Study of the article
Differentially Private Representation for NLP : Formal Guarantee and An Empirical Study on Privacy and Fairness

Samy Zouhri & Rony Abecidan

January 13, 2021

Introduction

Nowadays, numerous NLP algorithms are used for many purposes individual to each user of a service. For instance, we can translate a text using an online translator, obtain propositions for the end of the sentences we are writing, detect suspicious emails or exchange with chatbots in order to get some information. Concretely, we are sharing personal information everyday with such algorithms and it would be annoying to have a leakage of these information. Hence, it seems necessary to have a way to ensure privacy for NLP applications.

When we have a closer look on NLP models, we can easily understand that such models have to embed the words into a continuous space before solving any tasks. Hence, we could naively think that this embedding ensure privacy by itself since the words are embedded into an abstract space and hence, "hidden" in some sense. In reality, for the model to really work, this embedding should encode properties of the words on which it has been trained and hence, this does not completely ensure privacy.

In this report, we are going to study the paper **"Differentially Private Representation for NLP : Formal Guarantee and An Empirical Study on Privacy and Fairness"** written by Lingjuan Lyu, Xuanli He and Yitong Li and published for the 2020 Conference on Empirical Methods in Natural Language Processing [1]. In this paper, the authors propose for the first time a method enabling to guarantee formally the privacy of a word embedding while maintaining a satisfying utility and wiping off discriminations in most cases.

Before talking about the paper, we are going to bring to light some privacy leakages we could face with word embeddings. Then, we will give a state of the art for private word representations, pointing some limitations. Thereafter, we will examine the contribution of the paper and we will present some experiments we have done in order to validate or not the efficiency of their model. All our experiments are in the repo : `https://github.com/RonyAbecidan/PrivateWordEmbeddings`

1 Privacy leakage for NLP representations

In this part, we are going to present in a first time several ways to embed words in NLP so that we can understand intuitively why words embeddings are not private. Then, we will discuss about some experiments we have made showing that word embeddings can effectively leak sensitive information.

1.1 State of the art of Word Embeddings

In this section, we will discuss about different strategies which have been proposed during the decade for having smart word representations. For convenience, let's fix a situation. We are in a sentiment analysis task : given a mail, we want to know if the author was angry, happy or sad.. From the mails, we can create a huge dictionary containing a certain amount of written words and our goal will be to embed cleverly the words of each mail so that an algorithm can perform the sentiment analysis.

The simplest approach consists in simply computing the occurences frequencies of each word in a given mail and put them in a vector. This is called the **bag of words (BOW)** strategy. Thanks to these frequency vectors, we can make an algorithm learn if the author was angry or not rather easily. For example, naturally someone who is angry will tend to write aggressive words and this can be easily captured by an algorithm. Now, concerning the privacy of such simple embedding, we could say that if an attacker has access to the BOW of an email he can deduce if there is potentially a sensitive information in it since this kind of information should have a low occurence frequency. However, it doesn't seem easy to retrieve the information with only the count frequencies. So it's a rather private embedding but, it present some flaws.

For instance, we can quickly see that this type of representation is really limited because it does not catch all the information encoded in a text. Text is not just a "bag of words", some more implicit information are really important like neighboring words and the relation between them. For example, one could want to catch the fact that the words "king" and "queen" are closer in meaning than "orange" and "king". That's why in pratice more complex embeddings are used to achieve better results.

In 2013, a team at Google led by Tomas Mikolov created the method **word2vec**. It is a neural network made of two components : The Skip Gram and The Common Bag Of Words (CBOW).

Given an input word, the goal of the Skip Gram model is to guess its "neighboring" words also called **context words**. On the contrary, we expect from the CBOW to guess a target word given some contexts words. Thanks to their collaboration, these two models enable to obtain a particularly interesting word embedding.

In practice, the embedding produced by word2vec encodes clearly some relations between the word such that using the cosine similarity between two words, we could easily guess if they are close in meaning. Due to that property, this embedding is very useful for NLP tasks but, it intuitively also imply to loose privacy in some sense since they encode richer information about each word. We will see that in details in the next section.

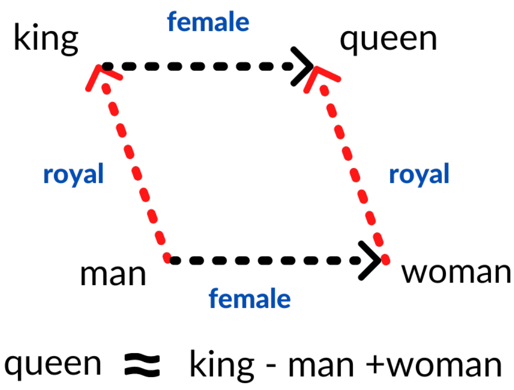


Figure 1: Example of a relation captured by Word2Vec

Finally, researchers have recently proposed other embedding strategies more efficient than word2vec that encode more information about the words. We can think notably of : GloVe (Global Vectors for Word Representation) by Pennington et al.in 2014 and BERT by Google in 2018. In their experiment, the authors of the paper studied have precisely used BERT as a baseline.

1.2 To what extent embeddings are not private ?

We have seen in the previous part that there exists different methods for representing a word in NLP. Globally, the final aim of these embeddings strategy is to have a representation which encodes properties of the words. For instance, we could expect from two similar words to be close in the embedded space. This property of embeddings ensures a good utility for NLP algorithms but also a rather weak privacy.

In order to demonstrate that point, we have conducted three experiments available in **this illustrative notebook.** The goal here was to studied how many information can leak a famous word embedding such as **Word2vec**. In principle, Word2vec is embedding a word through a neural network and hence, the ways the words are embedded into low dimensional vectors is opaque. However, we have shown in simple scenarios that it is clearly possible to deduce private information from these embeddings.

