Generation of Synthetic and Private Medical Texts from Electronic Healthcare Records

Technological advancements in data science have offered us affordable storage and efficient algorithms to query a large volume of data. Our health records are a significant part of this data, which is pivotal for healthcare providers and can be utilized in our well-being. The clinical note in Electronic Health Records (EHRs) is one such category that collects a patient's complete medical information during different timesteps of patient-care available in the form of free-texts. Thus, these unstructured textual notes contain events from a patient's admission to discharge, which can prove to be significant for future medical decisions. However, since these texts also contain sensitive information about the patient and the attending medical professionals, such notes cannot be shared publicly. This privacy issue has thwarted timely discoveries on this plethora of untapped information. Therefore, in this work, we intend to generate synthetic medical texts from a private or sanitized (de-identified) clinical text corpus and analyze their utility rigorously in different metrics and levels. Experimental results promote the applicability of our generated data as it achieves more than 80% accuracy in different pragmatic classification problems and matches (or outperforms) the original text data.

CCS Concepts: • **Security and privacy → Privacy-preserving protocols**; **Pseudonymity, anonymity and untraceability**; • **Computing methodologies → Machine learning algorithms**; • **Information systems** → *Language models.*

Additional Key Words and Phrases: Electronic Health Records, Generate EHRs, Privacy preserving EHR Generation

ACM Reference Format:
. 2020. Generation of Synthetic and Private Medical Texts from Electronic Healthcare Records. *ACM Trans. Comput. Healthcare* X, X, Article X (X 2020), 23 pages. https://doi.org/X

1 INTRODUCTION

The systematic adoption of digital data collection in healthcare has resulted in an abundance of Electronic Healthcare Records (EHRs) [1]. Unstructured texts, among these EHRs, are a piece of vital information as it describes a patient's hospital visit, his/her past and present history of illness, medications, assigned physicians, and other medical details. These free-texts are named clinical note [2] as medical professionals type them during the patient's treatment. Nevertheless, these unstructured textual clinical notes contain medically relevant information that can be utilized to reduce cost, investigate human errors, and ultimately provide better healthcare.

However, clinical notes are highly sensitive as they contain private information about an individual [3]. In Canada, Personal Information Protection and Electronic Documents Act (PIPEDA) [4] restrict any public access to documents containing Personally Identifiable Information (PII) or Protected Health Information (PHI) such as name, date of birth, etc. Similarly, in the USA, the Health Insurance Portability and Accountability Act (HIPAA) [5] requires removing all PII/PHI before any dissemination [6]. Furthermore, access to such databases demands consent forms, certifications, and lengthy administrative processes impeding timely scientific achievements [7].

One common approach for privacy-preserving EHR publishing is *de-identification* , which detects and removes the PII from EHRs before the public dissemination. Throughout the last two decades, a considerable amount

Author's address:

2637-8051/2020/X-ARTX $15.00
https://doi.org/X

of effort has been devoted to the de-identification problem. At first, expert human annotators tried to find sensitive information manually in the EHRs and remove them, which unfortunately turned out to be both costly and error-prone [8–10]. Although the subsequent automated systems involving regular expressions and machine learning algorithms were somewhat an improvement over the manual approaches, these still were very much dataset-specific. In 2016, the first neural network-based de-identification system was proposed by Dernoncourt *et al.* [11], and as of today, it is regarded as the state-of-the-art model for de-identification . The neural network-based approach achieves more accuracy than the previous automated or semi-automated methods without any dataset dependency or manual feature extraction.

On paper, a state-of-the-art model is a reasonable approach to the de-identification problem as the reported recall value is over 97%. In other words, 3% PII tokens will be released erroneously, which admittedly is a high recall value for any neural network-based model. Unfortunately, the 3% inaccuracy reported bears more significance than the 97% accuracy. A 97% accuracy signifies that it cannot detect three PII instances for every hundred PII instances. These three PII instances in the hands of a worthy adversary can lead to the re-identification of the patients, consequently rendering the 97% accuracy ineffective. Therefore, a permissible error from a de-identification model should be lower than 1% [12]. Nevertheless, ideally, we need a neural network model with 99% recall value (or more). Unfortunately, until now, despite multiple reported attempts, none managed to achieve this feat.

Now, the inadequacy of the current de-identification approaches curbed their adoption in real-world EHR publishing and limited the number of publicly available datasets. Rather than being utilized in medical research, most of the electronic clinical notes collected since 2009 are only stored and left unexplored; hence, approaching the problem from a new standpoint has become imperative. Irrespective of the computational de-identification techniques, there is another way to attain both privacy and publish clinical notes for potential medical research. We can generate synthetic clinical notes from a subset of de-identified data collected from the whole corpus. Notably, such de-identification can be done using human expertise or computational methods, depending on the required privacy or expenditures.

Recent advancements in machine learning allow us to generate synthetic data from real datasets, representing the original records' statistical properties. Here, the synthetic text data generated using the deep learning techniques is probabilistic and exhibit similar patterns (e.g., medications, disease association) as the input textual dataset but withhold sensitive information. Furthermore, with a privacy-preserving technique such as differential privacy, we can generate an arbitrary number of private records that can be disseminated for data analysis. The privacy risks in publishing these newly generated texts will be lower since they are generated from a mechanism with a standard privacy guarantee and do not include any PII elements. Nevertheless, the utility of the private and machine-generated synthetic data needs to be comparable to the original dataset. For example, the physical symptoms or corresponding medications in this dataset need to be similar to show similar trends as the original data.

In this work, we propose a differentially private neural network-based model for generating synthetic EHRs that satisfy both the privacy and utility requirements (as mentioned above). Importantly, we generate synthetic unstructured medical texts under two different privacy models: 1) from de-identified data where the PII tokens were never present in the input data (sanitized), and 2) generate texts with a differential privacy guarantee. Here, in both cases, the resulting text dataset will not contain any identifying information on any individual. However, we emphasize heavily on the usability of such machine-generated texts as it was the foremost reason behind such generated synthetic private data in the first place. The primary contributions of this article can be summarized as follows:

- We employ a self-attention based [13] generative neural network to create synthetic and private unstructured medical texts from multiple original and sanitized datasets (Section 3.1).

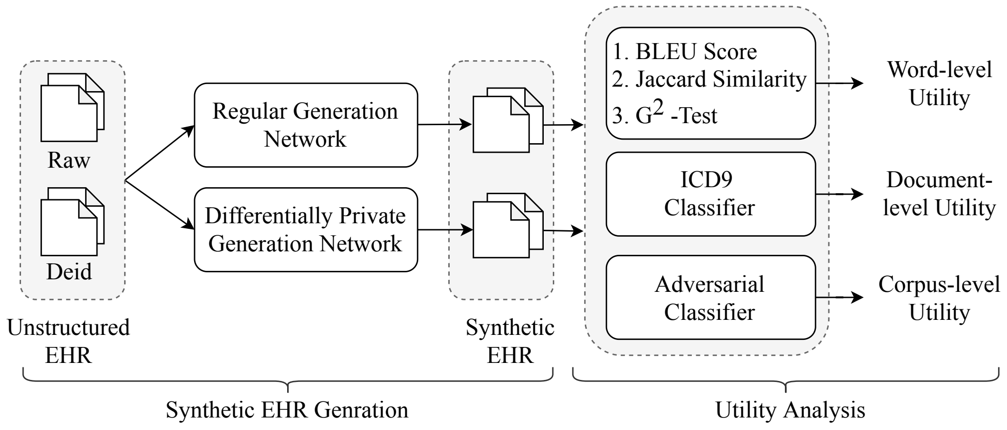


Fig. 1. Overview of the solution mechanism separating two different tasks: a) Generation, and b) Utility analysis

- Furthermore, a differentially private mechanism is incorporated with the generation technique to guarantee quantifiable privacy over the generated synthetic EHR text data (Section 3.1.3). To the best of our knowledge this is the first work on generating long medical texts using a differentially private mechanism.
- We rigorously analyzed the utility of the generated text corpora at three different levels (word, document, and corpus) and demonstrated that our generated synthetic EHR corpus could be utilized as a substitute of real-world datasets in applications such as medical document classifiers ((Section 3.2).
- The proposed approach is compared with another text generation model, SeqGAN [14]) and show the efficacy of our method in generating long unstructured medical documents. Experimental results show that the generated texts from our method outperform the existing method and deliver satisfactory results in all utility metric as it achieves BLEU-1 scores of a minimum 0.89 and more than 80% accuracy in different disease classifications and adversarial success (Section 4).

The paper is organized as follows: in Section 2, we elaborate on the problem at hand and an overview of our approach to the problem. In Section 3 we discuss our methods in detail. After that we present and discuss our experimental results in Section 4 and Section 5, respectively. We conclude our work with a discussion on the related works concerning synthetic text generation in Section 6 and conclude in Section 7.

2 PROBLEM DESCRIPTION

The primary objective of our work is to generate synthetic medical (EHR) texts which do not contain the original PII values but represent the other statistical properties from the original EHR texts. For example, our synthetic dataset needs to be privacy-preserving as it cannot have the name, date, or contact information, which can be used to re-identify a patient if published. However, the medically relevant information such as patient's underlying symptoms, disease diagnosis, or the medications at different timesteps will be coherent in these generated EHRs.

