Article

On the Privacy–Utility Trade-Off in Differentially Private Hierarchical Text Classification

Dominik Wunderlich [1], Daniel Bernau [1], Francesco Aldà [1], Javier Parra-Arnau [2,3,*] and Thorsten Strufe [2]

1   SAP SE, 76131 Karlsruhe, Germany
2   Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
3   Department of Network Engineering, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain
*   Correspondence: javier.parra@upc.edu

Abstract: Hierarchical text classification consists of classifying text documents into a hierarchy of classes and sub-classes. Although Artificial Neural Networks have proved useful to perform this task, unfortunately, they can leak training data information to adversaries due to training data memorization. Using differential privacy during model training can mitigate leakage attacks against trained models, enabling the models to be shared safely at the cost of reduced model accuracy. This work investigates the privacy–utility trade-off in hierarchical text classification with differential privacy guarantees, and it identifies neural network architectures that offer superior trade-offs. To this end, we use a white-box membership inference attack to empirically assess the information leakage of three widely used neural network architectures. We show that large differential privacy parameters already suffice to completely mitigate membership inference attacks, thus resulting only in a moderate decrease in model 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, convolutional neural networks are preferable.

Keywords: text classification; differential privacy; membership inference

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 due to the inherent ambiguity of natural languages [1]. However, the automated classification of unstructured data overcomes manual data labeling activities and thus is a key capability for organizing data at scale [2]. 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 [3–6]. HTC leverages machine learning to automatically organize documents into taxonomies, predicting multiple labels in a predefined label hierarchy.

After data owners have trained HTC models on their data, the models may be shared with data analysts such as contractors, customers, or even the general public. However, sharing a model may leak information about the training data [7–10]. Perturbation with differential privacy (DP) (for conciseness, throughout this work, we use the acronym DP to refer to both "differential privacy" and its adjective form "differentially private") limits information leakage by anonymizing the training data or model training function [11–13]. DP introduces an inherent trade-off between privacy and utility, which means that a stronger privacy guarantee implies a decrease in informative value. Balancing this trade-off is especially hard when training Artificial Neural Networks (ANNs). On the one hand, ANN utility can only be assessed empirically after training and even small perturbations can have a high impact on utility [14]. On the other hand, DP anonymization parameter ϵ formulates a theoretic upper bound on information leakage that does not reflect the empirical information leakage for a concrete dataset. For ANNs, empirical information leakage can be assessed with Membership Inference (MI) attacks [7], which aim at identifying

single instances of the training data by sole access to the trained model [8,15]. A rather large gap has been observed between the high theoretical bound on information leakage under MI attacks that can be derived from DP guarantees and the empirical information leakage posed by MI attacks [16]. Consequently, choosing the anonymization strength via privacy parameter ε remains a challenging problem, since a data owner can either choose to consider the theoretical or empirical information leakage.

Our work compares the empirical privacy–utility trade-off of multiple HTC models by quantifying privacy under MI. We hypothesize that there are preferable ANN architectures with respect to the privacy–accuracy trade-off when applying DP to HTC. Unlike previous studies on the privacy–accuracy trade-off for numerical or image data [15–17], our work focuses on textual data which requires different ANN architectures, such as Transformers [18]. The main contributions of this paper are:

- Empirically quantifying and 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 (CNNs) and Transformer-based architectures.
- Connecting DP privacy guarantees to MI attack performance for HTC ANNs. In contrast to the adversary considered by DP, the MI adversary represents an ML specific threat model.
- Recommending HTC model architectures and privacy parameters for the practitioner based on the privacy–utility trade-off under DP and MI.

This paper is structured as follows. Section 2 recalls key aspects of DP, MI attacks and HTC. Section 4 formulates our approach for modeling the privacy–utility trade-off in HTC. Section 5 introduces reference datasets and Section 6 presents the experiment setup. Experiment results results are presented in Section 7 and subsequently discussed in Section 8. Section 3 introduces related work. Conclusions are drawn in Section 9.