For these experiments, we have imagined the following scenario :

Someone uses an application in which he types some text and, without knowing it, his words are embedded and transmitted to an API enable to suggest him what to type next. Unfortunately for him, a malicious person know his identity and succeed in capturing this transmission : Is his privacy at risk ?

To study that scenario, we considered three gradually realistic cases :

- Situation 1 : We imagine that the user express itself rather simply such that we can find online a dataset containing sentences equal or close to what he has written.

- Situation 2 - We imagine that the user is going to make an anonymous question on a website such as quora.com and the attacker know it and decide to built a dataset made of questions of the same site.
- Situation 3 : We consider again that the user is going to make a question on quora.com but this time the attacker does not know that in advance. In that situation, he decides to use a public dataset made of numerous tweets and hope he could find something interesting

Word2vec is a famous model for embeddings words. We used for these experiments the one from the library **gensim** and, since this is a well-known embedding, we have considered that the attacker will use also this embedding by default on his own dataset and see if he can deduce words he captured from it.

■ For the first situation, we used the dataset of English and French sentences from Anki [1]. Since it's a dataset made for translation purposes, the sentences contained in it are really simple and could be used in practice by non native speakers. In order to simulate the situation, we sampled twice 50% of words from the dataset that is made of **179903** sentences. One of these sets was seen as the user set, i.e. the sentences he already typed on his phone and, the other one was seen as the attacker dataset. The two sets are overlapping but present also differences so the embeddings constructed for each one are maybe close but different.

To see if the attacker can find potentially sensitive information we envisaged fictive sentences that could have been typed by the user. Then, we extract their embedded representation and for each of these words, we displayed the closest word in term of cosine similarity in the embedded space of the attacker[2]. We were not especially confident on the potential effectiveness of this attack since embeddings are by construction really dependent of their training set and here, the distributions of the words were rather different even if there were common sentences. Yet, we succeed in recovering sensitive sentences such as *"my doctor said that I only have a few days left to live"* and *"Il se sent seul"*. In parallel, in other sentences the sensitive information were hidden. For instance, *"My son was ill yesterday"* was recovered by the attacker as *"My* **bag** *was ill yesterday"*.

In order to go further in this study, we have computed the percentage of words in the private model that can be **approximately recovered** by the attacker where, **approximately recovered** means that the private word belongs to the 5 most closest words of the attacker in terms of cosine similarity. At the end, 33% of English words were approximately recovered by the attacker against 41% for the French words and this is not negligible. We have noticed in practice that it's easier to recover French words compared to English ones and this show that the level of privacy leaks in word embeddings depend also on the nature of the languages as we could have expected.

Now, our results have to be moderated. Here we have chosen a very simple situation but, in real life, it is harder for an attacker to capture private words. Let's see now what we have found for our second situation.

■ The second situation is a bit more realistic since the attacker has not at his disposal a part of the user dataset but rather, a dataset from the same distribution. For simulating this situation, we used the famous pairs dataset of Quora[3]. It is made of a set of questions that are potentially duplicates of each other in the sense that they express the same request. Hence, for the user dataset we naturally chose a 40% sample of the first column of questions and, for the attacker, a 70% sample of the second one[4]. We have also considered that in reality the dataset of the user contains not only his own questions but also the ones from the users in the platform that share the same interests. At last, there are around twice more non duplicates questions than duplicates ones. Hence, our study is reasonable since the datasets are not completely correlated.

As we expected, it was more difficult this time to recover private sentences but it was still possible. For instance, we succeed in recovering a real question asked by a user *'How can you lose weight quickly'* into *'How can you lose belly quickly'* that express the same idea. At the end, the attacker approximately recovered 18% of the private words. Now, this situation implied that the attacker knew that the user was typing on quora.com. If now, he does not have any idea where is typing the user, it's an other story... The story of the third and last case !

■ In our last situation, we kept as the user dataset a sample from the first questions column of the quora dataset and, we take as our attacker dataset a completely different one which is made of tweets[5]. This time the attack was clearly compromised since embeddings are completely dependent of the contexts associated to the words and here, the two datasets came from two different distribution.

In this setting, the attacker succeed in approximately recovering only **0.7%** of the private words. Hence, we could think that this is a very weak attack. However, we still succeed in recovering the sensitive sentence *'I really like Trump'* as *'I well like trump'*. So, even in a very tricky situation for the attacker, it seems that it is still possible to recover something sensitive.

[1] `http://www.manythings.org/anki/`

[2] This is in fact the attack scheme we considered for all the cases

[3] `https://www.kaggle.com/c/quora-question-pairs`

[4] This is credible since the attacker could potentially collect any question he wants from the platform by webscrapping

[5] `https://github.com/marsan-ma/chat_corpus`

Here, we have shown with very simple experiments that a famous embedding such as Word2Vec is not completely private. **Even with two completely different datasets, we succeed in recovering sensitive information**. However, in practice if the attacker use a different embedding than the one used for the user dataset, the attack should not lead to satisfying results. But, usually, people tend to use famous pre-trained models that have made their proofs such as BERT. Hence, it may be a baseline for an attacker even if he can't be totally sure of what to use.

Since we are aware of privacy leakage with NLP representations now, we could wonder what is the state of the art for making them private.

2 State of the art of privacy preserving text representations

As said earlier, the success of word2vec enabled to launch a revolution around the manipulation of syntatic and semantic relations among words. As a result, a lot of NLP tasks are now deployed on commercial systems. This has enabled many capabilities and tasks related to text processing, leading to several high-impact applications. With this development comes also a data collect which becomes more and more sensitive which can be very dangerous if it is leaked. That is why researchers have developed many defenses for having private representation for NLP. One naive solution is to remove sensitive attributes (or try to combine different attributes,..Etc). This defense is called **"data anonymization"** and unfortunately, it has been shown that this method is not effective at all because attributes can be highly correlated.

To deal with this privacy issue, the authors discussed about two papers which have proposed some solutions without giving a formal privacy guarantee. Let's see in details these strategies.