An overview of our work is depicted in Figure 1. We generate private synthetic EHR text data from a small subset of de-identified text corpus using deep learning techniques with a competitive utility. In other words, we have three separate tasks-(i) Generation of clinical texts from de-identified EHRs, (ii) Provide a quantifiable privacy guarantee over the generated data, and (iii) Utility analysis of these private and synthetic texts (Figure 1). In the following subsections, we briefly describe how we approach and evaluate these two tasks:

DISCHARGE SUMMARY
Patient Name: Mr. Noah Russell
Admission: 2012-3-1 **Discharge:** 2012-3-7
Date of Birth: 1921-03-17 **Sex:** M
History of Present Illness: Patient is a 91 yo man from Missoula, Montana with multiple medical problems including insulin dependent diabetes secondary to severe pancreatitis in 2006, remote history of Hodgkin's disease in 1996 treated with among other things, radiation therapy which has left the patient with severe osteoporosis and resulting compression fractures, history of alcohol abuse, ...
Discharge Medications:

- Lasix 20 mg p.o. q.d.
- Pantoprazole 40 mg p.o. q.d.

Discharge Diagnoses:

- Chronic obstructive pulmonary disease
- Congestive heart failure
- Insulin dependent diabetes

Follow-Up: He will follow-up with Dr. John in two weeks.

DISCHARGE SUMMARY
Patient Name: [*Name*]
Admission: [*Date*] **Discharge:** [*Date*]
Date of Birth: [*Date*] **Sex:** M
History of Present Illness: Patient is a [*90+*] yo man from [*Location*] with multiple medical problems including insulin dependent diabetes secondary to severe pancreatitis in [*Date*], remote history of Hodgkin's disease in [*Date*] treated with among other things, radiation therapy which has left the patient with severe osteoporosis and resulting compression fractures, history of alcohol abuse, ...
Discharge Medications:

- Lasix 20 mg p.o. q.d.
- Pantoprazole 40 mg p.o. q.d.

Discharge Diagnoses:

- Chronic obstructive pulmonary disease
- Congestive heart failure
- Insulin dependent diabetes

Follow-Up: He will follow-up with [*Name*] in two weeks.

Fig. 2. Sample of original (replaced with arbitrary PII tokens) (left) and de-identified Electronic Health Records (right)

2.1 Synthetic EHR Generation

Publishing the vast resources available in the textual EHRs is the primary motivation behind generating textual EHRs. This task will take raw unstructured de-identified EHRs as inputs and build a model that can output similar EHR texts (synthetic data). Here, the newly generated text should follow the original text's statistical properties, such as the correlation of prior medical history (and medications) with the patient's present conditions. For example, in Figure 2 the original discharge summary (left), the patient had congestive heart failure and was prescribed with Lasix (to treat fluid due to heart failure). As these relations are not sensitive and need to be preserved in the synthetic data, we hope to achieve similar trends in the generated data too. We further discuss this topic in Section 3.1.

2.2 Privacy-Preserving Generation

Since the contemporary state-of-the-art de-identification techniques do not perform convincingly in terms of finding the PII tokens, we incorporate a differentially private mechanism (Definition 3.1) on the synthetic data generation. With arguable de-identification techniques, resulting de-identified data as an input to the synthetic model might leak private information on the generated medical texts. Therefore, we adopted an additional mechanism to generate tokens with manageable noise, regulated by a strict differential privacy guarantee. Our primary goal here is to compare both of these generated texts (Section 3.1 and 3.1.3) in terms of various utility metrics which we describe next.

Table 1. HIPAA Safe Harbor Method defined 18 PIIs types [15]

No.	PHI Type	Description
1	Names	First, Last, Hospital Names
2	Location	Any geographic divisions smaller than a state
3	Dates	Birth date, admission or discharge date etc.
4	Contact	Home, office or cell phone numbers
5	Vehicle	Vehicle serial or license plate numbers
6	Fax	Fax information
7	Device	Device Identifiers and Serial Numbers
8	Email	Any electronic mail addresses
9	URLs	Web Universal Resource Locators
10	SSNs/SINs	Social Security or Insurance Number
11	MRNs	Medical Record Numbers
12	IP	Internet Protocol address
13	Biometric	All finger or voice-prints
14	Insurance	Health Plan Beneficiary Numbers
15	Photo	Full-face (or similar) photographic images
16	Accounts	Bank accounts, social media profile
17	Certificate	a License or certificate number
18	ID	Any unique identifying numbers

2.3 Utility Analysis

We use different methods and metrics to measure the utility of the generated EHRs. We can broadly categorize these utility metrics according to the level of granularity:

- The first utility metric operates on the *word-level*, where we compare the word frequency from the original and generated EHRs. This can also be termed micro-level analysis. We employ three standard techniques (Jaccard similarity, BLEU score comparison, and G^2-test) to analyze how the generated texts are different from the original de-identified texts.
- *Document-level* or our macro utility analysis will utilize a classification task to show the utility of the generated EHRs in a real-life application. Specifically, we will use a supervised machine learning task to train two separate ICD9 code [16] classifiers using the original and generated EHR documents, respectively, and compare their performance.
- Lastly, we perform a *Corpus-level* evaluation where we employ a binary classifier to distinguish or separate the original EHRs apart from the synthetic ones. It is similar to the Turing Test, which will utilize human trials and expertise to classify the text documents (whether synthetic or original). This metric will consider the original and generated documents as two different types and classify a random document. The better the quality of the synthetic document, the closer the classifier's accuracy to 50%.

Notably, our three utility metrics use different techniques, and these techniques are described in Section 3.2.

2.4 Privacy Model

The privacy of the synthetic texts is important as it should not reveal any of the patient information in the generated data. In this paper, we incorporate privacy-preserving techniques in two stages: 1) We first generate de-identified EHR data from the original textual data, and then 2) utilize a differentially private generation technique to produce a synthetic dataset. We define de-identification as an identifying information removal mechanism such that no particular individual can be linked back to the de-identified data. We follow the de-identification mechanism according to the HIPAA Safe Harbour Method [6], where there are 18 defined categories of Personally Identifiable Information (PII). All PIIs should be removed to de-identify any EHR. In this work, we adopt the HIPPA specified privacy model and detect, remove the PII elements from any EHR as defined in Table 1.

Since the original or raw EHRs contain PII tokens that can identify any individual (e.g., patient, caregivers), we need to utilize a de-identified mechanism that ensures their privacy. Specifically, we intend to generate de-identified synthetic EHRs, which will not contain any PII tokens as inputs in the first place. Therefore, we consider the original EHRs and de-identified EHRs separately as we generate synthetic medical texts for both corpora and analyze their utility.

However, as we cannot solely rely on the de-identified techniques on such large corpus, we also explore other methods to ensure the privacy of the generated data so that it cannot be linked back to any individual. Here, we use a differentially private mechanism, DP-SGD [17] incorporating in our generation technique by introducing a privacy budget ϵ, which operates as a tuning parameter between privacy and the quality of the generated texts (or utility). This parameter, ϵ also provides a quantifiable privacy guarantee over the generated synthetic data.

3 METHODS

3.1 Synthetic Text Generation

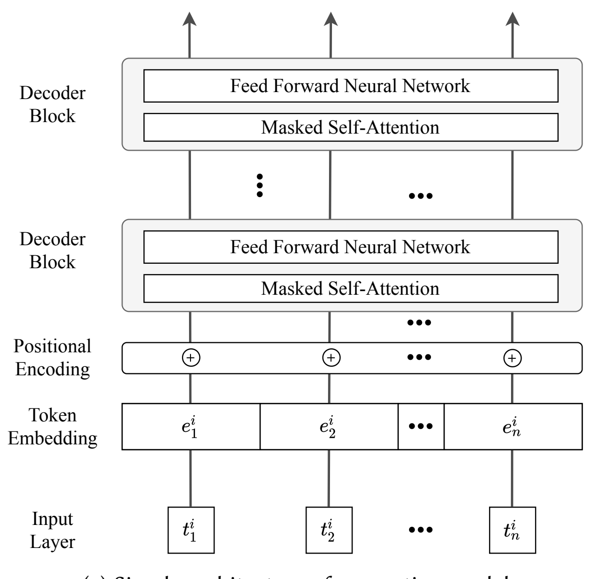
We consider two standard dataset: i2b2 2014 de-identification dataset [18] and MIMIC-III [19] while generating synthetic EHRs. These datasets were chosen due to the availability of PII annotations which were necessary for our targeted privacy model.

3.1.1 Input Data. Before synthetic EHR generation, we prepare two different text corpora from one dataset. In one set, we include all the private EHR records that contain all PII tokens that are sensitive and can be used to identify any individual. This dataset is referred to as original (or raw) data throughout the paper, which is not safe to publish. Consequently, the synthetic EHRs generated from this set are not publishable as they might contain sensitive PII tokens (i.e., names), In the other set, we create a corpus which can be a subset of the original dataset where the texts are de-identified according to the HIPAA privacy requirements. This dataset is named as *de-identified* (or sanitize) dataset in the rest of the paper. Notably, these PII annotations were categorized according to Table 1 employing human expertise.