2. Preliminaries

In the following, we provide preliminaries on DP in Section 2.1, MI in Section 2.2, HTC in Section 2.3 and lastly HTC specific machine learning concepts in Section 2.4. Throughout this paper, we will use the abbreviations and variables denoted in the list of acronyms in back matter.

2.1. Differential Privacy

In DP [11], a statistical aggregation function $f(\cdot)$ is evaluated over a dataset $\mathcal{D}$, and the result is perturbed before being provided to the data analyst. By means of perturbation, DP prevents an adversary with arbitrary auxiliary knowledge on all but one participant in a dataset $\mathcal{D}$ 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. The perturbation strength is steered by setting privacy parameters (ϵ, δ), and small privacy parameters will result in high perturbation. A formal definition of DP is provided in Definition 1.

Definition 1 ((ϵ, δ)-DP [19])**.** *A randomized mechanism $\mathcal{M}$ on a query function f satisfies (ϵ, δ)-DP for $\delta > 0$ if, for all pairs of neighboring databases $\mathcal{D}, \mathcal{D}'$ and for all outputs $\mathcal{O} \subseteq \mathrm{range}(\mathcal{M})$,*

$$\Pr[\mathcal{M}(\mathcal{D}) \in \mathcal{O}] \leq e^\epsilon \Pr[\mathcal{M}(\mathcal{D}') \in \mathcal{O}] + \delta. \tag{1}$$

DP is enforced by mechanisms. Mechanisms for numerical data perturb the original query value $f(\mathcal{D})$ by adding numerical noise. DP mechanisms need to add noise scaled to the *global sensitivity*. Global sensitivity is formally defined in Definition 2.

Definition 2 (Global ℓ_1-sensitivity [11]). *Let $\mathcal{D}$ and $\mathcal{D}'$ be neighboring databases. The global ℓ_1-sensitivity of a function f, denoted by Δf, is defined as*

$$\Delta f = \max_{\forall \mathcal{D}, \mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_1.$$

In this work, we use the Gaussian mechanism for gradient perturbation to perturb the Adam optimizer for ANN training (the Tensorflow privacy package was used throughout this work: https://github.com/tensorflow/privacy) as suggested by Abadi et al. [12]. For simplicity, we shall refer to the perturbed Adam optimizer as DP-Adam. A DP optimizer for ANN training, such as DP-Adam, uses a randomized mechanism $\mathcal{M}_{nn}$. The optimizer updates the weight coefficients θ_t of an ANN per training step $t \in \{1, \ldots, T\}$ 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. α is a scaling function on $\tilde{g}$ to compute an update (i.e., learning rate). After T steps, DP-Adam outputs a DP weight matrix θ that is used by the ANN prediction function. In case of DP-Adam, $\mathcal{M}_{nn}$ is a Gaussian mechanism as specified in Theorem 1.

Theorem 1 (Gaussian Mechanism [19]). *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 the Normal distribution $\mathcal{N}(0, \sigma^2)$.*

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 use Rényi DP as suggested by Mironov [20] to calculate the tight, overall privacy guarantee ϵ under composition. $(\alpha, \epsilon_{RDP})-$Rényi DP (RDP), with $\alpha > 1$, quantifies the difference in distributions $\mathcal{M}(\mathcal{D}), \mathcal{M}(\mathcal{D}')$ by their Rényi divergence [21]. For a sequence of T mechanism executions each providing $(\alpha, \epsilon_{RDP,i})$-RDP, the privacy guarantee composes to $(\alpha, \sum_i \epsilon_{RDP,i})$-RDP. The (α, ϵ_{RDP})-RDP guarantee converts to $(\epsilon_{RDP} - \frac{\ln \delta}{\alpha-1}, \delta)$-DP. The Gaussian mechanism is calibrated to RDP by:

$$\epsilon_{RDP} = \alpha \cdot \Delta f^2 / 2\sigma^2 \qquad (2)$$

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. Throughout this paper, we refer to the trained machine learning model as a *target model* and the data owner's training data as $\mathcal{D}_{target}^{train}$. We solely consider ANNs as target models in this paper. ANNs are structured in layers of neurons that are connected by weights. We denote the weights between a layer l and its preceding layer $l-1$ as $w^{(l)}$. The output of the l-th layer is denoted as $o^{(l)}$. The ANN's final output is the output of the last layer.