The first paper [2] present a training method designed as a defense from the same attack scenario we considered before : The attacker only knows the word representation. For solving the privacy issue, they proposed an architecture inspired by adversarial learning. Notably, they presented a supervised NLP algorithm for a classification task made of a predictive model M which has to solve a particular NLP task like sentiment analysis and, a discriminator D which is seen as an attacker model which try to identify sensitive information like the gender of each user.

For making a privacy preserving NLP model, there are always two opposite aims : we are looking for a word representation enough rich to solve efficiently our NLP task while being enough poor for identifying sensitive attributes. It is in some sense the classic trade off between **privacy** and **utility** and that justify the adversarial architecture proposed.

Mathematically, if we denote b a private attribute that an attacker is trying to capture, θ_M and θ_D respectively the parameters of the NLP model and the attacker model and y the target associated to the words, this leads to the objective :

$$\hat{\theta} = \min_{\theta_M} \max_{\theta_{\mathrm{D}}} \quad \mathcal{X}\left(\hat{\mathbf{y}}\left(\mathbf{x};\theta_M\right), \mathbf{y}\right) - \lambda \cdot \mathcal{X}\left(\hat{b}\left(\mathbf{x};\theta_{\mathrm{D}}\right), b\right)$$

where χ defines the cross entropy function and λ is a hyperparameter which permits to reinforce more or less the adversarial loss during the training process.

Using this strategy, they succeed in making the embedding more private empirically while keeping a reasonable utility in different examples.

The second paper [3] presents different training methods designed as defenses from the same attack scenario. One training method lies on the setting as explained just before. Another strategy is based on the intuition that the attack should be harder if our model learns implicitly to cluster examples with a similar private variable z in the same regions within the embedding space so that they are literally hidden. For applying this intuition, they added in their loss a penalizing term for pairs (x, x') that have similar reconstructions $z(x) \approx z(x')$ and have hidden representations $r(x)$ and $r(x')$ in different region of space. This yields to the following loss :

$$L_m(x, y, z, \theta_r, \theta_p) = -\log P(y|x, \theta_r, \theta_p) + \alpha(0.5 - l(z, z'))||r(x) - r(x')||_2^2$$

where $l(.,.)$ is the Hamming distance, α is a hyperparameter for controlling the penalizing term and (x',z') is sampled from the training set.

On both papers, they have empirically shown that these defenses lead to models with a better privacy on different NLP tasks (sentiment analysis, document classification ..). However, they only provide an empirical privacy, without any formal privacy guarantees. Actually, it is often explained that adding a noise for ensuring a differential private embedding leads too often to poor utility and that's the reason why people discarded this method and prefered to use empirical methods to solve the privacy problem while maintaining a good utility.

Hence, it is intriguing to see that the authors of the article studied claims that they succeed in ensuring a formal privacy guarantee while maintaining a good utility. Let's see what they have done in the next section !

3 How to ensure formally a private word embedding while maintaining utility ?

As we have seen before, ensuring privacy of word representations is a real stakeholder and yet, the past work on this topic does not present any privacy formal guarantee. Here, we are going to see to what extent the authors of the article studied have proposed an algorithm solving this privacy issue both theoretically and empirically while ensuring a good utility.

3.1 Scenario

In the paper, we consider the following scenario :

It exists some NLP model available on a cloud and that can be used by anyone for personal purposes through an API. The user just have to write his text in a raw format, then this text is embedded locally in his computer by the service and it is transmitted to the final model in the cloud. The privacy leak proposed by the author consists in imagining that a potential attacker can access to the embedded representation of the user's dataset during the transmission.

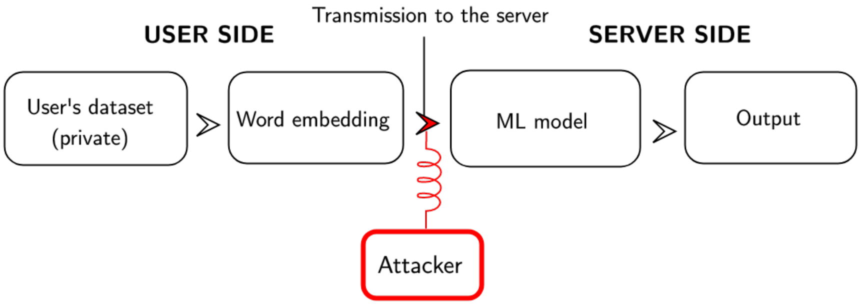


Figure 2: Illustration of the attack scenario proposed in the paper

We noticed that the authors present a case in which the final model is a classifier. Yet, in practice their idea could be used for any type of task in NLP since it is applied to the embedding of the dataset which is a preprocessing necessary for every type of NLP task.

Moreover, we can see that their approach is rather realistic. Indeed, they have considered a case where the attack is done on a testing set of a user rather than on the training set of the online model like we saw in the course and other papers. This kind of situation is in some sense more credible because it's certainly easier to attack the transmission of one particular user rather than the entity that created the online model (which is more likely to be well-protected).

Let's see now how to formalize this scenario and solve its privacy issue.

3.2 Framework

For clarity, we are going to consider here a classification task as the authors.

Let's call $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$ the **data universe** where $\mathcal{X}$ represent all the observations that we can retrieve in a user dataset and $\mathcal{Y}$ the corresponding labels. Please note here that $\mathcal{X}$ is a set of sentences, each one made of several words. In the paper, it is proposed to cut the records of each user into an union of insensitive and sensitive features. By simplicity, we consider in this report that all features should be private.

We recall that we are in a case where we want that each user send implicitly a locally differentially private embedding to the online service. This case corresponds to the **Untrusted curator model** we have seen in the course. In order to be able to satisfy this condition, we need to ensure local differential privacy for each user. Let's define to what it corresponds here.