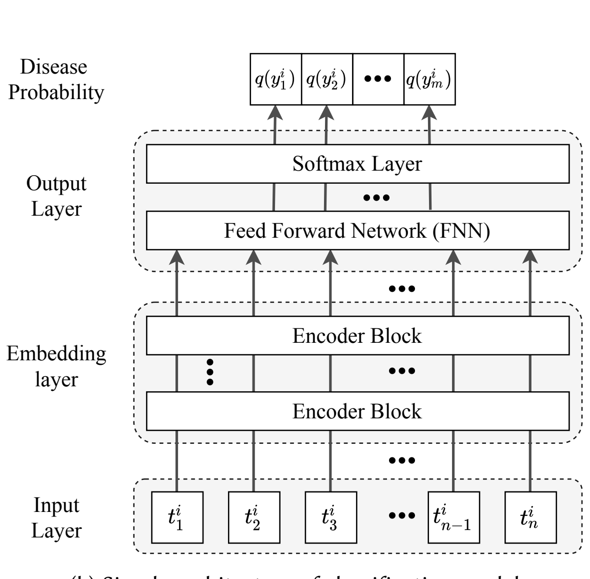
The two datasets (raw and de-identified) were fed into the generation model, indiscriminately. The documents were first broken into sequences (or sentences) and represented as a group of tokens (or words). For simplicity, we considered the maximum sequence length to be equal to the maximum number of tokens in the largest document in the corpus. These sequences are then forwarded to our generation model.

3.1.2 Generation Model. We ended up with two different EHR sets from the input layer, original and de-identified corpus, albeit the generation technique will remain the same for both corpora as we proceed to the utility analysis. The generation model receives the token sequences from the input layer as shown in Figure 3a, where, sequence $\{t_1^i, t_2^i, ..., t_n^i\}$ represents the i^{th} document with token length n.

The underlying generation depends on Language Modeling (LM) [20] technique, which gained popularity after recent breakthroughs in different NLP tasks [13, 21]. LM is an unsupervised task that is useful for training a large



(a) Simple architecture of generation model



(b) Simple architecture of classification model

Fig. 3. (a) $\{t_1^i, t_2^i, ..., t_n^i\}$ represents the i^{th} document with sequence length n. (b) Classification of the i^{th} document where $q(y_1^i), q(y_2^i), ..., q(y_m^i)$ are the prediction probability of m diseases.

corpus of unstructured text. Since the EHR texts are mostly a sequence of tokens, we can model each document i as conditional probabilities where we get the probability of j^{th} token t_j as [22]:

$$p(i) = \prod_{j=1}^{n}(p(t_j|t_1, \ldots, t_{j-1})) \tag{1}$$

Therefore, we can sample the next tokens from $p(t_{n-j}, \ldots, t_n|t_1, \ldots, t_{n-j-1})$ probability. Here, these probabilities are utilized in a deep learning model, which efficiently trains the sampling probability of each token in the vocabulary. Notably, for a large vocabulary, calculating the probability of each token from a massive corpus is computationally expensive. Therefore, we opted for a self-attention [23] based generation model as previous methods cannot process the tokens and their conditional probability in parallel and efficiently. The self-attention mechanism was first introduced in 2017. The original paper Vaswani *et al.* [23] described the Transformer model, which consists of an encoder stack and a decoder stack. Since the inception has been several variations to the Transformer model. In this work, we used an auto-regressive variation of the Transformer model. Similar to the GPT2 [13] model, this variation consists of only the decoder stack, as shown in Figure 3a.

As mentioned earlier, the token sequences are passed to the generation model. The token embeddings for each of these tokens are looked up in the pre-trained embedding matrix we used. In the Figure 3a, $\{e_1^i, e_2^i, \ldots, e_n^i\}$ represents the embedding for the token sequence $\{t_1^i, t_2^i, \ldots, t_n^i\}$, respectively for the i^{th} document. Before moving these embeddings to the first decoder block, a positional encoding is added to these tokens to indicate their order in the document/sequence. Each of the decoder blocks is composed of a multi-headed masked self-attention layer and a simple feed-forward network. In contrast to a self-attention layer, masked self-attention stops the model from seeing tokens that are at the right of the current position. This allows the model to take into account the last token when generating the next token. Hence, auto-regressive nature. For an input, 'Patient is a 91-year man from Missoula', we expect the model to output the next word 'Montana' (from Figure 2). Then, we add the current sample ('Montana') to the sentence and proceed to the next one. This auto-regression is the only similarity to the traditional generation model and the major difference with BERT [21], which uses a bi-directional approach while generating.

Now, the first decoder block passes the token embedding through the masked self-attention process to a feed-forward network. The resulting vector representation for each token is moved up through identical decoder blocks. Each decoder block, however, has its own weights, which is learned during training. At the final decoder block, the output vectors are multiplied by the embedding matrix. Each row in the embedding matrix corresponds to the embedding of a word. Therefore, the result of this multiplication is considered as the score for each word in the model's vocabulary. Even though selecting the token with the highest score would produce a reasonably competitive result, a better strategy is to sample a word from the entire list using the score as the probability of selecting that word. In our experiments, we set top_k to 40 and have the model consider the 40 words with the highest scores. This process describes one iteration, where each iteration generates a single word. The iteration is continued until the required sequence length is generated.

Furthermore, there was another condition in the generation relying on the type of EHR. We utilized the underlying disease or medical conditions as they were annotated according to the International Classification of Diseases (ICD9) specifications [16]. We shortlisted seven different diseases, and the documents from the original dataset were placed under each condition (more details in Section 3.2.2).

3.1.3 Differentially Private Generation. The private generation technique relies on a Differentially Private (DP) training method, which we incorporated with the GPT-2 [13]. In this method, the generation model was kept almost the same as we described earlier in Section 3.1. For example, the input layer, followed by the transformer-based generation model was the same. However, the optimization algorithm that calculates the difference between the prediction and the original to adjust the network's weights was altered to enforce our differential privacy guarantee. This difference is also known as the loss value, while the corresponding function, which calculates this difference known as the Loss Function.

We utilize Differential Privacy (DP) [24], which is a privacy mechanism offering theoretical and quantifiable bounds on the disclosure of data. Formally,

Definition 3.1. A randomized algorithm $\mathcal{A}$ is (ϵ, δ) differentially private if for a set of neighbouring datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, where $\mathcal{D}_1$ is different from $\mathcal{D}_2$ in at most in one record $(||\mathcal{D}_1 - \mathcal{D}_2|| \leq 1)$ and for all possible dataset created by $\hat{\mathcal{D}} \subseteq Range(\mathcal{A})$,

$$\mathcal{P}[\mathcal{A}(\mathcal{D}_1) \in \hat{\mathcal{D}}] \leq \exp(\epsilon) \times \mathcal{P}[\mathcal{A}(\mathcal{D}_2) \in \hat{\mathcal{D}}] + \delta,$$

where, $\mathcal{P}[\mathcal{A}(\mathcal{D}_1) \in \hat{\mathcal{D}}]$ denotes the probability of computing any function from $\mathcal{D}_1$.

Informally, the definition basically offers that with a predefined (ϵ, δ), the output to any query on the original dataset $\mathcal{D}_1$ may come from $\mathcal{D}_1$ or $\mathcal{D}_2$ with a probability ratio of $exp(\epsilon)$. Here, δ further reduces the probability of producing the output from the original dataset $\mathcal{D}_1$, which was fixed at 10^{-5}.

We utilize a differentially private Gaussian mechanism [25] in our optimizer. Here, we incorporate the DP parameters (ϵ, δ) and the dataset size while updating the network considering additional noise. Specifically, these noise values are calculated according to the Gaussian mechanism. For example, in every iteration, while querying for the original data to compute the loss function, a random value is added with the input. Here, each token from the text dataset is represented with a numeric vector of a fixed size (dynamic embedding) as they are perturbed by adding the random value. Therefore, the difference between the prediction and the input token (or loss) was different. The updates in the network had this effect as it generated differentially private tokens. Finally, the input layer of this network was different from the original network due to the noise, providing a privacy guarantee of (ϵ, δ).

3.1.4 Classification model. For the utility analysis, we will require multiple classifiers that provide accurate comparisons for our generated and original data. Analogous to the generation model, we proposed a generalized self-attention based classifier that operates similarly but classifies the documents rather than creating new ones. Furthermore, these classifiers are similar in architecture as they are only different at the very last layer based on the classification task. Notably, we utilized the self-attention based classifiers due to its simplicity of training and performance improvement over prior attempts based on recurrent neural networks or other machine learning approaches. The classifiers had the following layers:

Data Layer. The first step of the classifier is a data layer that splits every sentence from the input documents (clinical texts from EHRs) in the EHR corpus into sequences of tokens or words. Here, each word is assigned or indexed to a unique numeric value through tokenization, where a unique number represents a single token. Then these values are used to convert the sentences into a vector of numeric values. In summary, The data layer converts the entire corpus into a sequence matrix where each sequence is represented with a vector of unique numeric values assigned to each token. In Figure 3b, $t^{(i)} = \{t_1^{(i)}, t_2^{(i)}, t_3^{(i)}, \ldots, t_n^{(i)}\}$, is the i^{th} sequence in the input matrix and n is the maximum sequence length. For simplicity, we considered the maximum sequence length is equal to the maximum number of token in a document in the corpus.