This paper builds upon the white-box MI attack against ANNs proposed by Nasr et al. [8]. Essentially, the white-box MI attack assumes an honest-but-curious adversary with access to the target model weight matrix $w^{(l)}$. The white-box MI adversary leverages this knowledge to calculate attack features for each record (x, y) in the form of layer outputs $o^{(l)}(x; W)$, losses $L(o(x; w), y)$, and gradients $\frac{\partial L}{\partial w^{(l)}}$.

With the aforementioned data, the white-box MI adversary trains a binary classifier, the *attack model*. The attack model allows one to classify records into members and non-members with respect to the target model and training dataset. The adversary is assumed to know a portion of the training and test data $\mathcal{D}_{target}^{train}$ and $\mathcal{D}_{target}^{test}$, and it generates features for training the attack model by passing the known records repeatedly through the trained target model. Nasr et al. [8] assumed the portion of known records at 50%, and we follow this assumption to allow comparison. The performance metrics of an MI attack model are 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 data preparation for the white-box MI attack and its evaluation is shown in Figure 1.
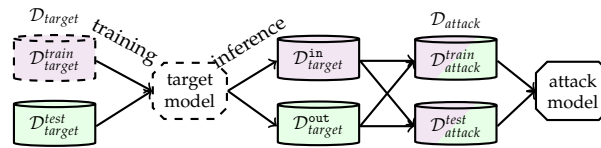
Figure 1. White-box MI with attack features. DP perturbation is applied on the target model training (dashed). The data that were used by the data owner during target-model training is colored: training (violet) and validation (green).

2.3. Hierarchical Text Classification

The task of *text classification* consists of categorizing texts into a predefined set of categories that do not have any hierarchical structure. Text classification is crucial in NLP [22], which is a subfield of linguistics, computer science, and artificial intelligence that is concerned with how computers can be programmed to process and analyze natural language data.

On the other hand, HTC addresses the task of classifying a text document into a *hierarchy of classes and sub-classes*. Text classification can therefore be regarded as a special case of HTC with only one hierarchy level and no sub-categories. Hierarchical classification problems can be categorized on the basis of their *hierarchy structure*, *label type* and *label depth*. In this work, we shall consider *tree* hierarchies with *single partial-depth* labels, which means that every text shall be assigned to a single label that can be any node in a tree hierarchy.

In many circumstances, hierarchical classification may be desirable as categorizing documents into varying levels of abstraction better fits the nature of certain applications, e.g., product categorization or support-ticket classification. In addition, as it has been consistently shown by numerous cognitive studies [23], people tend to favor categorization at different levels of abstraction.

Formally, HTC comprises a collection of text documents $x_1, \ldots, x_j \in \mathbb{X}$, where $\mathbb{X}$ is a document space; and a fixed set of classes $\mathbb{Y} = \{y_1, y_2, \ldots, y_k\}$ belonging to some hierarchy. Given a training set of labeled documents $(x_1, y_1), \ldots, (x_n, y_n)$ on a hierarchy, where $(x_i, y_i) \in \mathbb{X} \times \mathbb{Y}$, we wish to learn a classifier or classification function γ that maps documents to classes,

$$\gamma : \mathbb{X} \to \mathbb{Y},$$

so that each text document is only assigned to a single label, and a certain utility metric (see Section 4.2) is maximized.

2.4. Embeddings

ANNs employ *embeddings* in 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 [24]. Although embeddings have the ability to map semantically similar tokens to the same region in the vector space, a common shortcoming is that a word is always assigned to the same vector, ignoring previous and subsequent words (e.g., see Word2Vec [25,26]).