A word embedding WE is locally (ϵ,δ)-differentially private for the user if :

$$\forall x \sim x' \in \mathcal{X} \;\; \forall O \in \mathcal{O} \;\; \mathbb{P}(\mathsf{WE}(x) \in O) \leq e^{\epsilon}\mathbb{P}(\mathsf{WE}(x') \in O) + \delta$$

where $x \sim x'$ means that the sentence x and the sentence x' differ by at most 1 word and $\mathcal{O}$ is a probability space corresponding to the output space.

When ϵ is very low, this will mean that the embedding will output practically the same vector for neighbouring words. It corresponds to the situation where the embedding is super private but will not be useful for the classification task.

Conversely, when ϵ is high, this will mean that the embedding will be rather well dependent of the user dataset. It corresponds to the situation where the embedding is not really private but useful for the classification task.

At last, δ is a parameter that controls the probability that the word embedding is locally ϵ- differentially private. The lower it is, the better is our privacy guarantee.

The authors of the article have chosen $\delta = 0$ in order to be in the best situation for the privacy and, prove by the same occasion that their strategy doesn't impact too much the utility even in this case. Now, according to the level of sensitive information in the user dataset, we could choose a slightly higher value for δ to gain in utility.

Let's talk now of the strategy of the authors for making a word embedding locally ϵ-DP.

3.3 A privacy preserving embedding

The solution proposed is quite simple. In order to make the user embedding private, one could simply add a noise to each embedded sentence. Doing so, the user will send to the server a local-differentially private word representation that will not leak any information if an attacker succeed in capturing it.

However, this could also imply a good loss of utility for the classifier. Hence, so that they could mitigate this effect, the authors chose to be in a situation where the same level of noise is added to the training embedding of the online model. This simple idea enables to have **a robust target model** according to them.

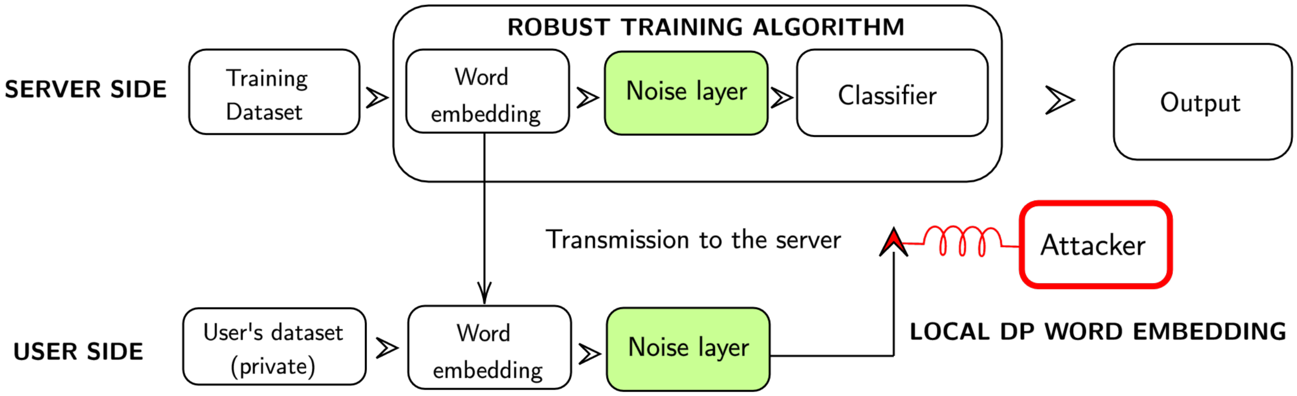


Figure 3: Strategy proposed for making a robust and privacy preserving classifier

Despite it seem to be a good strategy, there is something which is not very clear in this setting. How the training on the server side could be done using the same level of noise as the user side ? In order to do so, this level of noise should be shared from the service to the user but it is not precised in the paper.

Intuitively, we think that the private local embedding is implicitly done by the user phone thanks to an algorithm provided by the online service. Hence, the strategy described above could be indeed applied. But, in this situation, we could not entirely consider that we are in a "Untrusted curator" setting since we need to trust the owners of the service we are using for making our embeddings private before process them. To solve that issue, we could imagine that each user adds a noise to its embedding by his own means to be sure that it will be private at the end even if the service doesn't make it private. We will talk about that situation later.

For the choice of the noise, the authors have opted for a **Laplace noise**. This is consistent with their will to make an ϵ-local differentially private embedding. In order to find a good scale for this noise, they have done a classic preprocessing which consists in normalizing the embedded sentences into the [0,1] range.

Doing so, $\Delta_1(WE) = \max_{x,x'} |WE(x) - WE(x')| = 1,$ and we can take as our noise $n \sim \mathcal{L}ap(\frac{1}{\epsilon})$.

An other interesting idea given in the paper involves a dropout masking enabling to make the embedding even more private. This operation consists in masking some words randomly for each sentence. Intuitively, if we do such masking it's clear that we are hiding information and hence, that will make the embedding more private. Moreover, as they suggest, we can imagine that we want to hide some words putting them to 0 because we know that they are too sensitive. From this idea, they are derived the following result :

Given a sentence $x \in \mathcal{X}$ and $\mathcal{M}$ the dropout mask which masks randomly the words of x with a probability p : $\boldsymbol{x} \odot \mathcal{M}$,

If we suppose that $\mathcal{A}(\boldsymbol{x}) = \text{WE}(\boldsymbol{x}) + \boldsymbol{r}$ is ϵ -differentially private

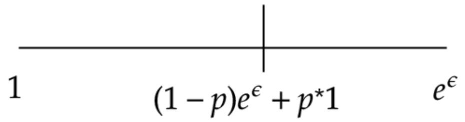
Then $\mathcal{A}(\boldsymbol{x} \odot \mathcal{M})$ is ϵ' -differentially private, where $\epsilon' = \ln[(1-\text{p})e^{\epsilon} + \text{p}]$

Proof :
Let $x \sim x' \in \mathcal{X}$ two sentences that differs only at the i^{th} word. Let S an output sentence of the mechanism $x \to \mathcal{A}(\boldsymbol{x} \odot \mathcal{M})$