Embedding Layer. Instead of a traditional fixed embedding like Word2vec [26] and GloVe [27], we used contextualized dynamic word embedding employing a multi-headed self-attention mechanism [1]. As mentioned earlier, the self-attention mechanism was first introduced by Vaswani *et al.* [23] in 2017. Prominent among the variations of the original transformer model is the BERT model, proposed by Devlin *et al.* [21]. BERT [21] uses only the encoder blocks from the transformer model, which could be called transformer-encoders. We used pre-trained weights for the initialization of the transformer-encoder parameters, and during training, the settings are fine-tuned on the training set. The construction details of the encoder blocks are described in Appendix A.1.1. This encoder block helps to calculate different vector representations for each token based on the use-cases and context. Hence, the name contextualized embedding. Here, the significant difference with the previous approach is this dynamic embedding available for each word.

Output layer. For this layer, we used a softmax function to calculate the maximum likelihood of the labels (ICD9 codes) of the document. The softmax function normalizes the values received from the Feed Forward Network (FNN) to probabilities. First, the matrix containing the context information received from the embedding layer is multiplied by a weight matrix and added to a bias in the FNN shown in the output layer of the Figure 3b. Both the weight matrix and the bias is learned during training. Let $x^{(i)}[l]$ be the output feature matrix of the embedding layer where i is the document number, and l is the encoder block number. b here is the bias vector, and its dimension is $1 \times m$ where m is the number of diseases. The output matrix for the i^{th} document, $y^{(i)}$ is calculated with Equation 2.

$$y^{(i)} = W_f^T x^{(i)}[l] + b \tag{2}$$

As shown in Figure 3b, the y matrix calculated in the FNN then forwarded to the softmax sub-layer, where the probability of each disease for the documents are calculated using Equation 3. Here as mentioned previously, i refers to the i^{th} document in the dataset and $q_j(y_j^{(i)})$ refers to the predicted probability of j^{th} disease for the i^{th}

[1]For continuity we omit the discussion on fixed and dynamic embeddings here and direct enthusiastic readers to Appendix A.1.1

document.

$$q_j(y_j^{(i)}) = \frac{\exp y_j^{(i)}}{\sum_m \exp y_m^{(i)}} \tag{3}$$

The dimension of matrix q is also $1 \times m$, where m is the total number of diseases in the dataset. The network is trained to minimize the cross-entropy loss, $\mathcal{L}(p, q)$ where p is the original probability matrix and q predicted probability matrix for the diseases related to the documents.

For the document-level utility analysis, we used a multi-label-classifier where the ICD9 codes related to the synthetic EHRs are predicted. For the corpus-level utility, we used classifiers with two and three output classes. The details of these two methods are described in the following two subsections.

3.2 Utility Evaluation

3.2.1 Word-level Utility. Our first utility metric operates on the token or word-level, where we compare the word frequency from the original and generated EHRs. In this micro-level analysis, we employ standard techniques (e.g., Jaccard similarity, NLL-test, BLEU) to analyze how different the generated texts are. In word-level utility, we mostly analyze the word (/phrase) co-occurrence or similarity, checking whether the generated EHRs have similar token distributions as the original corpora. We use three different technique to do so:

BLEU. We use Bilingual Evaluation Understudy (BLEU) [28] as our first evaluation metric for the word level utility. A text generation task could easily be considered as a machine translation task. Thus the use of the BLEU metric, which in machine translation task determines the closeness of the synthetic translation to one or more reference translations, applies to the synthetic text generation task. In our work, we used BLEU to compare the n-grams (words) of the synthetic documents to the original document and then to count the number of matches. Notably, these matches are position-independent, and more matches result in higher scores (maximum 1). For example, if there are 50% matches for bi-grams ($n = 2$) in both documents, then the BLEU-2 score will be 0.5. The higher the values of the BLEU score, the better the performance.

The centerpiece of the BLEU score is a *precision* measure. This precision measure is naively calculated by simply dividing the total number of n-grams ($n = 1, 2, \ldots, n$) that are present in the reference/original document by the total number of n-grams in the generated document. This naive approach, however, produces a high precision value wrongfully, when any n-grams present in the reference document is generated multiple times. Hence, a more sophisticated approach is adopted in the 'Modified n-gram Precision' measure, where the total count for the n-grams in the generated text is clipped by maximum count for the n-grams in the reference/original text.

Jaccard Similarity. Jaccard similarity can best be defined as the ratio of the intersection's size by the size of their union. For two text corpora with token vectors O and D (original and de-identified , respectively), Jaccard similarity can be calculated as [29],

$$JaccardSimilarity(O, D) = \frac{O \cap D}{O \cup D}$$

G^2-Test. G^2 is a frequentist approach for measuring corpus similarity utilizing the standard log-likelihood ratio test [30]. This statistical test represents whether two text corpus is the same in terms of a particular token's usage based on its frequency. Here, we consider each token's (or n-gram) frequency in both corpora and use the corpus size to output the log-likelihood value. Let the frequency for a specific token t_i in the original and synthetic dataset be a and b, respectively. Then, we retrieve the G^2 value for that particular t_i by following the

equations below [31]:

$$E1 = c * (a + b)/(c + d),$$
$$E2 = d * (a + b)/(c + d),$$
$$G^2 = 2(a \ln(a/E1) + b \ln(b/E2))$$

Here, $E1, E2$ are the expected frequencies of a token in two corpora, which is calculated from a, b, c, and d, where c, d represents the total words in both corpora, respectively. Effectively, the G^2 value provides a statistical significance [32] of a chosen token's presence or frequency in multiple corpora. For example, if the G^2 value is greater than 3.84 (critical value), then the difference between the two corpora (for that particular token) happening by chance is less than 5% (or $p < 0.05$). In other words, higher values of G^2 denote a statistical significance in the difference between each corpus; therefore, lower scores are preferred.

$$\text{EffectSize} = G^2/(c + d) \ln min(E1, E2) \tag{4}$$

On the other hand, lower G^2 values for every token represent the ambiguity whether two datasets are similar or not. The *effect size* of the tokens is also equally important. In equation 4, the effect size is calculated using the G^2 values and it ranges between 0 and 1 (inclusive) [33]. This value for any token represents the deviation from observed (a, b) to expected frequencies $(E1, E2)$. Notably, G^2 or the effect size is independent of the corpus size, which is important as the corpora's size is not fixed.

3.2.2 Document-level Utility. As the purpose of a generated synthetic dataset is to mirror the original sensitive corpus, we expect them to be used in a realistic scenario as well [34]. Therefore, we analyze the utility of these synthetic documents using a classification task. Here, the classification task will determine whether the underlying document contains a certain disease or not.

As mentioned earlier in the generation mechanism (Section 3.1), we generated the documents according to the underlying disease or conditions. We used the ICD9 specified codes, a standard code-book for each different medical condition as provided from the MIMIC-III dataset [19]. The dataset contained human annotations of these ICD9 codes for each document representing each visit from the patient. For example, if the patient had Type II Diabetes and Hypertension, then 250.00 and 401.9 codes were flagged as accurate as they represent the diseases, respectively.

Hence, in our document-level utility analysis, we will build different classifiers trained on original and synthetic dataset *separately*. Afterward, they will be tested on different test-sets from both corpora. For example, we will train a model with synthetic data and test it on a separate original (along with synthetic) test-set to analyze the utility of the generated data. The outcome of such a test will dictate how well the synthetic corpus will perform in real-life use-cases if we intend to publish them. Similarly, the original data trained model will be tested on synthetic datasets to test their efficacy. We selected seven diseases from the MIMIC-III dataset and built a multi-label classifier on it. The architecture of the classifier model is detailed in Section 3.1.4

3.2.3 Corpus-level Utility. We analyze the generated EHRs on corpus-level containing all EHR documents posing it as an adversarial classification problem [35]. In this task, we create a dataset having documents from the original and generated EHRs randomly. Then, we label each document as 0 or 1, reflecting whether they are original or synthetic. Then, we train multiple classifiers on these documents predicting whether they belong to the original or synthetic pool.

The purpose of such a classifier is to distinguish between original and synthetic records. This is termed as Adversarial Classification (AC), where a machine learning approach detects the source of the data based on its characteristics [36]. This is closely analogous to the Turing test, where a human evaluator examines the data and predicts whether it is real or fake. However, we did not opt for medical professionals or utilize any human annotations for the underlying medical texts. Hence, we replaced the human effort with an ML-based solution where a classifier differentiates between the original and synthetic data.

For brevity and better performance, we kept the classification model similar to the architecture discussed earlier in Document-level utility (in Section 3.1.4). Initially, a fixed-size sequence of tokens was taken from arbitrary documents (original/synthetic) and placed as inputs to the embedding layer. The final prediction is obtained from the output layer where we have a fully connected (FC) layer. Here, relative to the earlier approach, the only difference is the final FC layer will have binary or trinary classes, whereas it was the number of diseases for document classification. Notably, these are entirely different classifiers trained separately for both utility analyses.

