REVISITING WORD-LEVEL LDP ANALYSIS IN (LYU ET AL., 2020A)

This section aims at revisiting privacy protection in (Lyu et al., 2020a) and describes existing issues of privacy accumulation over the embedding dimension in terms of theory and experimental results of their approach, which is confirmed by the authors (Lyu et al.). Then, we provide corrected Theorems based upon our theoretical analysis, and we compare them with our approaches.

In the paper, they aim at preserving privacy of the extracted test representation from the user while maintaining a good performance of the classifier, which is trained at a server by the data collected from users. To achieve the goal, they consider a word-level DP; i.e., two inputs x and x' are considered to be adjacent if they differ by at most 1 word. Additionally, they introduce a DP noise layer r after a predefined feature extractor $f(x)$. To train a robust classifier at the server, they add the same level of noise as the test phase in the training process and optimize the classifier by minimizing the loss function as follows:

$$\mathcal{L}(x, y) = \mathcal{X}(C(f(x) + r), y) \tag{10}$$

where C is the classifier, y is the true label, $\mathcal{X}$ is the cross entropy loss function. The Laplace noise layer r is injected into the embedding $f(x)$ in which its coordinates $r = \{r_1, r_2, \ldots, r_k\}$ are i.i.d. random variables drawn from the Laplace distribution defined by $Lap(b)$ with $b = \frac{\Delta_f}{\epsilon}$, ϵ is the privacy budget, and Δ_f is the sensitivity of the extracted representation. Here, k is the dimension of the embedding $f(x)$.

Algorithm 1 describes how to derive differentially private representation from the feature extractor f. Note that x_s in the Algorithm 1 is a sentence (equivalent to x in our notation), which is considered to be sensitive and be protected.

Algorithm 1 Differentially Private Neural Representation (DPNR) (Lyu et al., 2020a)

1: **Input**: Each sensitive input $x_s \in \mathbb{R}^d$, feature extractor f
2: **Parameters**: Dropout vector $I_n \in \{0, 1\}^d$
3: Word dropout: $\tilde{x}_s \leftarrow x_s \odot I_n$, where $\odot$ performs a word-wise multiplication.
4: Extraction: $x_r \leftarrow f(\tilde{x}_s)$
5: Normalization: $x_r \leftarrow x_r - \min(x_r)/(\max(x_r) - \min(x_r))$
6: Perturbation: $\hat{x_r} \leftarrow x_r + r$, $r_i \sim Lap(b)$
7: **Output**: Perturbed representation $\hat{x_r}$.

Theorems 1 and 2 in (Lyu et al., 2020a) do NOT hold. Lyu et al. (2020a) consider adjacent databases differing by one word. It is clear that changing one word in x may result in changing the entire embedding vector $f(x)$. In their paper, they normalize each element of $f(x)$ into the range $[0, 1]$ (Line 5, Algorithm 1), hence each element sensitivity of $f(x)$ is $\Delta_f = 1$, the noise is $Lap(\Delta_f/\epsilon)$. Therefore, each element of the embedding $f(x)$ consumes a privacy budget ϵ. Since the k elements of the embedding are derived from a single sensitive input x, applying the LDP mechanism $\mathcal{A}(.)$, i.e., $Lap(b)$, k times will consume the privacy budget $k\epsilon$. This follows the composition property in DP, also known as the curse of dimensionality in local DP. Note that the k elements cannot be treated by using the parallel property in DP (Dwork and Lei, 2009), since all of them are derived from a single input x (data), NOT from k different inputs (k different datasets). Consequently, Theorem 1 of (Lyu et al., 2020a) does NOT hold at their reported ϵ word-level DP. Since Theorem 2 of (Lyu et al., 2020a) is derived from the result of Theorem 1 of (Lyu et al., 2020a), it is clear that Theorem 2 of (Lyu et al., 2020a) also does NOT hold.

Element-level DP does NOT hold. During our discussion with the authors of (Lyu et al., 2020a), the authors mentioned that their approach preserves a new notion of $(\epsilon, 0)$-element-level DP, i.e., two embeddings differ from one element, instead of a word-level DP. However, for the element-DP to hold, all the elements in the embedding $f(x)$ must be independent from each other, that is, changing one element will not result in changing any other element. If changing one element results in changing all the remaining elements, then element-DP will be suffered from the dimension of the embedding by following group privacy. In the current approach, changing one element means there is a change in the input data x to occur. Equivalently, using BERT, any change in the input data x will result in changing the whole

embedding (all elements). Therefore, the condition of two neighboring embeddings only differing in only one element does NOT hold in theory and practice. Consequently, the introduced element-level DP does NOT hold at the level of $(\epsilon, 0)$-DP.