- With probability p, $\mathcal{M}(i)$=0 and in that case, $\boldsymbol{x} \odot \mathcal{M} = \boldsymbol{x'} \odot \mathcal{M}$. Hence, $\mathbb{P}(\mathcal{A}(\boldsymbol{x} \odot \mathcal{M}) = S) = \mathbb{P}(\mathcal{A}(\boldsymbol{x'} \odot \mathcal{M}) = S)$
- With probability $1 -$ p, $\mathcal{M}(i)$=1 and in that case, $\mathbb{P}(\mathcal{A}(\boldsymbol{x} \odot \mathcal{M}) = S) \leq e^{\epsilon}\mathbb{P}(\mathcal{A}(\boldsymbol{x'} \odot \mathcal{M}) = S)$
- We deduce from our two previous analysis that,

$$\begin{aligned}
\mathbb{P}(\mathcal{A}(x \odot \mathcal{M}) = S) &= \text{p}\mathbb{P}(\mathcal{A}(x \odot \mathcal{M}) = S|\mathcal{M}(i) = 0) + (1-\text{p})\mathbb{P}(\mathcal{A}(x \odot \mathcal{M}) = S)|\mathcal{M}(i) = 1)\,[6] \\
&\leq \text{p}\mathbb{P}(\mathcal{A}(x' \odot \mathcal{M}) = S) + (1-\text{p})\,[e^{\epsilon}\mathbb{P}(\mathcal{A}(x' \odot \mathcal{M}) = S) \\
&= [(1-\text{p})e^{\epsilon} + \text{p}]\mathbb{P}(\mathcal{A}(x' \odot \mathcal{M}) = S) \\
&= e^{\ln[(1-\text{p})e^{\epsilon}+\text{p}]}\mathbb{P}(\mathcal{A}(x' \odot \mathcal{M}) = S) \quad \square
\end{aligned}$$

Hence, doing a dropout enables to reinforce privacy since we end up with a lowered ϵ. Indeed, we can easily notice that $\epsilon' = \ln[(1-\text{p})e^{\epsilon} + \text{p}] \leq \epsilon$ simply drawing the segment $[1, e^{\epsilon}]$ in which $(1-\text{p})e^{\epsilon} + \text{p}$ lives:



When p is too high, we are masking a lot of words and we see that our privacy parameter $\epsilon' \sim 0$ in that case. This is perfectly consistent since here we are loosing a strong level of information in the process. We have a high privacy but no real utility.

Reversely, when p is too low, we end up with a privacy level as strong as the situation where there is no dropout. Hence, as explained in the paper, there is a tradeoff to make for this value of p.

[6] There was a slight mistake in the paper at this level of the proof. They forgot the conditional conditions.

3.4 Algorithms

We recall below the algorithms proposed by the authors for making a robust and privacy preserving NLP model.

Algorithm 1 Robust Training on the Server

```
Input: Training record $(x_t, y_t)$; Feature extractor $f$; Classifier $C$.
Extraction: $x_r \leftarrow f(x_t)$
Normalization: $x_r \leftarrow x_r - \min(x_r) / (\max(x_r) - \min(x_r))$
Perturbation: $\hat{x}_r \leftarrow x_r + r, r_i \sim \mathrm{Lap}(b)$
Calculate loss $\mathcal{L} = \mathcal{X}(C(\hat{x}_r), y_t)$ and do backpropagation to update $f$ and $C$ .
```

Algorithm 2 Differentially Private Neural Representation (DPNR)

```
Input: Each sensitive record $x_s \in \mathbb{R}^d$; Feature extractor
Parameters: Dropout vector $I_n \in \{0,1\}^d$ ;
Word Dropout: $\tilde{x}_s \leftarrow x_s \odot I_n$ ;
Extraction: $x_r \leftarrow f(\tilde{x}_s)$
Normalisation: $x_r \leftarrow x_r - \min(x_r) / (\max(x_r) - \min(x_r))$;
Perturbation: $\hat{x}_r \leftarrow x_r + r, r_i \sim \mathrm{Lap}(b)$
Output: Perturbed representation $\hat{x}_r$
```

We can notice that the word dropout is applied only in the user side. However, this could potentially help utility for the trained model by analogy with the dropout masking of weights in neural networks.

These algorithms have been implemented by our own means using a homemade embedding. We will discuss about the results we obtained in the last section of this report. For the moment, it's time to talk about the results of the authors ! .

4 Experiments and results of the paper

4.1 Main settings

For their experiments, we can clearly see the will of the authors to show the potential of their strategy. As a result, they decided to use 3 datasets for 2 NLP tasks : sentiment analysis and topic classification. The **TP** (Trustpilot Sentiment) dataset contains reviews associated with a sentiment score and 3 sensitive attributes which are gender, age and location. For the study, they decided to only consider reviews located in the United States, let's denote this dataset **TP-US**. Secondly, they used **AG News** dataset which contains documents and an associated topic considered as the label (we are going to use this dataset also for our experiments). Finally, they remixed a blog posts datasets (**BLOG**) such that it contains blog posts associated with topic labels.

As we said earlier, the authors decided to make the classic data pre-processing detailed in **3.4** which is done also for instance in the paper [3].

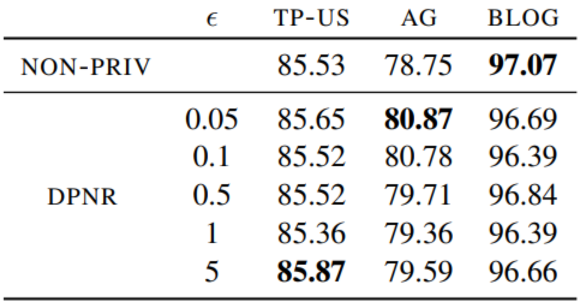
In order to test to what extent their embeddings are private, they have designed some attacks using a MLP classifier which is trying to identify the presence of private attributes[7]. Concerning the choice of the metrics enabling to evaluate the attack performance, the authors decided to use an **empirical privacy**. When this metric is high, that corresponds to a bad attack performance and reversely a good privacy. More precisely, if we denote X the accuracy of the prediction by the attacker (or the F1 score according to the situation), the empirical privacy is defined by the authors as 1-X.