Word embeddings enable transfer learning, meaning that they can be trained on large unlabeled text corpora and afterwards be used in NLP systems for various tasks. In the case of ANNs, this is achieved by pre-initializing the first layer of an ANN with the pre-trained word embeddings.

3. Related Work

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

Stein et al. [27] 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 Transformer-based HTC model, even though Transformer architectures represent state-of-the-art for text classification.

Abadi et al. [12] formulate an implementation of the DP stochastic gradient descent, which uses the Gaussian mechanism to perturb gradient descent optimizers for ANNs. As ANNs 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. [28] learn DP word embeddings for user-generated content, so that the resulting word embeddings can be shared safely, we focus on the safe sharing of whole ML models. Other works use DP for author obfuscation in text classification [29,30]. In contrast, our work addresses the privacy–utility trade-off for perturbation of the gradient descent optimizer.

Carlini et al. [10] 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. [7] in the form of black-box MI. 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. Misra [31] uses black-box MI attacks to assess the information leakage of generative models.

Nasr et al. [8] 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. [15] analyze the effect of different values for ϵ on the effectiveness of only black-box MI attacks, Bernau et al. [17] 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. [32] introduce a membership advantage to measure the success of an MI attack. Furthermore, they formulate a theoretical upper bound for the membership advantage that depends on the DP guarantees of the target model. Humphries et al. [33] derive a bound for membership advantage that is tighter than the bound derived by Yeom et al. by analyzing the impact of giving up the i.i.d. assumption. However, there is a gap between the theoretic upper bound for the membership advantage and the membership advantage of state-of-the-art MI attacks, as has been shown by Jayaraman et al. on numeric and image data [16]. In our work, we investigate this gap in the context of ANNs for HTC.

4. Quantifying Utility and Privacy in HTC

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

4.1. HTC Model Architectures

Architectures for HTC include training a single classifier predicting classes in the flattened hierarchy (*flat*), training multiple classifiers predicting classes for a given level or node (*local*) and training a single classifier that respects the class hierarchy (*global*).

We chose a global HTC approach that features a single ANN with one output layer per level L_n in the tree hierarchy of the given HTC task. Each output layer is a fully connected

layer with a *softmax* activation which uses the output of the last hidden layer to predict the class for the input text on L_n. The architecture thus consists of only one model and does not ignore the class hierarchy. This yields two main advantages. First, our HTC classifier exhibits a reduced training time and therefore also a lower overall privacy cost in comparison to local HTC approaches. Secondly, the data analyst can still retrieve the prediction probabilities per level in contrast to flat HTC approaches that only provide prediction probabilities of the individual nodes [34].

Our HTC approach, however, has the disadvantage that a post-processing step is needed to obtain predictions consistent with the hierarchy, since each output layer makes predictions independently for its hierarchy level. We show an example in Figure 2a where the prediction for L_2 does not coincide with the prediction for L_1 and L_3, resulting in an undefined assignment. We suggest to resolve such inconsistencies by multiplying the softmax probabilities along the path from the root to each possible node. After comparing the multiplied probabilities, we output the path with the highest probability as prediction, leading to only 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. BoW models represent ANNs that ignore the word order within a text. BoW models comprise a single embedding layer in which the word vectors are added or averaged, and which is followed by one or more feed-forward layers with a softmax activation in the last layer. The BoW classifier used in this paper is based on the architecture of *fastText* [35], which achieves high accuracy and is computationally efficient due to the usage of a single feed-forward layer for classification. Each token is first embedded and the mean of all embeddings before the output layer is computed afterwards. Prior to training, the embedding layer is initialized with the widely used GloVe (v.1.2) embeddings (https://nlp.stanford.edu/projects/glove/) that are pre-trained on the "Wikipedia 2014" and "Gigaword 5" corpora [26,36]. For training, we use the Adam optimizer. The classifier architecture is visualized in Figure 3a.