For AC, we created a dataset with an equal number of EHR documents from the original and generated corpus, and we train four different models. For example, the dataset for AC contained 50% original and synthetic documents, and later, these were randomly split into 80:20 ratio for training and testing. The 20% of testing data had an equal amount of original and synthetic records. The following five settings (four models) were tested:

- **AC1:** We train and test a neural network model with 50% original and synthetic EHRs, each labeled as 0 and 1, respectively. The best adversarial classifier will be able to separate the original from synthetic ones, achieving a 100% accuracy.
- **AC2:** We randomly assign 0 to half of the original dataset and 1 to the other half of the same original dataset. Then we train a different model that has never seen the synthetic data. However, an ideal AC should be able to point the original records correctly and resulting in 50% accuracy.
- **AC3:** In this case, we repeat the procedure with the *generated dataset* and label half of them to 0 and 1 vice versa, randomly. Similarly, the highest accuracy here should be 50% since we did not consider any of the original data.
- **AC4:** In the fourth adversarial classifier, we change the binary classification to a 3-class problem where we consider two input sequences where they are randomly taken from the original and/or the synthetic corpus. Here, if both the sequences are taken from the original dataset, then we label them as 0, whereas two sequences from the synthetic corpus are labeled as 2. If one of the sequences is original, whereas the other collected from synthetic documents randomly, the label is set as 1. Thematically, this classifier will show whether it can identify the source of the EHR texts.
- **AdversarialSuccess (AdvSuc):** We also report the percentage of generated documents that were successful in deceiving the classifier model (*AC1*) and were tagged as original. Here, we send all documents from the synthetic dataset through the AC1 classifier and report the percentage of the documents that were mis-classified as the original. Therefore, the highest value will be 100%, which is desired from the ideal data generator which can fool the adversarial classifier.

The documents which are labeled as 1 in AC2 and AC3 should originally be 0; however we deliberately mislabeled them to understand the generalizability of these trained adversarial models. We did not consider the incurring losses from these AC models and merge them to the generative networks as this would change the generation technique by biasing towards fooling the ACs rather than producing a generalized texts. Thus, none of the utility metrics mentioned here are integrated with the generation network to improve utility performance.

4 RESULTS

4.1 Experimental Setup

4.1.1 EHR corpus. We used two datasets that contained PII tokens along with the unstructured medical texts from different patients: i2b2 2014 [18] and MIMIC-III [19]. The i2b2 dataset was constructed in 2014 as a de-identification benchmark dataset as it categorized the EHRs into the train, validation, and test sets and contained human-annotated PII tokens. Specifically, the i2b2 dataset had the sensitive PII tokens marked. On the contrary, the MIMIC-III is a large corpus that is available in a de-identified format (without PII), and the PII tokens are not available. These tokens were captured using computational methods [37], which the data publishers have followed the HIPAA standards [19].

We split the i2b2 dataset into two different corpora: a) `i2b2-original`, and b) `i2b2-deid`. The first corpus contained all PII tokens as the generated synthetic dataset based on this corpus should also contain similar PII information. As the original i2b2 dataset (resembling real-life medical text) cannot be published publicly, we consider this version of the dataset for utility analysis and comparison. In i2b2-deid, we remove all PII tokens (de-identification) according to the dataset's human-annotated labels as they are considered sensitive information. This version is safe to disseminate, and the generated text from this corpus should fall under the same privacy guarantee as well. Thus, the utility difference from the synthetic texts from i2b2-original and i2b2-died, respectively, will demonstrate the utility loss of de-identification . Notably, in these de-identification procedures, some of the regular tokens and PIIs are also sanitized, which is why the utility comparison is essential. We show that the utility from the generated original and de-identified datasets are similar while the de-identified datasets overcome the privacy constraints.

The i2b2 dataset, however, was much smaller than the MIMIC-III as we looked at 9,817 MIMIC-III discharge summaries (from 2,083,180) whereas i2b2 only had 521 documents. However, the MIMIC-III dataset did not provide the PII labels as it was published originally in a de-identified format. Thus, we only consider one corpus of MIMIC-III. Furthermore, The MIMIC-III dataset contained ICD-9 disease code annotations as we selected seven diseases according to their frequency. These disease classes are hypertension, congestive heart failure, atrial fibrillation, kidney failure, type II diabetes, respiratory failure, and urinary tract infection. These diseases were used as class labels for a supervised task employed in our document-level utility analysis, as described in Section 3.2.2.

All utility experiments considered the synthetic text data from i2b2-original, i2b2-deid, and MIMIC-III datasets. For the classifiers, each dataset was split 80:20 randomly, where 20% was set as a test set, which was never utilized during the training of the classifiers. We demonstrate one of the synthetic text data in Figure 4 generated from the MIMIC-III dataset. As expected, the unique identifiers are also kept de-identified in congruence with the original dataset as MIMIC-III did not contain the PII tokens. Interestingly, our generator did preserve the gender of the patient in the generated texts, but the date of birth, age, and admission date do not add up.

We also generated two separate differentially private corpora from i2b2 and MIMIC-III with ϵ of 23 and 8, respectively. These values were selected as the generated tokens, which were too noisy (or random) for lower epsilon values (more private). The i2b2 dataset had the PII tokens, whereas the MIMIC-III was de-identified . We choose higher ϵ for the i2b2 dataset as it had only 514 records compared to the 9k records of MIMIC-III. Therefore, according to privacy budget analysis outlined by [38], `i2b2-DP` generation required more privacy budget for smaller dataset.

4.1.2 Neural Network Hyper-parameters. Since we utilized different classifiers for the three utility analyses, we kept the neural networks' hyperparameters the same for all settings. For example, the number of iterations was defined by the convergence of the training loss as if the loss was not updated in the last five steps, then the training phase was concluded. We also set the dropout parameter in the final FC layer to be as 50% to avoid overfitting.

Furthermore, all training and testing ratios were set as 80:20; given the 20%, of the test set were never revealed to the model while training. However, due to the GPU memory constraints and the inherited classifier design from BERT [21], we confined the models from inputs more than 512 tokens on both documents and adversarial classifiers. However, we did experiment with lower sequence length (i.e., 256, 400), which resulted in poor and inconclusive outputs.

GENERATED DISCHARGE SUMMARY
Admission Date: [*2180-8-14*] **Service:** MEDICINE
Date of Birth: [*2114-3-16*] **Sex:** M
History of Present Illness: The patient is a 41-year-old man with a past medical history significant for chronic renal insufficiency, known type 1 diabetes, a prior history of stroke, and a recent admission for fungemia secondary to a urinary tract infection, who came in with acute kidney injury and hypertension to the Hospital. He was transferred to the [**Hospital1 18**] on [**2108-6-2**] after he developed acute onset of renal failure with a creatinine of 1.6 on admission. He was transferred to the [**Hospital1 18**] after he developed worsening renal failure with a Cr of 3.0 on admission. On [**2108-6-3**] he was transferred to the [**Hospital1 18**] after developing hypotension with an SBP in the 60s.
Physical Exam: *Vitals:* BP:100/55 P:110 R: 18 O2: 98% on 100% non-rebreather *HEENT:* Sclera anicteric, MMM, oropharynx clear...

Fig. 4. Sample of a generated EHR text data from MIMIC-III dataset

Table 2. Jaccard Similarity and G2 test on the different synthetic datasets along with benchmark from SeqGAN [14]

Dataset	SeqGAN [14]		Our Method	
	Jaccard ↑	G^2-Test ↓	Jaccard ↑	G^2-Test ↓
i2b2-original	69.67	0.83	68.15	1.83
i2b2-deid	75.58	1.3	76.77	1.95
i2b2-DP	-	-	75.04	3.45
MIMIC-deid	82.39	3.81	84.4	2.81
MIMIC-DP	-	-	95.48	5.64

4.2 Word-level Utility Result

4.2.1 Jaccard Similarity. At first, we show the utility results in word-level as it is the most granular analysis on the text corpus operated on the token level. Firstly, we use the Jaccard similarity test (equation 3.2.1), which tests how many tokens from the generated corpus match the original one. In other words, we consider the tokens from both corpora as two sets and retrieve the ratio of the intersection over the union (maximum 100%).

From Table 2, the Jaccard test shows that our generative method offers higher Jaccard similarity on i2b2-deid in comparison to the original i2b2. It is reasonably due to the sensitive PII tokens available in the original corpus, which were not generated accurately in the synthetic ones. For example, if the original corpus contained a patient with the name 'John' getting admitted in 31^{st} December, it is unlikely that the generator outputs the same name and date while producing the synthetic data.

In the generated de-identified corpus, there was no such issue as the name or dates were de-identified already. Furthermore, as the MIMIC dataset was already de-identified , it yielded a better similarity score. The differential private i2b2-DP and MIMIC-DP also showed impressive Jaccard scores as MIMIC-DP had 95.48% similarity. Notably, our Jaccard scores were competitive with the generated text from SeqGAN [14] as well. SeqGAN also did not have any privacy-preserving component, which is why we do not report the corresponding private results. Details on the SeqGAN's method are discussed in Section 6.

