ON THE PRIVACY-UTILITY TRADE-OFF IN DIFFERENTIALLY PRIVATE HIERARCHICAL TEXT CLASSIFICATION

Dominik Wunderlich, Daniel Bernau
SAP
Karlsruhe, Germany
firstname.lastname@sap.com

Javier Parra-Arnau
Karlsruhe Institute of Technology
Karlsruhe, Germany
javier.parra-arnau@kit.edu

Francesco Aldà
SAP
Karlsruhe, Germany
firstname.lastname@sap.com

Thorsten Strufe
Karlsruhe Institute of Technology
Karlsruhe, Germany
strufe@kit.edu

July 5, 2021

ABSTRACT

Hierarchical models for text classification can leak sensitive or confidential training data information to adversaries due to training data memorization. Using differential privacy during model training can mitigate leakage attacks against trained models by perturbing the training optimizer. However, for hierarchical text classification a multiplicity of model architectures is available and it is unclear whether some architectures yield a better trade-off between remaining model accuracy and model leakage under differentially private training perturbation than others. We use a white-box membership inference attack to assess the information leakage of three widely used neural network architectures for hierarchical text classification under differential privacy. We show that relatively weak differential privacy guarantees already suffice to completely mitigate the membership inference attack, thus resulting only in a moderate decrease in utility. More specifically, for large datasets with long texts we observed transformer-based models to achieve an overall favorable privacy-utility trade-off, while for smaller datasets with shorter texts CNNs are preferable.

***K*eywords** Anonymization, Text Classification

1 Introduction

Organizing large corpora of unstructured data such as text documents, news articles, emails, and support tickets in an automated manner is a considerable challenge [12] due to the inherent ambiguity of natural languages. However, the automated classification of unstructured data overcomes manual data labelling activities and thus is a key capability for organizing data at scale [36]. Due to the wide range of applications, hierarchical text classification (HTC) has received particular interest by the Natural Language Processing (NLP) community in recent years [21, 32, 2, 29]. HTC leverages machine learning to efficiently organize documents into taxonomies, predicting multiple labels in a predefined label hierarchy based on the content of the document at hand.

After a data owner has trained an HTC model, it may be necessary to share models with data analysts such as contractors, customers, or even the general public. Sharing a model yields information leakage risks w.r.t. the confidentiality of the training data [34, 27, 43, 5]. The cryptographic all-or-nothing approach does not suffice for mitigation of such leakage risks since it does not alter what the model actually learns from the training data. Instead, to mitigate information leakage risks data owners can alter or suppress information in the training data during model training. Differential privacy (DP)[1] limits information leakage by anonymizing the training data or model training function [7, 1, 13]. However, as any anonymization technique, DP introduces an inherent trade-off between privacy and utility (i.e., informative value), which means that a stronger privacy guarantee implies a decrease in utility and vice versa. Balancing this trade-off is especially hard when training artificial neural networks (ANN), since the resulting decrease in model utility can only be assessed empirically after the model has been completely trained [1]. Privacy can similarly be assessed empirically with membership inference (MI) attacks [34], which aim at identifying single instances of the training data by sole access to the trained model and have been illustrated to be mitigated by DP before [27, 33]. However, a rather large gap has been observed between the high theoretical risk for MI that can be derived from DP guarantees and the lower empirical MI risk posed by factual state-of-the-art MI attacks [14]. Consequently, choosing the anonymization strength via privacy parameter ε remains a challenging problem since a data owner can either choose to lower the theoretical or empirical MI risk. In addition, model utility and MI risk do not only depend on each other, but further hinge on several other factors such as model architecture, hyperparameters, and training data.

Our work compares the privacy-utility trade-off between multiple text classification models by quantifying privacy under MI, and mitigating MI with DP. Unlike previous studies on the privacy-accuracy trade-off for numerical or image data [33, 4], our work focuses on textual data which requires different sophisticated ANN architectures, such as Transformers [37]. Furthermore, we consider ANN for hierarchical classification, which in general are more complex than ANN for standard (non-hierarchical) classification tasks. The main contributions of this work are:

- Comparing the privacy-utility trade-off for three widely used HTC ANN architectures on three reference datasets. In particular, we consider Bag of Words (BoW), convolutional neural networks (CNN) and Transformer-based architectures.
- Evaluating how these DP models are susceptible to an adversary who aims to ascertain whether or not a certain record (e.g., an email) is present in the training data. In other words, we connect the privacy parameter ε to an ML specific threat model, shedding light on the relationship between DP and MI attacks on HTC machine-learning tasks.
- Recommending HTC model architectures and privacy parameters for the practitioner based on the privacy-utility trade-off under DP and MI.

The remainder of this paper is structured as follows. Section 1 recalls key aspects of DP and MI attacks. Section 3 introduces our methodology for modelling the privacy-utility trade-off in HTC. Section 4 presents the results of the conducted experiments, which are subsequently discussed in Section 5. Section 6 reviews the state of art relevant to this work. Conclusions are drawn in Section 7, which also identifies some directions for future research.

2 Preliminaries

This section introduces key concepts from differential privacy, membership inference and hierarchical text classification that are used in this work.

2.1 Differential Privacy

In DP [7] a statistical aggregation function $f(\cdot)$ is evaluated over a dataset $\mathcal{D}$ and perturbed by a trusted curator before the result of $f(\mathcal{D})$ is returned to the data analyst. By means of perturbation DP prevents an adversary with arbitrary auxiliary side knowledge from confidently deciding whether $f(\cdot)$ was evaluated on $\mathcal{D}$, or some neighboring dataset $\mathcal{D}'$ differing in one element. Assuming that every participant in $\mathcal{D}$ is represented by a single record $d \in \mathcal{D}$, privacy is intuitively provided to any individual. Especially since the information learned about the individual from the database output is limited to what an attacker could learn from the data without the individual taking part in the computation of $f(\cdot)$. In other words, the purpose of DP is to hide the presence or absence of any single individual within $\mathcal{D}$. Definition 1 introduces the DP privacy parameters (ϵ, δ).

Definition 1 **((ϵ, δ)-Differential privacy [8])** *A randomized mechanism $\mathcal{M}$ on a query function f satisfies (ϵ, δ)-differential privacy for $\delta > 0$ if, for all pairs of neighboring databases $\mathcal{D}, \mathcal{D}'$ and for all $\mathcal{O} \subseteq$ range($\mathcal{M}$),*

$$\Pr\{\mathcal{M}(\mathcal{D}) \in \mathcal{O}\} \leq \exp(\epsilon) \Pr\{\mathcal{M}(\mathcal{D}') \in \mathcal{O}\} + \delta. \quad (1)$$

DP mechanisms for numerical data perturb the original query value $f(\mathcal{D})$ by adding noise. The amount of noise that needs to be added depends on the maximum difference between the query functions $f(\mathcal{D})$ and $f(\mathcal{D}')$, referred to as *global sensitivity* and introduced in Definition 2.

[1] For conciseness, throughout this work we use the acronym DP to refer to both "differential privacy" and its adjective form "differentially private".

Definition 2 (Global ℓ_2-sensitivity [7]) *Let $\mathcal{D}$ and $\mathcal{D}'$ be neighboring databases. The global ℓ_2-sensitivity of a function f, denoted by Δf, is defined as*

$$\Delta f = \max_{\forall \mathcal{D}, \mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_2.$$

Several noise distributions such as the Laplace and the Gaussian distribution have been used in DP mechanisms [8]. For a fixed ε, the higher the sensitivity Δf of the query function $f(\cdot)$, the more noise needs to be added. As a matter of fact, satisfying Definition 1 requires more noise when $f(\mathcal{D})$ and $f(\mathcal{D}')$ can differ significantly. On the other hand, for fixed Δf, the smaller ε, the more noise is added: when ε is very small, Definition 1 nearly requires that the probabilities on both sides of Equation 1 be equal, which needs the randomized function $\mathcal{M}$ to give very similar results for all pairs of neighbor data sets; adding significant amounts of noise is a way to attain this.