Convolutional Neural Networks. A CNN contains one or more convolutional layers that convolve the input with a filter of a given width to extract and detect patterns. Although CNNs are widely used for detecting patterns in images, CNNs have also been effectively used for detecting patterns in a text [37,38] and do not ignore the word order. In this paper, we use the original CNN classifier for text classification proposed by Kim [39]. Their architecture first applies three convolutional layers to the embeddings, which are then concatenated and passed through a dropout layer to foster generalization. The architecture of the resulting HTC model is provided in Figure 3a. For training, we use the Adam optimizer and Glove embeddings. The training hyperparameters are taken from Kim [39]: filter sizes of 3, 4, and 5 for the three convolutional blocks with 100 filters each and a dropout probability of $p_{do} = 0.5$.

Transformer Networks. Transformer layers are ANN layers that are especially suited for processing longer texts due to a mechanism called *self-attention*. Due to self-attention, a single Transformer layer can relate all tokens of a text to each other [18]. In contrast, CNNs require multiple convolutional layers to relate the information between two arbitrary tokens in a text. The transformer classifier we use in this paper is the BERT model as formulated by Devlin et al. [40]. BERT comprises twelve transformer layers, each consisting of two sub-layers. To lower the computational effort that is needed for training BERT, we follow related work and initialize the BERT layers with pre-trained weights [40,41]. During training, we employ the Adam optimizer and a dropout probability of $p_{do} = 0.1$. The BERT HTC architecture is illustrated in Figure 3a.

In our methodology, we therefore do not consider recurrent neural networks (RNNs) such as long short-term memory (LSTM). The reason is that they can achieve comparable results to CNNs [39], while the training process can be more difficult due to tuning the

hyperparameters to avoid vanishing gradients. In addition, training RNNs is less efficient due to the lack of parallelization capabilities.
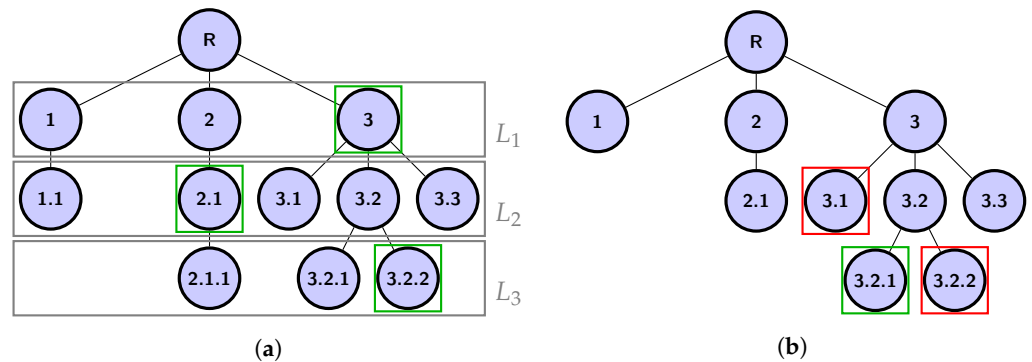
Figure 2. Visualization for obtaining and evaluating HTC predictions. (**a**) HTC predictions before postprocessing (*green*); (**b**) Two incorrect predictions (*red*) for a fictional input belonging to another category (*green*).
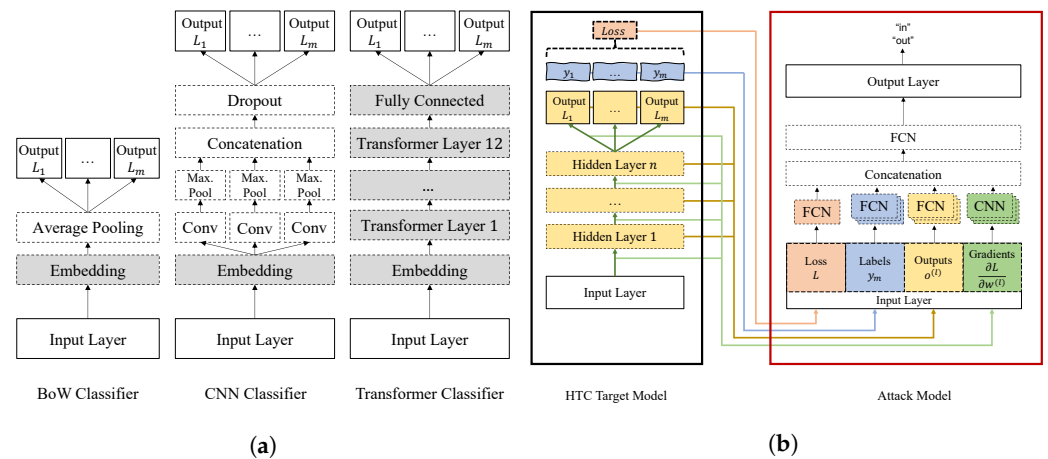
Figure 3. Architecture of target models and attack model. (**a**) Architecture of the HTC models used in the experiments. Pre-trained layers are marked *gray*; (**b**) Attack model architecture and observed attack features from an HTC target model.