Table 3. BLEU-$\{1, 2, 3\}$ scores on i2b2 and MIMIC-III dataset comparing SeqGAN [14] with our approach

Dataset	SeqGAN [14]			Our Method		
	B-1	B-2	B-3	B-1	B-2	B-3
i2b2-original	0.94	0.19	0.05	0.94	0.66	0.53
i2b2-deid	0.72	0.09	0.02	0.89	0.79	0.74
MIMIC-deid	0.98	0.34	0.06	0.98	0.78	0.48
i2b2-DP	-	-	-	0.91	0.73	0.67
MIMIC-DP	-	-	-	0.99	0.88	0.73

4.2.2 G^2 Test. The G^2 statistical test was conducted under the null hypothesis that for an arbitrary token, there is no statistical significance between the token's association with the original or synthetic corpus. In other words, we cannot determine with a high probability that the specific token was taken either from the original or the generated corpus. For this experiment, we considered all tokens from the generated corpus and calculated the G^2 values and corresponding effect size (according to equation 4). In short, smaller G^2 values will denote the uncertainty in a token's membership which represents that the generated data is closer to the original one.

In Table 2, we show the average of the all token's G^2 values, which is less than the critical value 3.84. Therefore, we *cannot* reject the null hypothesis for our method in all three corpora (95 percentile). Furthermore, the MIMIC synthetic dataset had slightly higher G^2 value due to its corpus size, although the effect size of each token, in this case, was much smaller compared to i2b2.

Furthermore, analogous to Jaccard scores, our results are similar to the earlier approach [14] in all settings as well. However, contradictory to these two utility measures, the BLEU scores will show a significant difference between these methods.

The G^2 scores for i2b2-DP and MIMIC-DP were 3.45 and 5.64, respectively. However, i2b2-DP's score was less than the critical value for 95 percentile ($p<0.05$), MIMIC-DP's score was not. Therefore, there is a significance between the word frequencies of the original MIMIC-III and MIMIC-DP. However, this difference is smaller than the critical value for the 99 percentile ($p<0.01$), which is 6.63. We did not experiment with higher ϵ values that might further reduce the G^2 scores.

4.2.3 BLEU Score. In Table 3, the BLEU scores ($B - n$) of our proposed method is presented. We considered the $n = \{1, 2, 3\}$-gram values for the BLEU metric as we outperformed the existing SeqGAN [14] for i2b2 dataset on all three n values. Furthermore, as $n = 3$ considers three tokens appearing together, whereas $n = 1$ only takes the occurrence of a token, it is easier to achieve higher B-1 scores compared to B-3. Our experimental results are consistent with this finding, as all B-3 scores are smaller than the B-1s in all settings or corpus.

For example, the B-1 scores of SeqGAN and our approaches are head to head, whereas the difference in the B-2 and B-3 scores is clear. It denotes that SeqGAN [14] can generate single tokens from the input corpus but cannot output the consecutive bi-grams or tri-grams (phrases) correctly. As a result, the B-2/3 scores are significantly low. However, we see promising BLEU scores for $n = 2, 3$ in our transformer-based generative method. In summary, these BLEU scores show the utility difference of our approach compared to earlier work in a different dataset.

4.3 Document-level Utility Result

The document-level utility was measured using the ICD9 codes available in the MIMIC-III dataset. Since we generated the synthetic medical notes pre-conditioned of the seven different diseases, we employ a classifier to check whether the trained models perform similarly on each dataset. For example, we will train a model on the original or synthetic dataset and test on the synthetic or original corpus to test the interoperability of the

Table 4. Disease classification accuracy on different MIMIC-III dataset and varying number of diseases

Test \ Train	Original			Synthetic/DP			Mix		
# of Disease	3	5	7	3	5	7	3	5	7
Original	*97.2*	83.6	82.4	84.5	75.9	*69.9*	77.6	74.7	72.8
Synthetic	80.4	75.4	68.1	98.1	91.2	*82.3*	89.8	82.1	74.9
DP	91.2	89.5	86.6	98.5	96.1	95.6	89.1	84.1	78.5
Mix	70.8	65.4	61.9	70.1	65.9	61.3	88.1	71.9	70.7

classifiers. Here, we selected (randomly) 7k documents where each disease had 1k documents and was split into 80:20 train, test-set.

We also vary the number of diseases among $\{3, 5, 7\}$ to check the scalability of the classifier's accuracy in terms of class labels. Furthermore, we also created a mixed dataset, containing 50% original, and 50% generated medical texts and repeated the procedure. The train and test ratio was set as 80:20, as mentioned earlier in Section 4.1, and a sequence length of 512 was considered due to our GPU's memory limitation. It is noteworthy that the i2b2 corpus did not present the ICD-9 codes. Thus we only opted for MIMIC-III dataset for this utility analysis.

In Table 4, we show the results as the columns represent the training dataset, whereas the rows have the testing ones. The Mix test dataset was completely different than the Original or Synthetic ones and was randomly chosen. We can observe that the accuracy actually decreases with the number of diseases for all cases. For example, for three disease classification, a model trained and tested on Original MIMIC-III corpus results in 97.2% accuracy, whereas it decreases to 82.4% for seven diseases.

On the other hand, the model trained on one dataset under-performs while being tested on the other one. Here, a model trained on the synthetic dataset gained 69.9% accuracy for all seven disease classes, whereas it achieved 82.3% on its own test set. However, the drop in accuracy is not significant, which underscores the strength of the generation approach. For a lower number of classes (three diseases), the performance drop is minor.

From Table 4, we can argue that the Mix dataset training favors the Synthetic test-set as it outperforms the original one. Also, it holds better accuracy for the Mix test-set as well. The generated data from the Differentially Private (DP) generator performed better in analogous test sets, whereas the accuracy dropped in Original and Mixed test sets. However, these privately generated sets were comparable with their non-private counterparts.

In summary, we see a general trend of higher accuracy for the same dataset used for training and testing. However, the experiment's motivation was to check how the classifiers perform on other datasets that it has a limited idea about (knowledge gained while generation). It seems that the accuracy results on each document are comparable, and our synthetic corpus can be utilized in place of real-world datasets to build such classifiers.

4.4 Corpus-level Utility Result

In Table 5, we show the corpus-level utility of the synthetic data employing the adversarial classifiers (ACs). There four different ACs, as described in Section 3.2.3, each emphasizing different properties of the data. In summary, we want to analyze whether an ML classifier can identify the synthetic records from the original ones.

As the maximum accuracy from the ideal AC1 should be 100% denoting that it can correctly distinguish the fake ones, the AC1 on our synthetic corpus resulted in less than 60% accuracy for all datasets. Also, AC2 and AC3 are lopsided towards a specific corpus (original or synthetic) and only get information from that corpus while training; hence, they can only achieve a maximum of 50%. Nevertheless, AC2 and AC3 accuracy values are around 50% for all cases, which is interesting. For example, it leads to the fact that AC2 or AC3 for MIMIC-III

Table 5. Corpus-level utility test using adversarial classifier on i2b2 and MIMIC-III dataset

Dataset	Type	AdvSuc	AC1	AC2	AC3	AC4	AvgAC
i2b2	with PII	83.7	37.3	44.1	30.4	51.3	40.8
	deid	86.9	43.2	42.1	51.9	45.8	45.7
MIMIC	deid	92.3	54.6	51.6	52.4	52.5	52.8
MIMIC-DP	deid	93.4	52.3	52	50.3	51	51.4

dataset can identify original or synthetic records while only trained with one dataset. However, if presented with both data (AC1), it performs poorly.

Nevertheless, these AC1, 2, 3 are binary classifiers resulting in 50% accuracy, which resembles the coin toss probability over a large sample size (1000 documents). Therefore, we test AC4 with 3 class outputs, which is more complicated than these classifiers (AC1-3) where the accuracy values should reflect the ability of the AC to understand the relation between original and synthetic data. Notably, the worst-case accuracy, in this case, is 33.33%; albeit, an ideal AC4 will have 100% accuracy as it can correctly identify whether the two sequences as inputs are incoming from the same corpus or the opposite. Also, if they are sourced from the same dataset, then it should be able to forecast which one was it.

We see that the AC4 values are above the minimum 33.3% in all cases, signifying that the classifier can distinguish some documents. However, the values are not significant enough to conclude that the AC4 successfully separated the synthetic and original records. Nevertheless, AC4's performance was better than the other three classifiers.

The Adversarial Success (AdvSuc) were tested only on the AC1 as sequences from all synthetic records were tested. For example, in this synthetic corpus from MIMIC-III, we used 7k generated records (1k per disease for equality), which were tested, and 92.3% (6461 documents) were predicted as original. The differentially private synthetic sets also performed satisfactorily as the adversarial success was 93.4%.

5 DISCUSSION

5.1 Utility of the Synthetic Data

In Section 4, the utility of the novel synthetic data was analyzed from three viewpoints. Firstly, the three word-level analysis demonstrated how similar the generated corpus is compared to the original ones. Interestingly, among the three tests, Jaccard (Section 4.2.1) and G^2 test (Section 4.2.2) do not reveal the difference between our approach and preceding SeqGAN [14]. It seems the Jaccard indices are almost similar (in one case larger for i2b2-original), so are the G^2 values.