Finally, concerning the embedding method, the authors decided to apply the BERT embedding because it's one of the state of the art method. Then, on top of this embedding, they implement directly a softmax layer to output the class probabilities.

[7] This is called a membership inference attack

4.2 Main privacy results

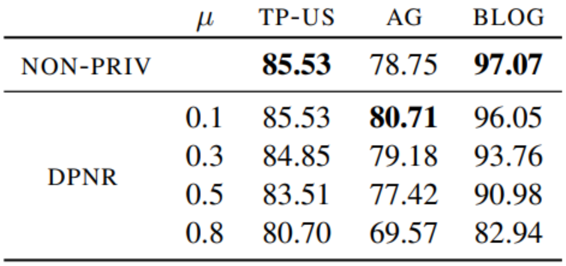
In a first time, the authors wanted to evaluate the impact of the privacy budget and the dropout rate denoted respectively ϵ and μ in order to see if the model behaves well regarding the theoretical expectations. For doing this, they analyzed the variation of the accuracies on the test sets for different values of one parameter while the other was fixed.



	ϵ	TP-US	AG	BLOG
NON-PRIV		85.53	78.75	**97.07**
DPNR	0.05	85.65	**80.87**	96.69
	0.1	85.52	80.78	96.39
	0.5	85.52	79.71	96.84
	1	85.36	79.36	96.39
	5	**85.87**	79.59	96.66

Figure 4: Results with μ fixed and ϵ varying

These results can be quite surprising. Indeed, we are used to see (with respect to the theoretical analysis) a decreasing accuracy as the parameter ϵ increases. We can interpret ϵ as a privacy budget : small values of ϵ require to provide very similar outputs when given similar inputs, and therefore provide higher levels of privacy; large values of ϵ allow less similarity in the outputs, and therefore provide less privacy. Here, it seems that **variation on ϵ does not affect the test accuracy** on every dataset. They made the hypothesis that the BERT embedding is in some sense resilient to the noise by construction for explaining this phenomenon. Now let's take a look at when we vary μ :



	μ	TP-US	AG	BLOG
NON-PRIV		**85.53**	78.75	**97.07**
DPNR	0.1	85.53	**80.71**	96.05
	0.3	84.85	79.18	93.76
	0.5	83.51	77.42	90.98
	0.8	80.70	69.57	82.94

Figure 5: Results with μ varying and ϵ fixed

Here, the results behave as expected. A large value of μ means a better privacy policy and so a worse test accuracy.

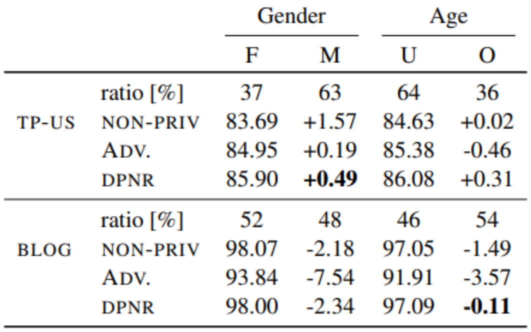
In a second part, the authors wanted to show how could perform an attacker trying to recover sensitive attributes of their private embeddings. They have made a simple attack based on a 2-layer MLP classifier with 512 hidden units. However, they don't precise how the attacker succeed in making a dataset from the same distribution as the user set and hence we deduced that they were in a **white box setting**. They have shown empirically that the attack was rather good for a non-private embedding and rather poor with private embedding made using their strategy, justifying all the point of this paper. More precisely, they observed that the attacker accuracy was lower than the classifier which returns the majority class for every input sentence and this is an improvement compared to the adversarial learning strategy from [3].

Now, on top of having a privacy preserving model, the authors claimed that it's also fairness preservering. Let's see exactly what they mean by that.

4.3 Fairness

Privacy is not the only field which knows a growth of interest as the digital consciousness of the population is rising. Indeed, fairness has also gained a lot of attention and many researchers try to work on it. Just like privacy, fairness has a lot of interpretation, but mathematical formalization is not well developed for the moment in this field.

For evaluating the fairness of their algorithm, the authors decided to split several datasets (BLOG and TP-US) among different demographic groups (age and gender). Then they measured the tests accuracies on these different subgroups. Here are the results :



		Gender		Age	
		F	M	U	O
TP-US	ratio [%]	37	63	64	36
	NON-PRIV	83.69	+1.57	84.63	+0.02
	ADV.	84.95	+0.19	85.38	-0.46
	DPNR	85.90	**+0.49**	86.08	+0.31
BLOG	ratio [%]	52	48	46	54
	NON-PRIV	98.07	-2.18	97.05	-1.49
	ADV.	93.84	-7.54	91.91	-3.57
	DPNR	98.00	-2.34	97.09	**-0.11**

Figure 6: Test accuracies over different datasets according to different subgroups with respect to the demographic variables age and gender[8]

Globally, they observed only empirically that their method could help also to improve fairness. But something perturbed us. Indeed, the will of the paper was to provide a formal guarantee for privacy being at the antipode of the empirical methods designed so far. On the other hand, they presented only empirically their results on fairness, with a weak definition of it. We found this quite ironical that they do what they criticised for the privacy field.

Moreover, we have to take care of these results: as said before, the BERT seems to be resilient to the noise but there is a possibility that these results may **depend a lot on the choice of the embedding**. That's why, we propose to take a look at this issue by using a more simple embedding for our experiments in the next section. One last thing to remark is that results can depend also on the choice of the data pre-processing (we can intuitively think that some pre-processing techniques can leak more or less information) but we will not analyze this part.