In this work we rely on the gradient-perturbation approach suggested by Abadi et al. [1] to perturb the Adam optimizer[2] for neural network training. We refer to the perturbed Adam optimizer as DP-Adam. A DP optimizer for neural network training is represented by a randomized training mechanism $\mathcal{M}_{nn}$ that updates the weight coefficients θ_t of a neural network per training step $t \in \{1, \dots, T\}$ with $\theta_t \leftarrow \theta_{t-1} - \alpha(\tilde{g})$, where $\tilde{g} = \mathcal{M}_{nn}(\partial loss/\partial\theta_{t-1})$ denotes a perturbed gradient and α is some scaling function on $\tilde{g}$ to compute an update (e.g., learning rate). In this update process, DP is attained by the Gaussian mechanism of Theorem 1.

Theorem 1 (Gaussian Mechanism [8]) *Let $\epsilon \in (0,1)$ be arbitrary. For $c^2 > 2ln(\frac{1.25}{\delta})$, the Gaussian mechanism with parameter $\sigma \geq c\frac{\Delta f}{\epsilon}$ satisfies (ϵ, δ)-DP, when adding noise scaled to $\mathcal{N}(0, \sigma^2)$.*

After T steps, $\mathcal{M}_{nn}$ outputs a DP weight matrix θ that is used by the prediction function $h(\cdot)$ of a neural network. DP-Adam bounds the sensitivity of the computed gradients by a clipping norm $\mathcal{C}$, based on which the gradients are clipped before perturbation. Since weight updates are performed iteratively during training, a composition of mechanism executions is required until the training step T is reached and the final private weights θ are obtained. We measure the privacy decay under composition by tracking the noise levels σ of the Gaussian mechanism under Rényi DP [25] (RDP), and transform the aggregate RDP privacy decay again to DP privacy parameters. To our knowledge RDP yields the tightest estimation for the composed DP privacy decay at the time of writing.

2.2 Membership Inference

Membership inference attacks strive for identifying the presence or absence of individual records in the training data of a machine learning model. We shall assume such data belongs to a data owner and refer to the trained machine learning model as *target model* and the data owner's training data as $\mathcal{D}^{train}_{target}$.

Our work uses the white-box MI attack proposed by Nasr et al. [27]. Essentially, the white-box MI attack assumes an honest-but-curious adversary with access to the trained prediction function $h(\cdot)$ corresponding to the trained target model. The white-box MI adversary is assumed to observe certain internal and external features of $h(\cdot)$: the losses $L(h(x;W))$, gradients $\frac{\delta L}{\delta W}$ and the learned weights W of $h(\cdot)$.

In addition, the white-box MI adversary strives for learning a binary classifier, the *attack model*, that allows to classify data into members and non-members (with respect to the target model of the training dataset) with high accuracy. The adversary is assumed to know a portion of the training and test data $\mathcal{D}^{train}_{target}$ and $\mathcal{D}^{test}_{target}$, and generates features for training the attack model by passing the known records repeatedly through the trained target model. Nasr et al. [27] assumed the portion of known records at 50% and we follow this assumption to allow comparison. The accuracy of an MI attack model is typically evaluated on a balanced dataset including members (target model training data) and an equal number of non-members (target model test data). An illustration of the white-box MI attack is shown in Figure 1.

2.3 Hierarchical Text Classification

Hierarchical text classification (HTC) addresses the task of classifying texts into a hierarchy of classes and sub-classes, as for example into a folder structure. Text classification can be seen as a special case of HTC with only one hierarchy level without any sub-categories. Text classification is an important task in the area of Natural Language Processing (NLP) [20] that refers to a variety of computational techniques to automatically analyze natural language data. Since natural language data comprises several forms of ambiguities, NLP tasks are difficult to solve computationally. However, deep Artificial Neural Networks (ANN) are achieving state of the art results for NLP tasks [11].

[2] The Tensorflow privacy package was used throughout this work: `https://github.com/tensorflow/privacy`.

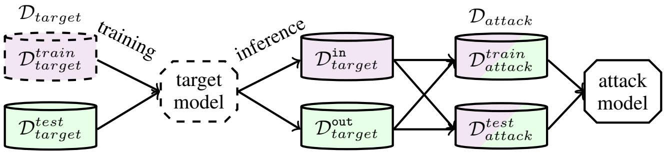

$\mathcal{D}_{target}$
training
$\mathcal{D}^{train}_{target}$
$\mathcal{D}^{test}_{target}$
target model
inference
$\mathcal{D}^{\mathtt{in}}_{target}$
$\mathcal{D}^{\mathtt{out}}_{target}$
$\mathcal{D}_{attack}$
$\mathcal{D}^{train}_{attack}$
$\mathcal{D}^{test}_{attack}$
attack model

Figure 1: White-box MI with attack features. DP perturbation is applied on the target model training (dashed). The data that was used by the data owner during target-model training is colored: training (violet) and validation (green).

Before a text can be used as input for an ANN, it has to be tokenized, which means that the text has to be split up into a sequence of smaller units called tokens. A natural tokenization technique is splitting the text into a sequence of words, so that each token represents a word. After tokenization, the token sequence is converted to an integer sequence since ANN only take numbers as input. During the conversion, a vocabulary is created that maps each token to a unique integer so that the same token is always converted to the exact same integer. The size of the vocabulary then represents the number of unique tokens in the text. Modern ANN for NLP employ embeddings at the first layer to capture the meaning of each token. An embedding is a token's vector representation of length n, that embeds the token into an n-dimensional vector space [11]. A useful embedding maps semantically similar tokens to the same region in the vector space. A shortcoming of embeddings such as Word2Vec [23, 30] is that a word is always assigned to the same vector, ignoring previous and subsequent words. Peters et al. [31] overcome this limitation with the deep contextualized word representations technique, that consists of a right-to-left and a left-to-right recurrent neural network, each trained on a task called language modelling: predicting the next word for a given sequence of words in a text. After pre-training, in order to calculate a contextual embedding for a given word, Peters et al. [31] evaluate its context words with both recurrent neural network models and concatenate their outputs. In our work, we consider three different ANN architecture types for text classification.

First, Bag-of-Words (BoW) classifiers, which represent ANN with a simple architecture for text classification. BoW models interpret a text as a bag of words, which means that the word order is ignored. After the embedding layer the word vectors are added or averaged, and passed through one or more feed-forward layers with a softmax activation at the last layer. A notable example is the fastText classifier [15], which achieves high accuracies and is computationally efficient due to using a single feed-forward layer for classification.

Second, Convolutional Neural Networks, which are particularly well suited for detecting position-invariant patterns in a text. After the embedding layer, one or more convolutional layers are applied to the sequence of word vectors. A convolutional layer applies a discrete convolution on the input vector using a filter of a given width, whose values are seen as learnable weights. Usually, a convolutional layer consists of multiple filters, resulting in multiple feature maps. In the end, the resulting feature maps are again passed through one or more feed-forward layers, with a softmax activation at the last layer. A simple example for a CNN text classifier was published by Kim [16] who uses one convolutional layer with multiple filter widths followed by a pooling and a fully connected layer that represents the output layer.

Third, Transformers, which are building blocks for ANN that apply a mechanism called self-attention to the sequence of word vectors returned by the embedding layer. Self-attention computes an attention score for every word that models the relationship to each other word in the sequence. Thereby Transformers improve processing of longer texts. In general, Transformer-based ANN use much deeper network architectures than the two previously mentioned ANN architecture. Thus, Transformers need to be trained on a larger amount of text corpora, which makes it very costly to train a transformer model from scratch and thus often pre-trained Transformer-based models are used. Transformers can be categorized into auto-regressive and auto-encoding models, and especially auto-encoding models are considered superior due to their ability to consider bidirectional context during pre-training. A notable auto-encoding model is Bidirectional Encoder Representations from Transformers (BERT) [6], which is available in several pre-trained versions varying in their depth. The base model size comprises twelve transformer blocks. It is worth mentioning that each Transformer block comprises multiple layers, including a self-attention and a fully-connected layer.

