37.48**	24.00	**5.44**	18.01

Table 3: Experimental Results on PubMed and XSum datasets.

formance of DiffuSum demonstrates the efficacy of our generation-augmented framework and the great potential to apply diffusion models in text representation generation. It is worth noting that most baseline methods contain pre-trained language model components, but our proposed framework DiffuSum trains Transformers from scratch and contains no pre-trained knowledge. We believe DiffuSum would achieve even better performance if combining pre-trained knowledge, and leave it to future work. We also notice that summary-level methods generally outperform sentence-level methods, proving the need to fill the inherent gap.

Results on XSum and PubMed We also evaluate DiffuSum on PubMed and XSum datasets, representing datasets of different domains and summary lengths as shown in Table 3.

For data with longer summaries like PubMed, DiffuSum shows highly strong performance and outperforms state-of-the-art baselines. The strong performance proves that our model can tackle longer input contexts and complex generations. Our summary-level setting also benefits data with longer summaries by considering summary sentence dependencies.

For data with shorter summaries like XSum, DiffuSum also achieves comparable performance to SOTA approaches, with a significantly higher ROUGE-2 score. Short-summary data tend to be simpler for matching-based methods like MatchSum since the candidate pool is much smaller.

Overall, DiffuSum achieves a comparable or even better performance compared to pre-trained language model-based baseline methods. The results demonstrate the effectiveness of DiffuSum on summarization data with different lengths.

Model	R-1	R-2	R-L
DiffuSum	**44.83**	**22.56**	**40.56**
w/o Sentence-BERT	43.53	21.63	40.23
w/o ORACLE	39.19	17.12	34.38
w/o Contrastive Loss	44.57	22.35	40.34

Table 4: Ablation study results on CNN/DailyMail dataset.

6 Analysis

6.1 Ablation Study

To understand the strong performance of DiffuSum, we perform an ablation study by removing model components of the sentence encoding module and show the results in Table 4. The second row shows that performance drops when replacing the initial sentence representation from Sentence-BERT to BERT-base encoder(Devlin et al., 2018). The performance drop indicates sentence-level information is necessary for the success of DiffuSum. The third row shows that replacing ORACLE with abstractive reference summaries degrades performance. As for the sentence encoding loss, both the matching loss and contrastive loss benefit the overall model performance according to rows 4 and 5. The matching loss is critical to the model, and the performance drops dramatically by more than 40% without it. The results prove the importance of jointly training a sentence encoder that produces accurate and diverse sentence representations with the generation module..

Model	R-1	R-2	R-L
DiffuSum(T=500, h=128)	**44.83**	**22.56**	**40.58**
DiffuSum(T=500, h=64)	43.36	21.27	39.89
DiffuSum(T=500, h=256)	44.53	22.49	40.27
DiffuSum(T=50, h=128)	42.60	19.71	38.96
DiffuSum(T=100, h=128)	44.61	22.24	40.32
DiffuSum(T=1000, h=128)	44.65	22.36	40.37
DiffuSum(T=2000, h=128)	44.64	22.37	40.40

Table 5: The performance of DiffuSum with different hyperparameter settings on CNN/DM dataset.

Train \ Test	CNN/DM	XSum	PubMed
CNN/DM	44.83/22.56	21.35/3.85	39.83(-1.57)/13.25
XSum	42.85/21.37	24.0/5.44	38.71(-2.69)/12.93

Table 6: ROUGE-1 and ROUGE-2 results for cross-dataset evaluation.

6.2 Hyperparameter Sensitivity

We also study the influence of our diffusion generation module's two important hyperparameters: diffusion steps T and the sentence representations dimension h in Table 5. The first row is our best model, and the second block shows the performance of DiffuSum with different sentence representation dimensions. The performance drops by a large margin when setting the dimension to 64, indicating severe information loss when shrinking the sentence dimension too much. The performance also drops a little when the dimension is set to 256, suggesting that a too-large dimension may bring in more noise. The third block shows the influence of diffusion steps, where we find that model performance first increases with more diffusion steps, then starts to decrease and oscillate if steps keep increasing. We argue that the noise injected in the forward pass cannot be fully removed if the steps are too small, and the model will introduce too much noise to recover if the steps are too big.

6.3 Cross-dataset Evaluation

We also notice that DiffuSum shows a strong cross-dataset adaptation ability. As shown in Table 6, the model trained on the news domain (CNN/DM and XSum) achieves comparable performance (only 1.57 and 2.69 ROUGE-1 drops) when directly tested on the scientific paper domain. The cross-dataset results demonstrate the robustness of our generation-augmented framework and the potential to build a generalized extractive summarization system.

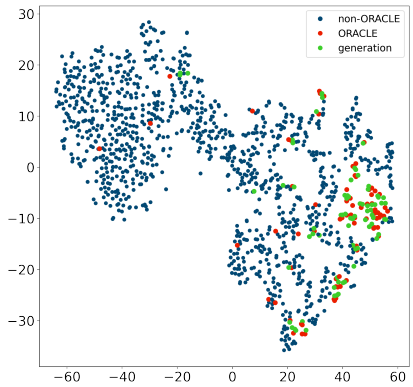

Figure 3: T-SNE visualization of sentence embeddings from 25 CNN/DM dataset documents.