However, Jaccard and G^2 tests utilize single unique tokens from the whole corpus. For example, if the generator repeats some specific or more frequent words (i.e., fever, admission, date, etc.) *repeatedly*, it should achieve a higher similarity score. Therefore, these two tests are often inconclusive considering single tokens. Nevertheless, the BLEU score for $n = 2, 3$, where two or three consecutive tokens are considered, demonstrates the difference with SeqGAN as our scores are significantly higher.

For disease (Section 4.3) and adversarial classifier (Section 4.4), the results show the efficacy of the synthetic data. As shown in Table 4, the classification accuracy drops for seven diseases compared to five. We expected that the model trained on mixed data would attain the highest for both original and synthetic corpus, albeit which was not noticeable. Interestingly, the synthetic data model performs satisfactorily in the original corpus and degrades on the Mixed test set. A similar trend is also present in the model trained with the original data showing the interoperability of the data. We investigated this as a case of overfitting and rigorously tested with unknown random test datasets, standard regularizers, and a high dropout of 50%.

The interesting observation from the adversarial classifier in Section 4.4 is the percentage of adversarial success. For example, the synthetic data from MIMIC-III showed an impressive 92.3% success, which is higher than the i2b2 dataset. This is due to the size of the input corpus as i2b2 had only 241 documents, whereas it was around 10k for MIMIC-III. Therefore, larger corpus produced much generalized synthetic data, which deceived the AC attaining a better overall performance.

5.2 Privacy of the Synthetic Data

In this work, we did not use any automated PII de-identifiers [11, 15]. We chose to avoid such tools partially due to their accuracy and the ultimate privacy guarantee of the synthetic data. As the accuracy of these de-identification techniques does not adequately identify all PIIs from long medical texts, we cannot guarantee that the generated texts from such de-identification will not contain any private information. Inherently, the synthetic data from such de-identified corpus will not be safe to publish.

We also utilized a differentially private data generator that can take arbitrary text input and produce medical texts with an ϵ privacy guarantee. Therefore, this technique can be utilized on private raw datasets without any de-identification . However, the output corpus may still contain identifying information about a certain group or individuals as the DP generation does not guarantee to completely remove all identifying information or prevent potential privacy or linkage attacks on the synthetic dataset.

An optimal solution would be to de-identification the synthetic corpus, using the state-of-the-art automated tool. However, to analyze the accuracy of such de-identification on the synthetic dataset will require the PII labels for each token. Such annotation is only possible by human efforts as expert annotators can separate the PII tokens (in the synthetic dataset), which can be sanitized before publishing. However, in the future, with an acceptable de-identification tool, we can sanitize the synthetic data generated from the original data, which should provide the same privacy as guaranteed by the de-identifier tool.

In contrast, we use the PII annotations retrieved from the datasets and sanitize accordingly (for non-DP generation). Notably, this mechanism is safe as both datasets as i2b2 and MIMIC-III was published with the PII annotations or de-identified , respectively. Therefore, it is safe to assume that the generated data will hold the same privacy guarantee as to the de-identified data, which should be the maximum in our case. Furthermore, we can create an arbitrarily large number of records from a small subset, which is representative of that de-identified corpus.

5.3 Limitations

The utility of the texts is tested with different ML applications or standard NLP techniques. However, we could utilize medical professionals from the field to pan out similar experiments. For example, people working in health-care can decide whether the data is synthetic or real, which resembles our adversarial classifier. However, due to the human efforts required, we opted for this to be future work. Furthermore, this can be coupled with the de-identification task using human efforts as the PIIs can be identified alongside the source of the texts.

Moreover, even with different utility metrics, we agree that the usability of these texts is always use-case specific. For example, there is a possibility that for some other applications, these generated data falls short. Also, researchers usually prefer a dataset without any noise (resulting from a DP mechanism) or the noise to be minimal due to the inherent implications in utility. We did not experiment or explore this privacy-utility relation as it was exhaustive due to the several ML training procedures (i.e., generation, utility classifiers) that were necessary to get the complete picture.

De-identified inputs to the generator also levy a limitation as the generated data will only contain the statistical properties and traits of this input subset. Therefore, a small de-identified corpus as input might not represent the whole dataset. Only if the input corpus to the generator is sufficiently large or holistic, then it may be used in different tasks in place of the original private corpus.

6 RELATED WORKS

Generating unstructured texts is an active area of research in deep learning and natural language processing. Since multiple methods are proposed for such generation, we categorize our related works according to the corresponding machine learning technique. Notably, we only discuss the related works that can be generalized for generating long textual EHRs:

6.1 Variational Autoencoder

The Variational Autoencoder (VAE) [39, 40] is a generative model that is based on a regularized version of the standard autoencoder combined with variational inference. VAE has developed as one of the well-known methods for unsupervised learning of complicated distributions [41]. Therefore, VAE was applied for several text generations problems. Usually, recurrent neural network-based language model generates sentences one word at a time step, instead of generating a complete sentence at a time. To solve this problem, Bowman *et al.* [42] introduced an RNN-based VAE generative model that combines distributed latent representations of the complete sentences. This method used simple deterministic decoding, especially to generate distinct and well-formed sentences. However, they have been outperformed by Generative Adversarial Net in different image generation tasks, which is why we did not opt for VAE-based solutions in generating EHRs.

6.2 Generative Adversarial Net (GAN)

GAN was proposed in 2014 by Goodfellow *et al.* [43], which utilizes a generator-discriminator based model. The generative component of a GAN generates synthetic data, and the discriminator tries to predict whether the data is real or fake. The generator learns to generate better data by trying to fool the discriminator and conceptualize the data inherently. However, regardless of this simplicity while training, GANs often behave unpredictably and prove to be challenging to understand its behavior.

In 2017, Yahi *et al.* [44] proposed an unsupervised framework to produce consecutive time-series data of EHRs exploiting GAN technique. The framework predicts the consequence of drug exposure on laboratory test data. Similarly, Esteban *et al.* [45] proposed a Recurrent GAN (RGAN) and Recurrent Conditional GAN (RCGAN) to generate multi-dimensional real-valued time series, which can prove to be useful for medical applications. RGAN uses a recurrent neural network (RNN) in the discriminative and generative model as they are trained on auxiliary information in RCGAN.

In 2016, Guan *et al.* proposed an approach named Medical Text Generative Adversarial Network (mtGAN), which generates synthetic data EHRs [46] using GANs. It uses a summarized version of the EHRs to train its `reinforcement` algorithms based network. In this work, we utilize the full unstructured EHR texts, which is longer and contains more information than a summary.

However, all GAN models generate texts by sampling words sequentially, and sometimes the outcomes of these models are not realistic due to the lower quality of their sampled words. Recently, MaskGan is proposed by Fedus *et al.* [47] in 2018, improving the quality of the sample token which comprise the synthetic data. It proposed an actor-critic conditional GAN framework that teaches the generator to generate high-quality samples and showed success in image generation. Primarily, the model is trained using the Cloze task (similar to fill in the blanks) on a large text corpus. Notably, this is the first unconditional generative model using sequence-to-sequence learning. In 2018, Jordon *et al.* [38] introduced a state-of-the-art GAN framework named PATE-GAN which can be trained to generate `differentially private` [25] synthetic data. They revised the Private Aggregation of Teacher Ensembles (PATE) framework [48] and applied it to GANs.

6.3 Reinforcement learning (RL)

However, GAN techniques have a limitation where the outputs of the model create the obstacles for passing the gradient update from the discriminative model to the generative model. To solve this problem, some studies apply reinforcement learning (RL), where RL policy used as the method of the discriminator guiding generator.

Yu *et al.* introduced a new GAN method named sequence generation framework (SeqGAN) [14] in 2017. SeqGAN altered the generative model to avoid differentiation and directly go to the policy gradient using the RL technique. The reward-based on RL trains the discriminative model on a complete sequence and goes back applying a Monte Carlo search. The BLEU and G^2 test results on smaller sequences were promising as we employed this method to generate textual EHRs, and added other utility metrics (Section 4.3 and 4.4).

However, there are some shortcomings when the required length of the generated text is arbitrarily long. To address this problem, Guo *et al.* [49] proposed a new RL framework called LeakGAN. In their studies, they allow leaking its own high-level extracted features from the discriminative model to strongly guide the generative model. The framework has two modules, namely Manager and Worker, where the manager learns the extracted features of the current generated words, and the worker learns to fulfill the next one. However, it was difficult to train on the EHR dataset, and we opted for its predecessor, SeqGAN, for benchmarking.

7 CONCLUSION

In this paper, we propose a synthetic text generator that addresses the privacy concerns of sharing medical text data. To the best of our knowledge, this is the first paper that proposed a differentially private medical text generation technique and analyzed their applications and usability. The utility of such synthetic data is also compared with the original datasets and benchmarked against another technique [14] on three different settings. The experimental results demonstrate an almost indistinguishable performance between the original and synthetic data supporting the effectiveness of these generated free-texts. Furthermore, our synthetic dataset surpassed the state-of-the-art generation technique on every utility metrics. However, in this work, we did not consider (or propose) a de-identification technique for EHRs, which is also an important research area in medical texts. We argue that our proposed method can potentially allow public dissemination of texts as they are probabilistic. In the future, any state-of-the-art de-identification tool can be easily integrated into the pipeline, making the synthetic data de-identified as well.