3 Methodology

This section describes our methodology for quantifying and comparing the privacy-utility trade-off in HTC for three relevant model architectures, under several utility and MI metrics.

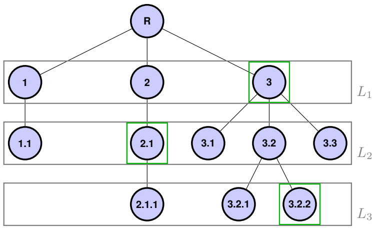


Figure 2: Inconsistent LCL classifier predictions (*green*).

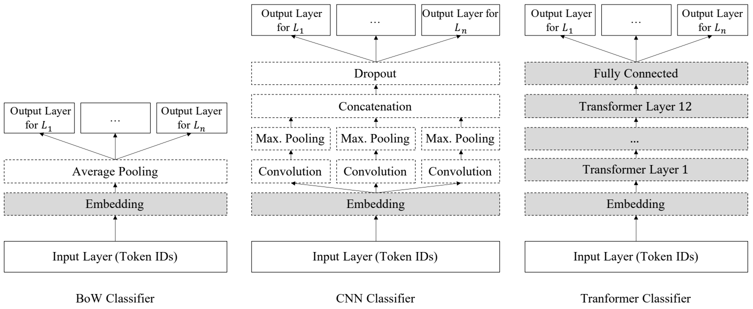


Figure 3: Architecture of the HTC models used in the experiments. Pre-trained layers are marked *grey*.

3.1 HTC Model Architectures

Recent text classifiers achieve the highest accuracy when a fully connected layer with *softmax* activation is used as output layer. We follow this approach for HTC by simultaneously training multiple output layers, one per hierarchy level. Each output layer then has a softmax activation function for predicting the classes on level L_n of the given HTC task. Our general HTC architecture thus consists of only one model and does not ignore the class hierarchy. This yields two main advantages. First, since only one model is trained, our HTC classifier can retain the benefits of *flat* classifiers in the sense of exhibiting lower overall privacy loss, compared to classifiers that train multiple models (i.e., one per hierarchy). In consequence computation time is also reduced. Secondly, the data analyst can still retrieve the prediction probabilities per level, instead of just those of the leaf nodes [3].

Our simultaneous training approach, however, has the disadvantage that a post-processing step is needed to obtain predictions consistent with the hierarchy. This disadvantage is common among classifiers with the local classifier per level (LCL) characteristic, since each local classifier makes predictions independently on its level. We show an example in Figure 2, where the prediction of the L_2 classifier does not coincide with the prediction of the other classifiers (L_1, L_3). We suggest to resolve these inconsistencies by multiplying the softmax probabilities along the path from the root to each node. After comparing the multiplied probabilities, we output the path with the highest probability as prediction. With this post-processing step we ensure that classifiers based on our design approach only make consistent predictions.

Even though we defined the output layer architecture for our HTC beforehand, multiple options for the architecture of the input and intermediary layers exist. We consider architectures widely used in state-of-the-art text classification as the basis for formulating a DP hierarchical text classifier. In the sequel, we briefly describe the three architectures employed in our methodology.

Bag-of-Words Classifier

The BoW classifier is based on the architecture of *fastText* [15] which first embeds each token and then computes the mean of all embeddings before the output layer. Prior to training, the embedding layer is initialized with the widely used GloVe embeddings[3] that are pre-trained on the "Wikipedia 2014" and "Gigaword 5" corpora [30, 17]. For training we use the Adam optimizer.

The classifier architecture is visualized in Figure 3.

[3] `https://nlp.stanford.edu/projects/glove/`

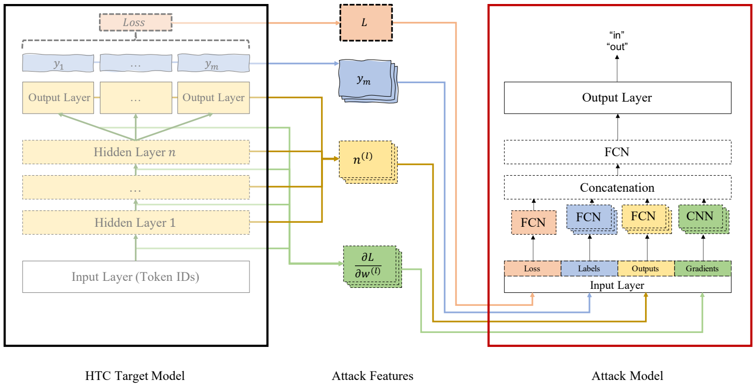


Figure 4: Attack model architecture and observed attack features from an HTC target model.

Convolutional Neural Network Classifier

The CNN classifier we shall use is based on the architecture proposed by Kim [16]. Their architecture first applies three convolutional blocks to the embeddings, and secondly the output of the convolutional blocks is concatenated and put through a dropout layer, which is finally followed by the output layer. Each convolutional block consists of a convolutional layer with a different filter size followed by a maximum pooling layer. The idea behind the maximum pooling layers is to capture the most important step resulting from the convolutional layer. We replaced the original output layer with several output layers according to our general HTC architecture, as can be seen in Figure 3. Furthermore, we leverage the same pre-trained word embeddings as in the BoW classifier. We use the same hyperparameters as Kim [16] for the architecture: filter sizes of 3, 4, and 5 for the three convolutional blocks with 100 filters each and a dropout rate of $p_{do} = 0.5$. Since Kim does not mention the employed optimizer, we utilize the Adam optimizer.

Transformer Classifier

Our transformer classifier relies on the BERT model architecture in the *base* size, as formulated by Devlin et al. [6]. The transformer layer uses dropout with a dropout rate of 0.1, and is followed by a fully connected layer, which outputs the hidden text representation that can be used for text classification. The architecture is illustrated in Figure 3. We initialize the BERT layers with pre-trained weights from the HuggingFace Python library [40] and use the Adam optimizer for further training.

3.2 Utility Metrics

Our evaluation of the utility provided by the described model architectures after training is guided by a recent work on HTC by Stein et al. [35]. Accordingly, we use the following flat, hierarchical and lowest common ancestor (LCA) metrics: *accuracy*, *hierarchical F-measure* and the *LCA F-measure*.

We decided to report the (flat) accuracy Acc rather than precision (P), recall (R) or *F-measure*, since it is an intuitive metric also for non-experts in machine learning. Furthermore, we report *hierarchical F-measure* and *LCA F-measure*, denoted respectively by F_H and F_{LCA}, since they are considered the state of the art in the field of HTC. All these metrics allow comparison with related work and are similar to non-hierarchical evaluation measures. They are defined as follows:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}, \tag{2}$$

$$F_H = \frac{2 P_H R_H}{P_H + R_H}, \tag{3}$$

$$F_{LCA} = \frac{2 P_{LCA} R_{LCA}}{P_{LCA} + R_{LCA}}. \tag{4}$$

3.3 Privacy Metrics

We assume that the data owner is equally concerned about the adversary's ability to identify members and non-members, and quantify privacy by two scores that are addressing members and non-members. First, *membership*

Table 1: Hyperparameters and composed ϵper model and dataset. The hyperparameters were identified via Bayesian hyperparameter optimization.