Surprisingly Good Experimental Results in (Lyu et al., 2020a) due to Inappropriate Privacy Analysis. In their experimental results, e.g., Table 2 of (Lyu et al., 2020a), it is surprising that the approach could achieve almost the same (and even better) model utility with noiseless model given the extremely low $\epsilon = 0.05$ using BERT embeddings. For Theorems 1 and 2 to hold and support the correctness of the approach, the privacy budget must be $\epsilon \times k$, which is at least $0.05 \times 768 = 38.4$, instead of just ϵ reported in the paper. Similar results were reported through out the all in experiments. Eventually, the approach in (Lyu et al., 2020a) cannot provide any practical DP protection to the embedding at any levels, including word-level, character-level, and even a single embedding element given an input data x.

Our Correcting Theorems 1 and 2 in (Lyu et al., 2020a). Based upon our analysis, we introduce corrected versions of the Theorems 1 and 2 in (Lyu et al., 2020a), as follows.

Theorem 1. ***Corrected Theorem 1 in (Lyu et al., 2020a).*** *Let the entries of the noise vector r be drawn from $Lap(b)$ with $b = \frac{\Delta_f}{\epsilon}$. The Algorithm 1 is $k\epsilon$-word-level DP, where k is dimension of the embedding $f(x)$.*

Proof. Each element of the embedding f is bounded in $[0, 1]$, so $\Delta_f = 1$ for each element. By adding i.i.d. random noise variables drawn from the Laplace $Lap(b)$ with $b = \frac{\Delta_f}{\epsilon}$ into each element of f, each element consumes ϵ/k privacy budget. Since the k elements of the embedding are derived from a single sensitive input x, applying the mechanism $Lap(b)$ k times on the k elements will consume the privacy budget $k\epsilon$. Therefore, the Algorithm 1 is $k\epsilon$-word-level DP. □

Theorem 2. *Given an input $x \in D$, suppose $\mathcal{A}(x) = f(x) + r$ is $k\epsilon$-word-level DP, let I_n with dropout rate μ be applied to x: $\tilde{x} = x \odot I_n$, then $\mathcal{A}(\tilde{x})$ is ϵ'-word level-DP, where $\epsilon' = \ln[(1-\mu)\exp(k\epsilon) + \mu]$.*

Proof. Suppose there are two adjacent inputs x_1 and x_2 that differ only in the i-th coordinate (word), say $x_{1i} = v$, $x_{2i} \neq v$. For arbitrary binary vector I_n, after dropout, $\tilde{x}_1 = x_1 \odot I_n$, $\tilde{x}_2 = x_2 \odot I_n$, there are two possible cases, i.e., $I_{ni} = 0$ and $I_{ni} = 1$.

If $I_{ni} = 0$: Since x_1 and x_2 differ only in i-th coordinate, after dropout $\tilde{x}_{1i} = \tilde{x}_{2i} = 0$, hence $x_1 \odot I_n = x_2 \odot I_n$. Then $Pr\{\mathcal{A}(x_1 \odot I_n) = S\} = Pr\{\mathcal{A}(x_2 \odot I_n) = S\}$.

If $I_{ni} = 1$: Since x_1 and x_2 differ only in i-th coordinate, after dropout $\tilde{x}_{1i} = v$, and $\tilde{x}_{2i} \neq v$. Since $\mathcal{A}(x)$ is $k\epsilon$-word level-DP, then $Pr\{\mathcal{A}(x_1 \odot I_n) = S\} \leq \exp(k\epsilon) Pr\{\mathcal{A}(x_2 \odot I_n) = S\}$.

Combining these two cases, and $Pr[I_{ni} = 0] = \mu$, we have:

$$\begin{aligned} Pr\{\mathcal{A}(x_1 \odot I_n) = S\} &= \mu Pr\{\mathcal{A}(x_1 \odot I_n) = S\} + (1-\mu) Pr\{\mathcal{A}(x_1 \odot I_n) = S\} \\ &\leq \mu Pr\{\mathcal{A}(x_2 \odot I_n) = S\} + (1-\mu)\exp(k\epsilon) Pr\{\mathcal{A}(x_2 \odot I_n) = S\} \\ &= [(1-\mu)\exp(k\epsilon) + \mu] Pr\{\mathcal{A}(x_2 \odot I_n) = S\} \\ &= \exp\Big(\ln[(1-\mu)\exp(k\epsilon) + \mu]\Big) Pr\{\mathcal{A}(x_2 \odot I_n) = S\} \end{aligned} \tag{11}$$

Therefore, after dropout, the privacy budget is $\epsilon' = \ln[(1-\mu)\exp(k\epsilon) + \mu]$. □