5 Let's test the strategy of the authors !

In this section we are going to discuss some results we observed reproducing the model proposed by the authors.

5.1 Details about the implementation

In order to test the algorithm of the authors, we have used the deep learning library **pytorch** and we have adapted **this tutorial**. You can see the code of our experiment on **this illustrative notebook** . We have chosen like the authors two classification tasks and our embedding procedure is very simple and consists in mapping words to vector of low dimension through a linear layer and then apply a softmax layer[9].

We have used as our datasets 2 famous datasets : The **AG News** dataset which contains news associated to some categories and which has been also used by the authors and, the **YelpReviewPolarity** dataset which is made of reviews associated to one sentiment. From each dataset, we have extracted an intermediate set and a test set. Then, the intermediate set was cutted into a training set and a validation set.

We have tested many situations that will be detailed after and these tests have been done for two embedding dimensions : **32** and **64**. Moreover, the computations have been done in a reasonable time thanks to the use of a GPU from Google Colab'. Nevertheless, to achieve this computation time we chose to keep for the intermediate set a size of **30000**, and we cut it in 95% for the training set and 5% for the validation set. The AG test set is made of sentences and the Yelp test set is made of sentences The goal of our implementation was to test to what extent the strategy of the authors conserve a good utility. Hence, to be able to answer to that question, we have considered several situations. For each test accuracy computed, it has been actually computed **5 times** and averaged.

[8] "Ratio" means the ratio between two subgroups of the demographic variable

[9] A softmax layer is the final output layer in a neural network that performs multi-class classification. It's a combination of a linear layer and a softmax activation function

5.2 To what extent can we consider that the training is "robust" ?

The authors claimed that their strategy enable to ensure privacy while guaranteeing utility and qualified the training phase of the online model as a **robust** training. However, surprisingly, they haven't compare the results obtained using their robust training against the classic training which doesn't involve a noise layer. Hence, we propose to test that situation to understand to what extent their training is robust compared to the classical one.

Like the authors, we have studied the impact of the noise level and the dropout rate varying one parameter while fixing the other. Notably we have proposed two scheme :

- ϵ in $\{0.05, 0.1, 0.5, 1, 2, 3, 4, 5\}$ while $\mu = 0$
- μ in $\{0.1, 0.3, 0.5, 0.8\}$ while $\epsilon = 4$ [10]

You can see in the next page the test accuracies obtained with an embedding dimension of 32. The results obtained with an embedding of 64 were similar and hence, we propose to not discuss about them for avoiding repetitions in our discussion.

Globally, we clearly observe that the robust strategy helps only with ϵ rather high, close to $3, 4, 5$. Indeed, for lower value of ϵ, we observe that the classic training algorithm which does not use a noise layer enable to obtain better test accuracies compared to the "robust" one. Yet, in practice it is common to use rather small value for ϵ so this is a bit problematic. However, we have seen previously that the authors have obtained very good results in term of accuracy with their robust strategy. How can we explain the gap between our results and theirs ? We maybe have some explications.

At first, we have used an homemade embedding which is very simple. It is certainly so simple that it can't be resilient to a noise addition. On the contrary, the authors have used the **BERT** embedding within which they explain that there is a denoising training procedure that could be the reason of their satisfying results. Hence, we could conclude now that their strategy is unfortunately dependent of the embedding used. This is in some sense a limitation.

Moreover, here we have considered like before that all the words are private and should be protected. However, in their case they add a noise only to sensitive attributes and that can also explain the gap we observed between our results.

We also observed in our case that the dropout is fatal for the utility on the contrary to the authors. In that case, we can deduce again that the impact of the dropout on the final result depend on the embedding.

[10] This choice of ϵ was made after seeing that it was a value acceptable for ensuring a reasonable privacy and utility

AG	**4 classes**	**Embedding dim = 32 | 5 epochs | 5 runs**
Non private	**0.80**	/
$\mu = 0$	**Non Robust**	**Robust**
$\epsilon = 0.05$	**0.26**	0.25
$\epsilon = 0.1$	**0.26**	0.25
$\epsilon = 0.5$	**0.31**	0.25
$\epsilon = 1$	**0.36**	0.25
$\epsilon = 2$	**0.48**	0.30
$\epsilon = 3$	**0.57**	0.51
$\epsilon = 4$	0.63	**0.66**
$\epsilon = 5$	0.67	**0.73**
$\epsilon = 4$	**Non Robust**	**Robust**
$\mu = 0.1$	0.53	**0.58**
$\mu = 0.3$	**0.37**	0.35
$\mu = 0.5$	0.32	**0.33**
$\mu = 0.8$	**0.27**	**0.27**
Yelp	**2 classes**	**Embedding dim = 32 | 5 epochs | 5 runs**
Non private	**0.88**	
$\mu = 0$	**Non Robust**	**Robust**
$\epsilon = 0.05$	**0.50**	**0.50**
$\epsilon = 0.1$	**0.51**	0.50
$\epsilon = 0.5$	**0.54**	**0.54**
$\epsilon = 1$	**0.58**	0.51
$\epsilon = 2$	**0.64**	0.60
$\epsilon = 3$	0.70	**0.75**
$\epsilon = 4$	0.73	**0.84**
$\epsilon = 5$	0.75	**0.82**
$\epsilon = 4$	**Non Robust**	**Robust**
$\mu = 0.1$	**0.69**	0.68
$\mu = 0.3$	**0.60**	**0.60**
$\mu = 0.5$	**0.56**	0.54
$\mu = 0.8$	**0.52**	0.5

Figure 7: Results obtained with the strategy of the authors

5.3 From the Robust strategy to the Adaptative strategy

If you remember well, we said in section 3 that, in order to be able to add the same level of noise in the user side and the server side, that means that in some sense this noise is not added by the user and hidden to the service but rather given by the service and added locally in the user device. Hence, the user should trust that the service has really made private his embeddings.