		BestBuy			Reuters			DBPedia		
		BoW	CNN	Transformer	BoW	CNN	Transformer	BoW	CNN	Transformer
learning rate	Orig.	0.001	0.001	0.005	0.001	0.001	0.005	0.001	0.001	0.005
	DP	0.01	0.001	0.08	0.008	0.001	0.005	0.06	0.001	0.01
batch size	Orig.	32	32	32	32	32	32	32	32	32
	DP	64	64	64	64	64	32	64	64	32
clipping norm $\mathcal{C}$	DP	0.19	1.48	2.07	0.33	6.28	12.86	0.03	0.21	1.6
microbatch size	DP	1	1	1	1	1	4	1	1	4
ϵ	$z = 0.1$	30253	33731	5902	2048	1091	21597	4317	6414	24741
	$z = 0.5$	11.5	11.1	6.58	4.19	4.11	4.4	5.1	6.29	4.88
	$z = 1.0$	1.51	1.38	1.04	0.79	0.79	0.77	0.87	0.96	0.81
	$z = 3.0$	0.26	0.5	0.29	0.22	0.22	0.22	0.2	0.21	0.21
Training records		$41,625$			$651,585$			$240,942$		
Validation records		4626			$72,399$			$36,003$		
Test records		5139			$80,443$			$60,794$		

advantage [42] Adv as difference between the MI adversary's true and false positive rates, and secondly the Receiver Operating Curve area under curve (AUC) of the attack model.

Yeom et al. [42] present a theoretical upper bound for the membership advantage that can be derived a priori (i.e., before model training) from privacy parameter ϵ. However, a considerable gap between the theoretic upper bound for the membership advantage and the membership advantage of state-of-the-art inference attacks has been observed by Jayaraman et al. [14] for numeric and image data. We will also state the theoretical upper bound on membership advantage in our experiments to illustrate the gap between theoretic and practical membership advantage for textual data.

4 Evaluation

In this section we first describe the experimental setup and the datasets for the evaluation. Afterwards, we experimentally assess privacy and utility for the three previously formulated HTC models and three datasets[4]. Lastly, we provide an additional analysis of drivers for MI in HTC models.

In all experiments we assume that the data owner would also want converging target models even when training with DP. Thus, all HTC models leverage early stopping with a patience of 3 epochs to terminate the training process before overfitting. Furthermore, we set the DP parameter C (i.e., the sensitivity Δf) in our experiments to the median of the norms of the unclipped gradients over the course of original training as suggested by Abadi et al. [1]. For all executions of the experiment CDP noise is sampled from a Gaussian distribution (cf. Definition 1) with scale $\sigma =$ *noise multiplier* $z \times$ *clipping norm* $\mathcal{C}$. According to McMahan et al. [22], values of $z \approx 1$ will provide reasonable privacy guarantees. We evaluate increasing noise regimes per dataset by evaluating noise multipliers $z \in \{0.1, 0.5, 1.0, 3.0\}$ and calculate the resulting ϵ at a fixed $\delta = \frac{1}{n}$.

4.1 Experimental Setup

For our experiments we split the datasets into training, validation and test data. Training data is used to learn the model parameters (i.e., weights), validation data to check the goodness of training hyperparameters and test data is used to assess generalization and real-world performance. Before target and attack model training so called hyperparameters have to be set manually before training (e.g., learning rate, batch size). We used Bayesian hyperparameter optimization for all target model experiments to ensure that we found good hyperparameters that yield high accuracies on the respective models and data. Bayesian Optimization is more efficient that a grid search since it considers past trials during the hyperparameter search (i.e., maintains state). An overview of all hyperparameters, dataset size for training, validation and test, and the overall ϵ per training is provided in Table 1. For the attack model we reused original hyperparameters of Nasr et al. [27] which already performed well. The majority of experiments were conducted on EC2[5] GPU optimized instances of type "p3.8xlarge". In all experiments, we used the "Deep Learning AMI (Amazon Linux)" machine image, building on a Linux 4.14 kernel, Python 3.6 and TensorFlow 2.2.

[4] We will publish all code and experiment scripts in case of acceptance.

[5] `https://aws.amazon.com/ec2/`

Hierarchy Level	Dataset	Classes	Assigned products
Level L_1	BestBuy	19	$51,390$
	DBPedia	9	$337,739$
	Reuters	4	$804,427$
Level L_2	BestBuy	164	$50,837$
	DBPedia	70	$337,739$
	Reuters	55	$779,714$
Level L_3	BestBuy	612	$44,949$
	DBPedia	219	$337,739$
	Reuters	43	$406,961$
Level L_4	BestBuy	771	$26,138$
Level L_5	BestBuy	198	$5,640$
Level L_6	BestBuy	23	346
Level L_7	BestBuy	1	1

Table 2: Classes and assigned records per level per dataset.

4.2 Datasets

We consider three real-world datasets in our experiments: the BestBuy dataset which represents a consumer product hierarchical classification task, the Reuters dataset which contains news articles and the DBPedia dataset with Wikipedia excerpts. The datasets have varying text lengths (34 to 212 words), are differing in the number of overall data (51,000 to 800,000 records) and have also been used in related work on HTC without differential privacy and membership inference.

BestBuy

The BestBuy dataset[6] contains $51,646$ unique products, each consisting of categorical features (e.g., SKU, type, manufacturer), numerical features such as price, textual features (e.g., name, description) and URLs, that are composed of one or more of the aforementioned features. In our experiments, we concatenate the features “name”, “manufacturer” and “description” to a single string and ignore the other features for classification. This selection is based on empirically observed superior classification accuracy. On average, the resulting concatenated texts have a length of 34 words. Additionally, every product holds a special feature called “category” assigning the product to a *single*, *partial-depth* class label in the BestBuy product hierarchy. The BestBuy product hierarchy is a *tree* represented in a separate file and consists of seven levels, each with a different number of classes, as shown in Table 2. As can be seen, level L_4 has the most classes. Also, we can see that even on the first level, not all of the existing $51,646$ products are assigned to a class. Particularly, we found that 256 products (0.50%) are assigned to classes not contained in the BestBuy product hierarchy. We removed these products as the assigned classes did not fit into the product hierarchy (e.g., “Other Product Categories” or “In-Store Only”). Furthermore, not every product is assigned to a class on every level, meaning the most specific class of many products is on a lower level than L_7. In our experiments, we only make use of the first three hierarchy levels. We decided to do so due to the long tail characteristic of the dataset. Thus, the predictions of our classifiers are less specific than potentially possible, but more robust due to a higher number of training examples in comparison to fine grained training for all hierarchies. 10% of the overall data was used for testing.

Reuters

The “Reuters Corpus Volume 1” (RCV1) dataset[7] is an archive of over $800,000$ manually categorized news articles [19]. Per news article, a headline, text block and topic codes representing the classes in the hierarchy are provided. In our experiments we use the concatenation of headline and text block as input for the respective classifiers. The resulting texts have an average length of 237 words. Table 2 shows the number of classes and assigned documents for each hierarchy level of the Reuters dataset. To ensure comparability with state of the art we follow the approach of Stein et al. [35] and randomly assign 80443 texts to the test dataset and assign each news article to the least frequent topic code. This approach is based on the assumption that the least common topic code is the one that most specifically characterizes the document.

[6] `https://github.com/BestBuyAPIs/open-data-set`

[7] `https://trec.nist.gov/data/reuters/reuters.html`

DBPedia

DBPedia is a community project that extracts structured knowledge from Wikipedia and makes it freely available using linked data technologies [18]. The DBPedia dataset for HTC[8] is used as a reference dataset in many state-of-the-art publications [15, 41, 24] on text classification. Overall, the dataset contains the introductions of $337,739$ Wikipedia articles, of which $240,942$ are pre-assigned to the training dataset and $60,794$ to the test dataset. Per article, the dataset contains a description of on average 102 words and three one-class label per hierarchy level (L_1, L_2, L_3). Table 2 shows the number of classes on each level of the DBPedia dataset hierarchy and indicates that all texts are assigned to a class on all levels, which means that the labels are *full depth*.