Comparison with Our Work. Apart from the privacy accumulation over the embedding dimension issue, in their work, during training the model, they draw the Laplace or Gaussian noise at every training iteration. It means that the model accesses the raw data at every iteration; therefore, the privacy budget at the training phase is accumulated over the number of training iterations, which can be a large number that results in an exploded privacy budget in training. In fact, they focus on protecting privacy at the inference time and use the noise in the training phase to obtain a more robust model without considering training data privacy. This is different from our goal to protect users and sensitive entities of training data, which is a more challenging task. Our UeDP-preserving model can be deployed to the end-users for a direct use in the inference phase, without demanding that the end-users send their data embedding to our server; thus offering a more rigorous privacy protection and better usability. In addition to this, our approach

offer a tight DP budget bound compared with the DPNR algorithm in (Lyu et al., 2020a), which consumes large DP budgets that is proportional to the large size of the embedding k.

REVISITING BINARY ENCODING-BASED LDP ANALYSIS IN (LYU ET AL., 2020B)

This section aims at revisiting privacy protection for the binary encoding-based LDP mechanism in (Lyu et al., 2020a) and it describes the existing privacy exaggeration problem and the loose privacy budget bounding problem in terms of theoretical results of their approach.

In general, let us apply a randomized response mechanism $B(v)$ on a binary bit string v to preserve LDP. We denote $B_i(v)$ when $B(v)$ is applied on bit i-th of v. If $B(v)$ is ϵ-LDP applied on all elements of the embedding v, then the privacy budget ϵ consumed by v is computed from:

$$\frac{P_{B(v)}(z)}{P_{B(v')}(z)} = \frac{\exp(-\frac{\epsilon d(B(v)-z)}{\Delta B})}{\exp(-\frac{\epsilon d(B(v')-z)}{\Delta B})} \leq \exp(\frac{\epsilon |d(B(v)-z) - d(B(v')-z)|}{\Delta B}) \leq \exp(\frac{\epsilon d(B(v)-B(v'))}{\Delta B}) \quad (12)$$

where $\Delta B = \max_{v,v'} |B(v) - B(v')|$ is the sensitivity of the function $B(v)$.

Hash Function-based LDP (Erlingsson et al., 2014; Bassily and Smith, 2015; Wang et al., 2017) If v is obtained by mapping a value or an input data into a random hash using a hash function, then all elements of v are i.i.d., and changing one bit of v will not result in changing any other bits of v. Therefore, $\Delta B = \sum_i \Delta B_i$, in which ΔB_i is the sensitivity of $B_i(v)$ computed by $\Delta B_i = \max_{v,v'} |B_i(v) - B_i(v')| = \max_{v,v'} \sum_{j=1}^{rl} |B_i(v_j) - B_i(v'_j)| = 1$, and $d(B_i(v) - B_i(v')) = 1$ since v_i and v'_i are adjacent databases.

Therefore, in hash function-based approaches (Erlingsson et al., 2014; Bassily and Smith, 2015; Wang et al., 2017), we have:

$$\frac{P_{B(v)}(z)}{P_{B(v')}(z)} = \Pi_{i=1}^{rl} \frac{P_{B_i(v)}(z)}{P_{B_i(v')}(z)} \leq \Pi_{i=1}^{rl} \exp(\frac{\epsilon_i d(B_i(v) - B_i(v'))}{\Delta B_i}) \leq \Pi_{i=1}^{rl} \exp(\epsilon_i) = \exp(\sum_i \epsilon_i) \quad (13)$$

As a result, the privacy budget spent on v is truthfully ϵ where $\epsilon = \sum_i \epsilon_i$.

Binary Encoding-based LDP (Lyu et al., 2020b) As discussed in Section 2, binary encodings of all the embedded features are concatenated into a large binary vector v of the size rl, where r is the number of embedded features, and l is the number of binary bits, i.e., 1 sign bit, m integer bits, and n fraction bits, used to encode each embedded feature's real value, i.e., $l = 10$ in (Lyu et al., 2020b). Straightforwardly, Lyu et al. (2020b) applied the randomized response mechanism B on the bit string rl and shown that they can achieve ϵ-LDP:

$$\frac{P_{B(v)}(z)}{P_{B(v')}(z)} = \Pi_{i=1}^{rl} \frac{P_{B_i(v)}(z)}{P_{B_i(v')}(z)} \leq \exp(\epsilon) \quad (14)$$

However, a critical mistake in their paper is considering $\Delta B = rl$ for the entire embedding v of size rl, which is equivalent to consider $\Delta B_i = 1$ for every bit. In addition, a binary encoding bit string cannot be treated as a random hash, since the released result of $B_i(v)$ can be transformed into a real-value exposing privacy risk to the true value v. This is fundamentally different for a hash, in which the perturbation cannot be transformed into a real-value. In other words, the bits in binary encoding is not independent as binary bits in a hash.