Here, we propose a new strategy potentially able to maintain utility in the situation where the user is the one who put the noise to his embeddings and the server has strictly no idea about his level. However, we consider that at least, the nature of the noise added is known by the server (for instance a Laplace noise).

Our strategy is simple. We propose to adapt the training of the online model so that it can potentially become resilient to any level of noise. For doing that, we are going to add a noise to each training embedding which decreases through the epochs. Hence, intuitively that could make the model learn how to react to different level of noises. We called that strategy the **Adaptative strategy** and that could be resumed via the following algorithm :

```
Algorithm 3 : Adaptative Training on the Server
Input: Training record $(x_t, y_t)$ ; Feature extractor $f$; Classifier $C$.
1: Extraction: $x_r \leftarrow f(x_t)$
2: Normalization: $x_r \leftarrow x_r - \min(x_r) / (\max(x_r) - \min(x_r))$
3: Perturbation: $\hat{x}_r \leftarrow x_r + r, r_i \sim \mathrm{Lap}(\frac{1}{\epsilon_n})$ with $\epsilon_n$ a decreasing sequence indexed by the epoch
4: Calculate loss $\mathcal{L} = \mathcal{X}(C(\hat{x}_r), y_t)$ and do backpropagation to update $f$ and $C$ .
```

We have tested at first our strategy with an embedding dimension of 32 and the sequence $\epsilon_n = \frac{1}{n}$. Unfortunately, it was a complete failure, the test accuracy was as good as a randomness. We thought about it and we deduced that, since a high noise level induced by a small ϵ is clearly too strong for our embedding, there is maybe a bad effect from starting from $\epsilon = 1$ until $\epsilon = 0.2$ since it can leads to a model that can't learn properly due to too noisy embeddings from the beginning. Hence, we proposed an other decreasing sequence that takes into account that remark : $\epsilon_n = \frac{10}{n}$. Moreover, we had the feeling that the embedding size was maybe too small and that increasing it could help the classifier to react better because it will have more information at his disposal even if they are noisy ones. This time, it was an other story. We put in the next page the results we obtained.

What we see is really interesting and could lead to further investigation. It seems that the adaptative strategy could be helpful precisely in situations where the robust strategy isn't efficient, i.e. when the noise level is rather high. On the contrary, when the noise level is rather low, the robust strategy seems to perform better than our adaptative one. So, in some sense, these are complementaries strategies and one could think about using one or the other according to the maximum level of noise added by the user.

AG	4 classes	Embedding dim = 64 | 5 epochs | 5 runs	
Non private	0.77	/	
$\mu = 0$	**Non Robust**	**Robust**	**Adaptative**
$\epsilon = 0.05$	**0.26**	0.25	**0.26**
$\epsilon = 0.1$	**0.26**	0.25	**0.26**
$\epsilon = 0.5$	0.3	0.25	**0.34**
$\epsilon = 1$	0.36	0.25	**0.42**
$\epsilon = 2$	0.46	0.31	**0.5**
$\epsilon = 3$	0.54	0.42	**0.57**
$\epsilon = 4$	0.59	0.50	**0.64**
$\epsilon = 5$	0.66	**0.68**	0.63
$\epsilon = 4$	**Non Robust**	**Robust**	**Adaptative**
$\mu = 0.1$	0.53	0.47	0.40
$\mu = 0.3$	**0.37**	0.29	**0.37**
$\mu = 0.5$	0.30	**0.28**	**0.28**
$\mu = 0.8$	**0.26**	0.25	**0.26**
Yelp	**2 classes**	**Embedding dim = 64 | 5 epochs | 5 runs**	
Non private	**0.89**		
$\mu = 0$	**Non Robust**	**Robust**	**Adaptative**
$\epsilon = 0.05$	0.50	0.50	**0.51**
$\epsilon = 0.1$	0.51	0.50	**0.53**
$\epsilon = 0.5$	0.53	0.61	**0.62**
$\epsilon = 1$	0.56	**0.64**	0.57
$\epsilon = 2$	0.60	0.62	**0.75**
$\epsilon = 3$	0.65	0.75	**0.77**
$\epsilon = 4$	0.68	**0.84**	0.8
$\epsilon = 5$	0.72	**0.81**	0.79
$\epsilon = 4$	**Non Robust**	**Robust**	**Adaptative**
$\mu = 0.1$	0.65	**0.76**	0.57
$\mu = 0.3$	**0.61**	0.52	0.58
$\mu = 0.5$	**0.57**	0.50	0.50
$\mu = 0.8$	**0.51**	0.50	0.50

Figure 8: Results obtained with the strategy of the authors against our adaptative strategy

Conclusion

This paper was very interesting and tackled a serious problem in a new perspective. Notably, the authors have demonstrated for the first time that it's possible to make a formally private word embedding that maintains a good utility of an NLP model. However, in our experiment we have tested their strategy and we have understood that its efficiency is dependent on the embedding method. That being said, we succeed in proposing a promising strategy that could help to reduce this problem. In order to see to until where this strategy could lead us, it would be interesting to continue the training over a higher number of epochs. Doing more epochs could potentially help the model to react to very high noise levels. At last, one could also think about proposing a smarter sequence of ϵ_n to see the impact on the final result.

References

[1] Lingjuan Lyu, Xuanli He, and Yitong Li. *Differentially Private Representation for NLP: Formal Guarantee and An Empirical Study on Privacy and Fairness*. 2020. arXiv: 2010.01285 [cs.LG].

[2] Yitong Li, Timothy Baldwin, and Trevor Cohn. "Towards Robust and Privacy-preserving Text Representations". In: *CoRR* abs/1805.06093 (2018). arXiv: 1805.06093. URL: http://arxiv.org/abs/1805.06093.

[3] Maximin Coavoux, Shashi Narayan, and Shay B. Cohen. *Privacy-preserving Neural Representations of Text*. 2018. arXiv: 1808.09408 [cs.CL].