4.3 Privacy and Utility

A theoretic comparison of the CNN, BoW and Transformer models with respect to their robustness towards noise that is introduced by DP is only insightful to a limited extend, since their architectures and pre-training paradigms vary. However, in general the bias-variance trade-off for neural networks allows us to formulate high-level expectations. Simple neural networks will likely be prone to high bias and thus underfit in comparison to larger neural networks. Thus, the BoW model will potentially perform poorer on test data than the CNN or Transformer architecture, even in the presence of pre-training [9]. In contrast, large neural networks will have high variance and thus require larger amounts of training data to generalize well. Thus, the Transformer model will likely perform poorer on small datasets. In general, the bias decreases and the variance increases with the neural network size [28]. In combination with DP, we expect high bias models such as the BoW to be less affected by the introduced noise.

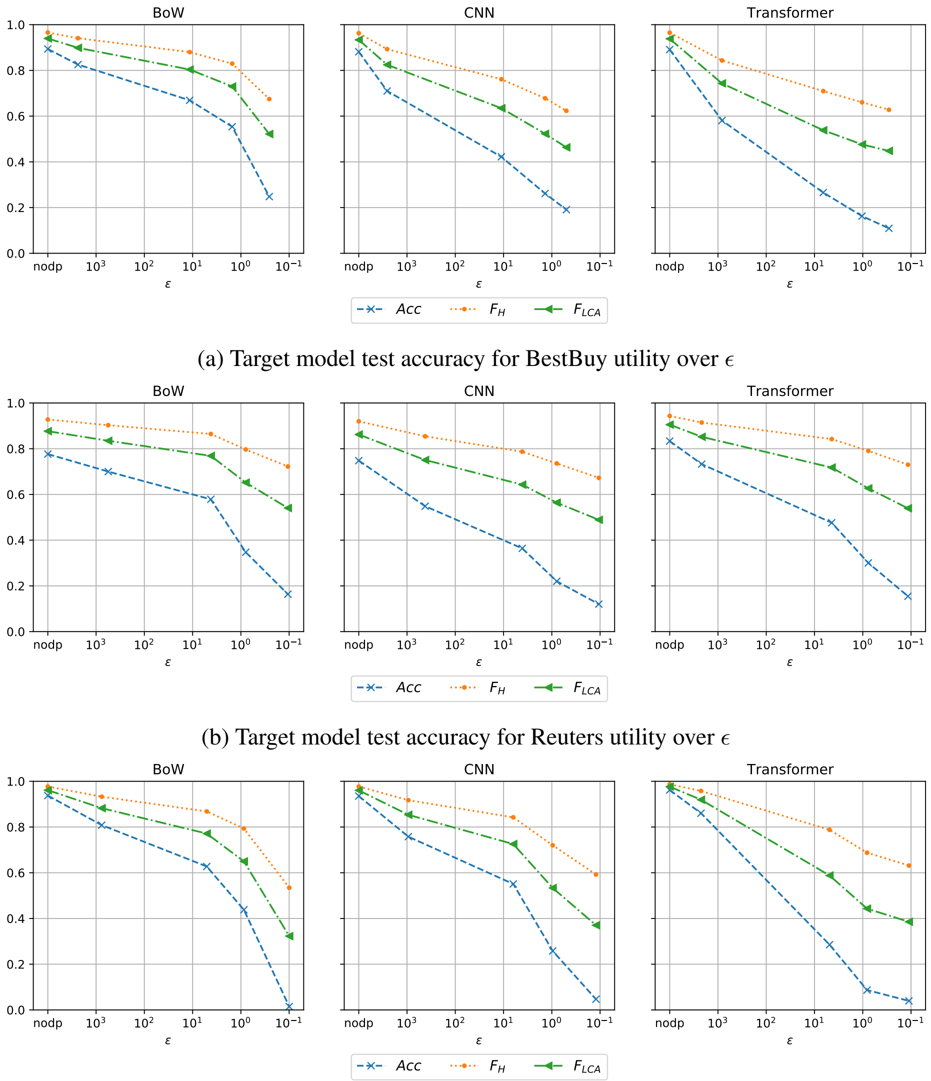
Figure 5 states the utility and Figure 6 the privacy scores over ϵ for the three datasets and model architectures. Furthermore, we additionally report the theoretical bound on Adv by Yeom et al. [42] to allow comparison of the theoretical and the empirical MI advantage. Notably, even if two classifiers were trained with the same noise multiplier z, they do not necessarily yield the same DP privacy parameter ϵ due to differing training epochs until convergence. All corresponding ϵ values per model and dataset were calculated for $\delta = \frac{1}{|\mathcal{D}_{target}^{train}|}$ per dataset and are stated in Table 1.

As expected, the model utility and adversary's success consistently decrease with stronger DP privacy parameters for all models and all datasets. Figure 5a shows that for BestBuy the BoW model's utility is most robust to the introduced noise, while the Transformer model's utility is most sensitive to the introduced noise. This observation becomes most evident when considering the flat accuracy Acc (*blue*), and is in line with our expectation for small datasets formulated at the beginning of this section. The hierarchical utility metrics F_H and F_{LCA} do not decrease as strong as Acc, since they also account for partially correct predictions. Interestingly, for BestBuy, the CNN model's MI metrics in Figure 6a already reach the baseline level at $\epsilon = 33,731$ ($z = 0.1$). The large ϵ points out that with respect to the upper bound a huge privacy loss is occurring (i.e., e^{ϵ}) and the advantage should also be maximal (i.e., $e^{\epsilon} - 1$ [42]). However, the empirical membership advantage lies far below this theoretical bound. In contrast to the CNN, the MI attack against the BoW and Transformer models is only reaching the baseline at $\epsilon = 1$ and $\epsilon = 1.5$, respectively.

The results for the Reuters dataset are provided in Figure 5b and 6b. Compared to BestBuy, the decrease in model utility on Reuters is smaller for all three HTC models, which can be explained with a significantly higher amount of training examples and a smaller amount of hierarchical classes. The BoW classifier's utility is most robust to the addition of noise to the training process, yet closely followed by the Transformer model. However, the CNN model exhibits the most severe decrease in model utility. Figure 6b indicate that the MI adversary's advantage drops to the baseline level again for very weak DP guarantees of $\epsilon > 10^2$ for all models. This behavior can be explained with the high amount of training examples and the smaller amount of hierarchical classes. Therefore, the gap between the empirically measured membership advantage and the theoretical upper bound on membership advantage diverge widely.

For DBPedia in Figure 5c, the BoW model is again most robust, and the Transformer model is least robust to the added noise during the training process, similar to the observations made on the BestBuy dataset. This is in line with our formulated expectations. The only exception are the measured utility metrics for $\epsilon \approx= 10^{-1}$, for which the BoW model performs worse than the CNN and Transformer model. MI metrics for the DBPedia HTC models are provided in Figure 6c. We see that the MI metrics for the BoW and Transformer models drop to the baseline level for very weak DP guarantees, similar to the Reuters models. Therefore, our MI adversary does by far not reach the theoretical upper Adv bound. Notably, the MI metrics for the CNN model do not drop to the baseline level for the considered range for z and resulting ϵ. Hence, the gap between the measured Adv and theoretical upper bound on Adv reaches its lowest value for this model.

[8] https://www.kaggle.com/danofer/dbpedia-classes



(a) Target model test accuracy for BestBuy utility over ϵ

(b) Target model test accuracy for Reuters utility over ϵ

(c) Target model test accuracy for DBPedia over ϵ

Figure 5: Target model test accuracy per dataset over ϵ.