6.4 Representation Analysis

We also analyze the generated sentence representation quality. We apply T-SNE (van der Maaten and Hinton, 2008) to reduce the sentence representation's dimension to 2 and show the encoded sentence representations as well as the generated summary sentence representations in Figure 3. The blue dots in the figure represent non-summary sentences, and the red dots represent summary sentences (ORACLE) from our sentence encoding module. The green dots are summary sentence representations reconstructed by our diffusion generation module. We can find that most of the ORACLE sentences gather on the right. This finding proves that our contrastive encoder could distinguish ORACLE sentences from non-summary sentences. We also find that the sentence representations generated by the diffusion module (green) are very close to the original summary representations (red). The finding demonstrates that our diffusion generation module is powerful in reconstructing sentence representations from random Gaussian noise.

7 Conclusions

This paper proposes a new paradigm for extractive summarization with generation augmentation. Instead of sequentially labeling sentences, DiffuSum directly generates the desired summary sentence representations with diffusion models and extracts summary sentences based on representation matching. Experimental results on three benchmark datasets prove the effectiveness of DiffuSum. This work is the first attempt to adapt diffusion models for summarization. Future work could explore various ways of applying continuous diffusion models to both extractive and abstractive summarization.

Limitations

Despite the strong performance of DiffuSum, its design still has the following limitations. First, DiffuSum is only designed for extractive summarization, and the diffusion generation module only generates sentence embeddings instead of token-level information. Thus, it is not applicable to the abstractive summarization setting. Moreover, DiffuSum is only tested on single document summarization datasets. How to adapt DiffuSum for multi-document and long document summarization scenarios need further investigation. In addition, our generative model involves multiple steps of noise injection and denoising, compared to discriminator-based extractive systems.

Acknowledgement

This work is supported by NSF through grants IIS-1763365 and IIS-2106972. We also thank anonymous reviewers for their helpful feedback.

References

Chenxin An, Ming Zhong, Zhiyong Wu, Qin Zhu, Xuanjing Huang, and Xipeng Qiu. 2023. Colo: A contrastive learning based re-ranking framework for one-stage summarization.

Ben Athiwaratkun, Cicero Nogueira dos Santos, Jason Krone, and Bing Xiang. 2020. Augmented natural language for generative sequence labeling. *arXiv preprint arXiv:2009.13272*.

Sanghwan Bae, Taeuk Kim, Jihoon Kim, and Sanggoo Lee. 2019. Summary level training of sentence rewriting for abstractive summarization. *arXiv preprint arXiv:1909.08752*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. *arXiv preprint arXiv:1804.05685*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. 2022. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. All nlp tasks are generation tasks: A general pretraining framework. *arXiv preprint arXiv:2103.10360*.

Elozino Egonmwan and Yllias Chali. 2019. Transformer-based model for single documents neural summarization. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 70–79.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. 2022. DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models. ArXiv:2210.08933 [cs].

Som Gupta and Sanjai Kumar Gupta. 2019. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121:49–65.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. 2022. Video diffusion models. *arXiv preprint arXiv:2204.03458*.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2020. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*.

Wojciech Kryściński, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Evaluating the factual consistency of abstractive text summarization. *arXiv preprint arXiv:1910.12840*.

Haoran Li, Junnan Zhu, Jiajun Zhang, Chengqing Zong, and Xiaodong He. 2020. Keywords-guided abstractive sentence summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8196–8203.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. 2022. Diffusion-LM Improves Controllable Text Generation. ArXiv:2205.14217 [cs] version: 1.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence

statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-first AAAI conference on artificial intelligence*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, et al. 2022. Self-conditioned embedding diffusion for text generation. *arXiv preprint arXiv:2211.04236*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. *arXiv preprint arXiv:2004.12393*.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Discourse-aware neural extractive model for text summarization. *arXiv preprint arXiv:1910.14142*.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various ner subtasks. *arXiv preprint arXiv:2106.01223*.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2022. Diffusion Models: A Comprehensive Survey of Methods and Applications. ArXiv:2209.00796 [cs] version: 7.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. 2022. Seqdiffuseq: Text diffusion with encoder-decoder transformers. *arXiv preprint arXiv:2212.10325*.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2022a. Hegel: Hypergraph transformer for long document summarization. *arXiv preprint arXiv:2210.04126*.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023. Extractive summarization via chatgpt for faithful summary generation.

Haopeng Zhang, Semih Yavuz, Wojciech Kryscinski, Kazuma Hashimoto, and Yingbo Zhou. 2022b. Improving the faithfulness of abstractive summarization via entity coverage control.

Haopeng Zhang and Jiawei Zhang. 2020. Text graph transformer for document classification. In *Conference on Empirical Methods in Natural Language Processing*.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. *arXiv preprint arXiv:1905.06566*.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. *arXiv preprint arXiv:2004.08795*.

Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2019. Searching for effective neural extractive summarization: What works and what's next. *arXiv preprint arXiv:1907.03491*.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. *arXiv preprint arXiv:1807.02305*.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
The Limitations Section

☒ A2. Did you discuss any potential risks of your work?
Our paper proposes a summarization model and we experiment with public datasets. The model will only summarize documents and has no potential risk.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1 Introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 5

☑ B1. Did you cite the creators of artifacts you used?
Section 5

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
We only use public available data and models.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 5

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 5

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 5.1

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 5.2

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Sections 5.2 and 6.2

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5.2

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 5.2

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.