From Eq. 14, for every bit i, there always exists a privacy budget ϵ_i quantified by $\frac{P_{B_i(v)}(z)}{P_{B_i(v')}(z)}$. Note that $\epsilon = \sum_i \epsilon_i$. This is equivalent to satisfying the following condition: $\frac{P_{B_i(v)}(z)}{P_{B_i(v')}(z)} \leq \exp(\frac{\epsilon_i d(B_i(v) - B_i(v'))}{\Delta B_i}) \leq \exp(\epsilon_i)$. Next, we will show that this condition does NOT always hold given the binary encoded bit string rl.

Given $\Delta B_i = 1$ for every bit, then we have:

$$\frac{P_{B_i(v)}(z)}{P_{B_i(v')}(z)} \leq \exp(\frac{\epsilon_i d(B_i(v) - B_i(v'))}{\Delta B_i}) = \exp(\epsilon_i d(B_i(v) - B_i(v')))$$

To bound the privacy loss given the $B_i(v)$, we need to bound the distance function $d(B_i(v) - B_i(v'))$.

Privacy Risk Exaggeration. Given a bit i in the integer part of one embedding element, we have that:

$$\text{If } i \in [1, m] \text{ is an integer bit : } d(B_i(v) - B_i(v')) \leq \max_{v,v'}(|B_i(v) - B_i(v')| \times 2^{i-1}) \tag{15}$$

We can see that $|B_i(v) - B_i(v')| \leq 1$. As a result, we have

$$\text{If } i \in [1, m] \text{ is an integer bit : } d(B_i(v) - B_i(v')) \leq 2^{i-1} \tag{16}$$

Consequently, we have that

$$\text{If } i \in [1, m] \text{ is an integer bit : } \frac{P_{B_i(v)}(z)}{P_{B_i(v')}(z)} \leq \exp(\frac{\epsilon_i d(B_i(v) - B_i(v'))}{\Delta B_i}) \leq \exp(2^{i-1}\epsilon_i) \tag{17}$$

From the Eq. 18, the actual privacy budget used for an integer bit $i \in [1, m]$ is $2^{i-1}\epsilon_i$, which is significantly larger than the privacy budget ϵ_i quantified by the binary encoding-based LDP approach in (Lyu et al., 2020b). We call this a privacy risk exaggeration in the integer part of the binary encoding-based LDP approach in (Lyu et al., 2020b). This problem is also true for the sign bit. If i is a sign bit, then the actual privacy budget is $2^{m+1}\epsilon_i$, since $d(B_i(v) - B_i(v')) \leq 2^{m+1}$, as follows:

$$\text{If } i \text{ is a sign bit : } \frac{P_{B_i(v)}(z)}{P_{B_i(v')}(z)} \leq \exp(\frac{\epsilon_i d(B_i(v) - B_i(v'))}{\Delta B_i}) \leq \exp(2^{m+1}\epsilon_i) \tag{18}$$

This privacy risk exaggeration problem is severe, since the actual privacy budget can be exponential, given $m \times r$ integer bits and r sign bits in the whole bit string rl.

Loose Privacy Budget Bounding. Similarly, for fraction bits $i \in [1, n]$, we have that

$$\text{If } i \in [1, n] \text{ is a fraction bit : } \frac{P_{B_i(v)}(z)}{P_{B_i(v')}(z)} \leq \exp(\frac{\epsilon_i d(B_i(v) - B_i(v'))}{\Delta B_i}) \leq \exp(2^{-i}\epsilon_i) \tag{19}$$

From Eq. 19, the actual privacy budget used for a fraction bit $i \in [1, n]$ is $2^{-i}\epsilon_i$, which is smaller than the privacy budget ϵ_i quantified in the binary encoding-based LDP approach in (Lyu et al., 2020b). In other words, the approach proposed in (Lyu et al., 2020b) quantified the privacy budget more than it needed. We call this a loose privacy budget bounding problem in the fraction part in (Lyu et al., 2020b).

To conclude, straightforwardly applying a randomized response mechanism on binary encoded vector as in (Lyu et al., 2020b) cannot correctly bound the actual privacy risk with local DP, due to the primitive difference between a random hash and a binary encoding bit string. The privacy risk exaggeration problem can severely loosen the privacy protection claimed in (Lyu et al., 2020b). Similar approaches with (Lyu et al., 2020b), such as (Chamikara et al., 2019), may suffer from the same problems.