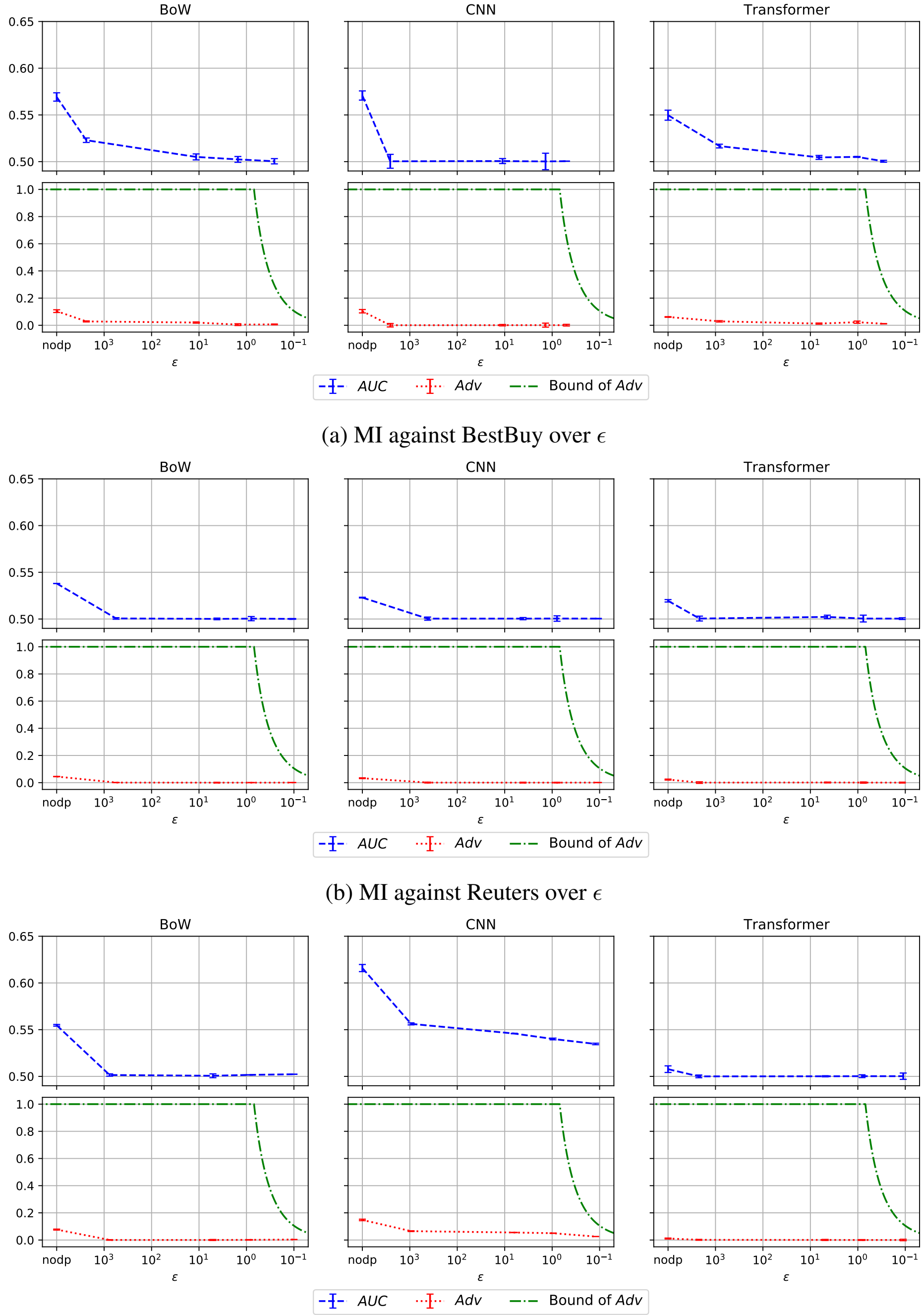
Overall, the privacy and utility results support our expectation that the utility of a high bias model such as BoW is less affected by the introduced noise than models with high variance such as Transformer. On the other hand, the Transformer model is less prone to the MI attack due to better generalisation capabilities through high variance.

4.4 Effect of Target Model Training Parameters

It is noticeable that the observed MI accuracies are rather low throughout all experiments. To validate whether this observation is due to our methodology which solely considers well generalizing and pre-trained models, we vary certain target model training parameters for the BestBuy Transformer classifier. Namely we consider the impact of excessive training, reducing the number of training examples and removing pre-trained weights of the target model.

Excessive Training

For this modification, the Transformer classifier is trained without early stopping for a fixed number of epochs, which is chosen significantly higher than the original number of epochs obtained with early stopping. In doing so, we deliberately force the model to overfit, i.e., adapt to the noise in $\mathcal{D}_{target}^{train}$ instead of the underlying distribution. Table 3 shows the metrics of the original BestBuy Transformer target model in the first column, which converged after 14



(a) MI against BestBuy over ϵ

(b) MI against Reuters over ϵ

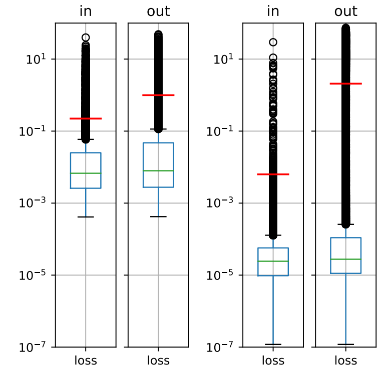
(c) MI against DBPedia over ϵ

Figure 6: MI AUC, Adv and Bound on MI Adv per dataset ϵ.

epochs. The second and third column reveal the metrics for the overfit models, which are trained for 50 and 100 epochs, respectively. As expected, the overfit models achieve a smaller training loss and a higher test loss. However, surprisingly, the achieved test accuracy does not drop compared to the original model, while the training accuracy on L_3 increase to over 99%. The corresponding attack model accuracies rise from 53.06% to 53.92% and 54.01% respectively. This insignificant change may appear counter-intuitive given the increased loss on $\mathcal{D}_{target}^{test}$. When analyzing the loss on $\mathcal{D}_{target}^{train}$ and $\mathcal{D}_{target}^{test}$, we observed that both distributions' median loss decreased similarly as depicted in Figure 7a and 7b. The reason for the high average loss on $\mathcal{D}_{target}^{test}$ lies in the high loss value of a few outliers.

		$n = 41,625$, 14 epochs	$n = 41,625$, 50 epochs	$n = 41,625$, 100 epochs	$n = 4000$ 30 epochs	$n = 400$ 30 epochs
$\mathcal{D}_{target}^{train}$	L_1	99.71%	99.94%	99.94%	99.94%	98.44%
	L_2	99.20%	99.86%	99.92%	99.44%	85.16%
	L_3	96.91%	99.74%	99.81%	96.07%	63.28%
	Loss	0.18	0.01	0.01	0.30	3.15
$\mathcal{D}_{target}^{test}$	L_1	97.24%	96.93%	97.06%	93.47%	84.31%
	L_2	95.00%	94.79%	94.73%	87.89%	59.23%
	L_3	89.32%	91.11%	91.35%	78.53%	37.76%
	Loss	0.89	1.60	1.87	2.08	3.97
L_3 Gap		7.60%	8.63%	8.47%	17.54%	25.52%
Loss Ratio		5.2	160	187	6.93	1.26
Acc_{MI}		53.06%	53.92%	54.01%	64.62%	75.00%

Table 3: Per-level accuracies and summarized loss for BestBuy without DP.



(a) Loss on original model after 14 epochs

(b) Loss on overfit model after 100 epochs

Figure 7: Loss distribution of members and non-members for BestBuy. Each boxplot is on a log scale depicting outliers (black), median (green) and mean (red) of the respective distribution.

Reduced number of training examples

In this variation, we reduce the size of $\mathcal{D}_{target}^{train}$, which originally contains $n = 41,625$ training examples. With this adaption, the hierarchical classifier should not be able to generalize as well as the original classifier due to two reasons. First, the training dataset is less representative of the problem domain, and second, underrepresented classes contain even fewer examples. Training the hierarchical classifier with only $n = 4,000$ training examples indeed leads to worse generalization with a maximum train-test gap of 17.54% on the third level as shown in Table 3. The trained attack model for this variation converged at 64.62% accuracy, which is a significant increase compared to the original target model. Further reducing the training data to $n = 400$ examples reduces the target model performance even more, with a maximum train-test gap of 25.52% on the third level, as evident from Table 3. For this target model with $n = 400$, we observe an attack accuracy of 75%. In conclusion, reducing the number of training examples thus results in a significant MI attack improvement.