AVAILABILITY OF MATERIALS

The code is publicly available in https://github.com/mominbuet/GenerateEHRs.git

COMPETING INTERESTS

The authors declare no competing interests.

REFERENCES

[1] Neil Mehta and Murthy V Devarakonda. Machine learning, natural language programming, and electronic health records: The next step in the artificial intelligence journey? *Journal of Allergy and Clinical Immunology*, 141(6):2019–2021, 2018.

[2] Yanshan Wang, Liwei Wang, Majid Rastegar-Mojarad, Sungrim Moon, Feichen Shen, Naveed Afzal, Sijia Liu, Yuqun Zeng, Saeed Mehrabi, Sunghwan Sohn, et al. *Journal of biomedical informatics*, 77:34–49, 2018.

[3] Robert H Miller and Ida Sim. Physicians' use of electronic medical records: barriers and solutions. *Health affairs*, 23(2):116–126, 2004.

[4] Josh Nisker. Pipeda: A constitutional analysis. *Can. B. Rev.*, 85:317, 2006.

[5] George J Annas et al. Hipaa regulations-a new era of medical-record privacy? *New England Journal of Medicine*, 348(15):1486–1490, 2003.

[6] Accountability Act. Health insurance portability and accountability act of 1996. *Public law*, 104:191, 1996.

[7] Md Momin Al Aziz, Md Nazmus Sadat, Dima Alhadidi, Shuang Wang, Xiaoqian Jiang, Cheryl L Brown, and Noman Mohammed. Privacy-preserving techniques of genomic data—a survey. *Briefings in bioinformatics*, 2017.

[8] M. Douglass, G. D. Clifford, et al. Computer-assisted de-identification of free text in the MIMIC II database. In *CinC*, 2004.

[9] MM Douglass, GD Cliffford, Andrew Reisner, WJ Long, GB Moody, and RG Mark. De-identification algorithm for free-text nursing notes. 2005.

[10] Ishna Neamatullah, Margaret M. Douglass, Li-wei H. Lehman, et al. Automated de-identification of free-text medical records. *BMC Medical Informatics and Decision Making*, 2008.

[11] Franck Dernoncourt, Ji Young Lee, Ozlem Uzuner, and Peter Szolovits. De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association*, 24(3):596–606, 12 2016.

[12] Vithya Yogarajan, Bernhard Pfahringer, and Michael Mayo. A review of automatic end-to-end de-identification: Is high accuracy the only metric? *Applied Artificial Intelligence*, 34(3):251–269, 2020.

[13] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[14] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[15] Simson L Garfinkel. De-identification of personal information. *National institute of standards and technology*, 2015.

[16] Kimberly J O'malley, Karon F Cook, Matt D Price, Kimberly Raiford Wildes, John F Hurdle, and Carol M Ashton. Measuring diagnoses: Icd code accuracy. *Health services research*, 40(5p2):1620–1639, 2005.

[17] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

[18] Amber Stubbs, Christopher Kotfila, and Özlem Uzuner. Automated systems for the de-identification of longitudinal clinical narratives: Overview of 2014 i2b2/uthealth shared task track 1. *In JBI*, 2015.

[19] Alistair E. W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. Mimic-iii, a freely accessible critical care database. *Nature Scientific Data*, 2016.

[20] Fei Song and W Bruce Croft. A general language model for information retrieval. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 316–321, 1999.

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

[22] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

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

[24] Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12, 2006.

[25] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[26] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[27] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[28] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[29] C. Plattel. *Distributed and Incremental Clustering using Shared Nearest Neighbours*. PhD thesis, Utrecht University, 2014.

[30] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[31] Paul Rayson and Roger Garside. Comparing corpora using frequency profiling. In *Proceedings of the workshop on Comparing corpora-Volume 9*, pages 1–6. Association for Computational Linguistics, 2000.

[32] Jefrey Lijffijt, Terttu Nevalainen, Tanja Säily, Panagiotis Papapetrou, Kai Puolamäki, and Heikki Mannila. Significance testing of word frequencies in corpora. *Literary and Linguistic Computing*, 31(2):374–397, 2016.

[33] Janis E Johnston, Kenneth J Berry, and Paul W Mielke Jr. Measures of effect size for chi-squared and likelihood-ratio goodness-of-fit tests. *Perceptual and motor skills*, 103(2):412–414, 2006.

[34] Homa Alemzadeh and Murthy Devarakonda. An nlp-based cognitive system for disease status identification in electronic health records. In *2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pages 89–92. IEEE, 2017.

[35] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, 2004.

[36] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.

[37] Ishna Neamatullah, Margaret M Douglass, H Lehman Li-wei, Andrew Reisner, Mauricio Villarroel, William J Long, Peter Szolovits, George B Moody, Roger G Mark, and Gari D Clifford. Automated de-identification of free-text medical records. *BMC medical informatics and decision making*, 8(1):32, 2008.

[38] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Pate-gan: generating synthetic data with differential privacy guarantees. 2018.

[39] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[40] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[41] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[42] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[43] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[44] Alexandre Yahi, Rami Vanguri, Noémie Elhadad, and Nicholas P Tatonetti. Generative adversarial networks for electronic health records: a framework for exploring and evaluating methods for predicting drug-induced laboratory test trajectories. *arXiv preprint arXiv:1712.00164*, 2017.

[45] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[46] Jiaqi Guan, Runzhe Li, Sheng Yu, and Xuegong Zhang. Generation of synthetic electronic medical record text. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 374–380. IEEE, 2018.

[47] William Fedus, Ian Goodfellow, and Andrew M Dai. Maskgan: better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*, 2018.

[48] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

[49] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[50] C. Goller and A. Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *ICNN*, 1996.

A ADDITIONAL PRELIMINARIES

A.1 Machine Learning Background

A.1.1 Word Embedding. Most machine learning tasks representing natural language will represent the words or tokens in a vector embedding. For example, a unique word will be represented by a fixed size numeric vector, which is handled in the embedding layer. This layer receives a sequence of tokens (sentences) as inputs from the data layer (Section 3.1.4). Then, it generates a random unique numeric encoding for each token. Previously, NLP methods encoded each token as a discrete number, which perceived no beneficial information about the dis/similarity among the tokens.

The weaknesses of these encoding are mostly addressed in the Vector Space Models (VSMs), where each token is represented in an endless vector space. In VSMs, semantically similar tokens are mapped to nearby locations in a fixed dimensional geometric space. In this mapping, a fixed-sized vector represents one word, as the method is called a word embedding. For example, if you subtract 'Man' from 'Brother' and add 'Woman,' it gives the embedding vector of 'Sister' (Figure 5). Here, these numeric word embeddings help answer questions like "How similar the tokens Brother and Sister is compared to Man and Women?". Nevertheless, there are two different embedding scheme used previously:

Fixed Embedding: In this embedding technique, each token will be represented by a unique numeric vector that is fixed or unchanged regardless of the usage or context. For example, in 'The doctor was right about ...' and 'the patient's right atrium ...', the word 'right' has different meanings. However, in the fixed embedding space, both words will have the same vector.

Dynamic Embedding: In the dynamic embedding, we take the context of the tokens into account, and they can have different vectors based on their context. For example, the word 'right' will have different embedding vector

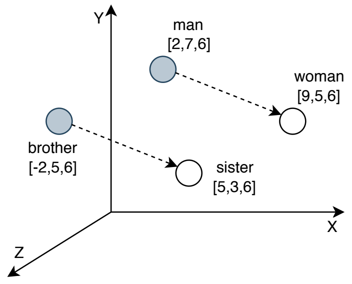


Fig. 5. Embedding space for four words where similar words have similar distances

in the dynamic embedding scheme as it means correctness and direction, respectively. self-attention [23] based mechanism to produce such dynamic embeddings which are described next.

A.1.2 Multi-headed Self-attention Model. The self-attention mechanism for language modeling and dynamic embedding calculation was first introduced by Vaswani *et al.* [23] as a replacement of the firmly established Recurrent Neural Network (RNN) [50] based models. The model introduced in [23] is named the Transformer model, which consists of an encoding component, a decoding component, and connections between them. Both the encoding and decoding component is a stack of encoders and decoders blocks, respectively. Each block in their respective stacks is identical to each other. However, the structure and functionality of an encoder block and decoder block are different from each other.

An encoder block has two sub-layers–(i) a Multi-headed Self-Attention sub-layer and (ii) a Feed-Forward Neural Network sub-layer. The inputs to the encoder flow through the self-attention layer first. This layer helps to look at every other word in a sentence when it encodes a specific word. Hence, the name self-attention. The output of the self-attention layer is fed to the feed-forward neural network. Each sub-layer has a residual connection around it and is always followed by a layer-normalization step.

The output of the final encoder bock this forwarded to the decoder component. As mentioned earlier, the decoder component is constructed with a decoder block. These decoder blocks also have two sub-layers, similar to the encoder block with one major distinction in the self-attention sub-layer. Whereas in the encoder blocks, the attention score of each input word is calculated with respect to every other input word, the decoder blocks only look at only to the words came before the current word for which the attention score is calculated. This slightly modified self-attention is called masked self-attention. As we have seen we used the only the encoder component in the classifier models and the decoder component in the generation model.