Removing pre-trained weights

In this variation, the hierarchical classifier was trained from scratch, without initializing the Transformer weights from a pre-trained model. We hypothesize that this classifier variation might be more vulnerable to MI attacks, since a model without pre-training might tend to memorize more information about $\mathcal{D}_{target}^{train}$. Training the original classifier from scratch did not converge to a useful model, with only 18% accuracy on the first level, which can be explained by the relatively small amount of training data compared to the large corpora the Transformer model is usually pre-trained on. The issue can be solved by replacing the BERT-Base layers with BERT-Tiny layers, as they contain fewer weights to train. The hierarchical classifier trained from scratch yields a model with 75.17% flat Acc. The trained attack model

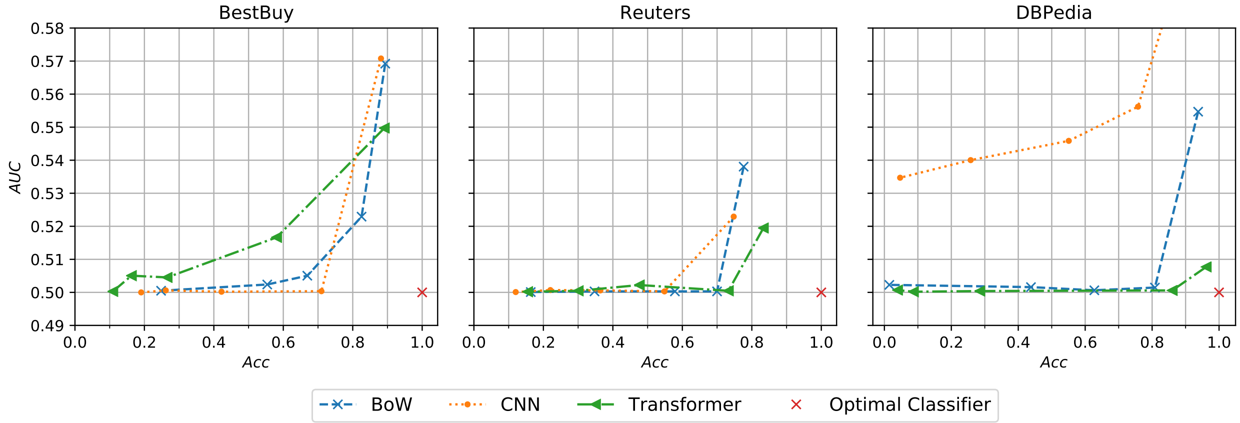


Figure 8: Privacy-utility trade-off per HTC model on each dataset. Privacy and utility are represented by MI AUC and classification Acc, respectively. An optimal classifier would exhibit 100% Acc and no vulnerability to the MI adversary, expressed by 50% AUC.

for this variation converged at 55.04% accuracy, which means that even without pre-training, the MI attack does not perform as well as in the work of Nasr et al. [27].

5 Discussion

Large values for privacy parameter ϵ are sufficient to completely mitigate the MI attacks with a moderate decrease in model utility. In our experiments, we enforced the HTC models to satisfy DP guarantees by clipping and perturbing the computed gradients during the training process. As expected, the experimental results confirmed that enforcing DP in this way reduces the effectiveness of the performed MI attacks but also harms model utility. Figure 8 summarizes the trade-off between classification accuracy and MI AUC for each dataset. As can be seen, for all examined datasets, it is possible to completely mitigate the MI attack while reducing classification Acc by $< 20\%$. For BestBuy, the CNN model yields the best model utility for $AUC = 50\%$. In contrast, for Reuters and DBPedia, the Transformer yields the best model utility for $AUC = 50\%$. This behavior may be explained by the simpler architecture of BoW (e.g., BestBuy) , which may be suited better for small datasets, in comparison to the sophisticated Transformer architecture, which demands more training data (e.g., Reuters, DBPedia).

Large HTC datasets are easier to protect against MI attacks. Our empirical evaluation indicates the trend that large datasets with relatively few categories are easier to protect against the MI adversary than small datasets with many categories. In particular, for Reuters the MI attack can be prevented already with very weak DP guarantees for all ANN architectures. These findings extend the findings of previous work for non textual datasets to HTC [34, 27].

Similar DP privacy parameters do not imply a similar MI attack effectiveness. The experimental results show that the empirical MI risk for similar DP guarantees varies within each dataset but also within each model architecture. Therefore, we can again summarize that the MI attack effectiveness depends on the chosen model architecture and the dataset. Unfortunately, the results do not point to a model architecture that is strictly better suited to mitigate the MI attack. However, we recommend using a model with relatively few parameters as the BoW model, since the BoW model's utility was least sensitive to added DP noise, as discussed before. This holds especially for smaller datasets, whereas for larger datasets the Transformer network yields a favorable trade-off.

The BoW model's utility was reduced least by the added DP noise. Across all datasets, we observed that the CNN and Transformer model's utility scores were impacted more heavily compared to the BoW model's utility for similar DP guarantees. On two of the three datasets, the Transformer model's utility is impacted even more severely than the CNN model's utility. This finding suggests that a higher number of weights in an ANN might correlate with a stronger impact of DP training on the ANN utility. Specifically, the number of ANN weights is lowest in the BoW model and highest in the Transformer model. This insight should be taken into account when a data scientist wants to train an ANN based on a given formal DP guarantee.

HTC ANN exhibit a big gap between empirical and theoretical MI risk. The obtained results support the conclusions by Jayaraman et al. [14], who find that there remains a big gap between what state-of-the-art MI attacks can infer and what is the maximum that can theoretically be inferred according to the bound presented by Yeom et al. [42].

During evaluation, we measured the membership advantage and compared it to the theoretical membership advantage bound, which can be calculated given the respective ϵ-DP guarantee. We showed that this conclusion also holds in the context of HTC.

6 Related work

This work is related to HTC, DP in NLP and MI attacks for evaluating the privacy of differentially private ML models. Therefore, in this section, we want to briefly introduce the most important publications in the respective research fields.

Stein et al. [35] analyze the performance of different hierarchical text classifiers on the Reuters (RCV1) dataset that we also use for the experiments in our work. The authors find that a fastText-based classifier works better than a CNN-based classifier initialized with the same embeddings. For evaluation, all possible types of metrics are used in the paper, namely flat, hierarchical, and LCA metrics. Interestingly, the authors do not consider any hierarchical text classifier based on Transformer blocks, even though such Transformer-based architectures pose the state-of-the-art for text classification.

Abadi et al. [1] formulate an implementation of the differentially private stochastic gradient descent, which uses the Gaussian mechanism to perturb gradient descent optimizers for ANN. As ANN are widely used in modern NLP and natural language data in many cases contain sensitive data, there are various publications regarding DP in NLP. While Vu et al. [38] learn differentially private word embeddings for user generated content, so that the resulting word embeddings can be shared safely, we focus on safe sharing of whole ML models. Other works use DP for author obfuscation in text classification [10, 39]. In contrast, our work addresses the privacy-utility trade-off for perturbation of the gradient descent optimizer.

Carlini et al. [5] successfully apply DP to prevent information leakage in a generative model, specifically an ANN generating text. They introduce the *exposure* metric to measure the risk of unintentionally memorizing rare or unique training-data sequences in generative models. Our work does not consider generative models, but solely classification models.

Empirical MI attacks against machine learning models such as the attack used in this work were first formulated by Shokri et al. [34] in the form of black-box membership inference. The authors compare MI attacks with model inversion attacks, which abuse access to an ML model to infer certain features of the training data. In contrast to model inversion attacks, MI attacks target a specific training example instead of targeting all training examples for a specific class. Therefore, the authors argue that successful MI attacks indicate unintended information leakage. Consequently, Misra [26] uses black-box membership inference attacks to assess the information leakage of generative models.

Nasr et al. [27] showed that white-box MI attacks, that take the target model's internal parameters into account, are more effective than black-box MI attacks. Additionally, the authors assume that the adversary owns a fraction of the data owner's sensitive data. This stronger assumption about the adversary's knowledge increases the overall strength of the MI attack compared with black-box MI attacks.

While Rahman et al. [33] analyze the effect of different values for ϵ on the effectiveness of only black-box MI attacks, Bernau et al. [4] take both black-box and white-box attacks into account. However, both publications mostly consider specifically crafted non-textual MI datasets. We consider real-world textual training data.

Yeom et al. [42] introduce membership advantage to measure for the success of a MI attack. Furthermore, they formulate a theoretical upper bound for the membership advantage that depends on the DP guarantees of the target model. However, there is a gap between the theoretic upper bound for the membership advantage and the membership advantage of state-of-the-art inference attacks, as has been shown by Jayaraman et al. on numeric and image data [14]. In our work, we investigate this gap in the context of ANN for HTC.

7 Conclusion

This work compared the privacy-utility trade-off in differentially private HTC under a white-box membership inference adversary. Based on our evaluation results for three HTC architectures and three real-world datasets we note that white-box MI attacks only pose a minor risk to HTC models. Furthermore, data owners should consider the Transformer-based HTC model in practice when striving for a model that is already very resistant to MI attacks, even without any DP guarantee. However, the privacy-accuracy trade-off for full mitigation of MI is differing widely for all considered models and datasets. Our results suggest that the Transformer model is also favorable for large datasets with long texts when using DP, while the CNN model is favorable for smaller datasets with shorter texts. This confirms and extends state-of-the-art insights for HTC without differential privacy. However, if hardware costs have to be kept low or the training examples should be protected with a strong formal DP guarantee, the fastText based BoW model is a good choice due to the high robustness against DP perturbation. Our experiments also confirmed a large gap between the achievable membership advantage of the MI white-box attack and the theoretical DP membership advantage bound for HTC datasets and models.

8 Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 825333 (MOSAICROWN).

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep Learning with Differential Privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, CCS, pages 308–318, 2016.

[2] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 13–24, 2013.

[3] R. Babbar, I. Partalas, E. Gaussier, and M.-R. Amini. On flat versus hierarchical classification in large-scale taxonomies. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 1824–1832, 2013.

[4] D. Bernau, P.-W. Grassal, J. Robl, and F. Kerschbaum. Assessing differentially private deep learning with membership inference, 2020. arXiv Preprint.

[5] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *Proceedings of the USENIX Security Symposium*, USENIX, pages 267–284, 2019.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics)*, NAACL, pages 4171–4186, 2019.

[7] C. Dwork. Differential privacy. In *Proceedings of the International Colloquium on Automata, Languages and Programming*, ICALP, pages 1–12, 2006.

[8] C. Dwork and A. Roth. *The Algorithmic Foundations of Differential Privacy*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2014.

[9] A. Ezen-Can. A comparison of lstm and bert for small corpus, 2020. arXiv Preprint.

[10] N. Fernandes, M. Dras, and A. McIver. Generalised Differential Privacy for Text Document Processing. In *Proceedings of the Confernece on Principles of Security and Trust*, POST, pages 123–148, 2019.

[11] P. Goyal, S. Pandey, and K. Jain. *Deep Learning for Natural Language Processing*. Apress, 2018.

[12] R. Hariri, E. Fredericks, and K. Bowers. Uncertainty in big data analytics: survey, opportunities, and challenges. *Journal of Big Data*, 6(44), 2019.

[13] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro. LOGAN: Membership Inference Attacks Against Generative Models. In *Proceedings on Privacy Enhancing Technologies*, PoPETs, Berlin, Germany, 2019. De Gruyter.

[14] B. Jayaraman and D. Evans. Evaluating differentially private machine learning in practice. In *Proceedings of the USENIX Security Symposium*, USENIX, pages 1895–1912, 2019.

[15] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of Tricks for Efficient Text Classification. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, EACL, pages 427–431, 2017.

[16] Y. Kim. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing)*, EMNLP, pages 1746–1751, 2014.

[17] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown. Text Classification Algorithms: A Survey. *Information*, 10(4), 2019.

[18] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[19] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[20] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*, chapter 16: Text Categorization. MIT Press, 1999.

[21] Y. Mao, J. Tian, J. Han, and X. Ren. Hierarchical text classification with reinforced label assignment. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 445–455, 2019.

[22] H. B. McMahan, G. Andrew, U. Erlingsson, S. Chien, I. Mironov, N. Papernot, and P. Kairouz. A General Approach to Adding Differential Privacy to Iterative Training Procedures. In *Privacy Preserving Machine Learning. Workshop during the Conference on Neural Information Processing Systems.*, NIPS, Mar. 2019.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations*, ICLR, 2013.

[24] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao. Deep Learning Based Text Classification: A Comprehensive Review, 2020. arXiv Preprint.

[25] I. Mironov. Renyi Differential Privacy. In *Proceedings of the Computer Security Foundations Symposium*, CSF, pages 263–275, 2017.

[26] V. Misra. Black Box Attacks on Transformer Language Models. In *Debugging Machine Learning Models. Workshop during the International Conference on Learning Representations.*, ICLR, 2019.

[27] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *2019 IEEE Symposium on Security and Privacy*, S&P, pages 739–753, 2019.

[28] B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas. A modern take on the bias-variance tradeoff in neural networks, 2019. arXiv Preprint.

[29] S. Peng, R. You, H. Wang, C. Zhai, H. Mamitsuka, and S. Zhu. Deepmesh: deep semantic representation for improving large-scale mesh indexing. *Bioinformatics*, 32(12):i70–i79, 2016.

[30] J. Pennington, R. Socher, and C. Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[31] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL, pages 2227–2237, 2018.

[32] B. Qu, G. Cong, C. Li, A. Sun, and H. Chen. An evaluation of classification models for question topic categorization. *Journal of the American Society for Information Science and Technology*, 63(5):889–903, 2012.

[33] A. Rahman, T. Rahman, R. Laganiere, N. Mohammed, and Y. Wang. Membership Inference Attack against Differentially Private Deep Learning Model. *Transactions on Data Privacy*, 11(1):61–79, 2018.

[34] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership Inference Attacks against Machine Learning Models. In *Proceedings of the IEEE Symposium on Security and Privacy*, S&P, pages 3–18, 2017.

[35] R. A. Stein, P. A. Jaques, and J. F. Valiati. An Analysis of Hierarchical Text Classification Using Word Embeddings. *Information Sciences*, 471:216–232, 2019.

[36] C. Taylor. What's the big deal with unstructured data?, Sept. 2013.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the International Conference on Neural Information Processing Systems*, NIPS, page 6000–6010, 2017.

[38] X.-S. Vu, S. N. Tran, and L. Jiang. dpUGC: Learn Differentially Private Representation for User Generated Contents. In *Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing, 2019.

[39] B. Weggenmann and F. Kerschbaum. SynTF: Synthetic and Differentially Private Term Frequency Vectors for Privacy-Preserving Text Mining. In *The International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, page 305–314, 2018.

[40] T. Wolf, L. Debut, V. Sanh, J. Chaumond, and C. Delangue. HuggingFace's Transformers: State-of-the-art Natural Language Processing, 2020. arXiv Preprint.

[41] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*, NIPS, pages 5754–5764, 2020.

[42] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *Proceedings of the Computer Security Foundations Symposium*, CSF, pages 268–282, 2018.

[43] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, ICLR, 2017.