4.2. Utility Metrics

There are two approaches for evaluating the utility provided of the described model architectures, namely, flat and hierarchical evaluation metrics [42,43]. To illustrate the difference between flat and hierarchical classification metrics, consider the tree hierarchy depicted in Figure 2b. Assume that the true category for a given test example x is 3.2.1 (*green*) and that two different classifiers output 3.2.2 and 3.1 as the predicted categories (*red*). When flat evaluation metrics are used, both systems are penalized equally since both predictions are counted as false negatives for the true category 3.2.1. However, the second classifier's error is more severe since its prediction is in an unrelated sub-tree of node 3, which is considered in hierarchical evaluation metrics.

In this work, we assess the utility of HTC based on a mix of flat and hierarchical metrics: *accuracy* and the hierarchical and lowest common ancestor (LCA) *F-measure*. We report the (flat) accuracy *Acc* due to its wide use in machine learning. We calculate the *hierarchical and LCA F-measure* since they are considered the state of the art in the field of HTC. The hierarchical F-measure F_H is calculated from *hierarchical precision P_H* and *hierarchical recall R_H*, and it is defined as follows [42]:

$$P_H = \frac{\sum_i |Anc_i \cap \hat{Anc_i}|}{\sum_i |\hat{Anc_i}|},$$

$$R_H = \frac{\sum_i |Anc_i \cap \hat{Anc_i}|}{\sum_i |Anc_i|},$$

$$F_H = \frac{2P_H R_H}{P_H + R_H}.$$

For a record i, Anc_i is the set consisting of the *true* class and all ancestors (except the root). Analogously, $\hat{Anc_i}$ is the set consisting of the *predicted* class and all ancestors (except the root). In Figure 2b, $Anc_i = \{3.2.1, 3.2, 3\}$ for the true class 3.2.1. $\hat{Anc_i}$ is $\{3.2.2, 3.2, 3\}$ and $\{3.1, 3\}$ with $R_H = \frac{2}{3}$ and $R_H = \frac{1}{3}$, respectively.

The LCA metrics $P_{LCA}, R_{LCA}, F_{LCA}$ are differing from the previous hierarchical metrics only by not considering the nodes above the lowest common ancestor in Anc_i, and thus voiding the overpenalization of errors for nodes with many ancestors [43]. Again, in the example shown in Figure 2b, for the prediction 3.2.2, we would have $LCA = 3.2$ with $Anc_i = \{3.2.1, 3.2\}$ and $\hat{Anc_i} = \{3.2.2, 3.2\}$ and therefore, $R_{LCA} = \frac{1}{2}$. For the prediction 3.1, we would have $LCA = 3$ with $Anc_i = \{3.2.1, 3.2, 3\}$ and $\hat{Anc_i} = \{3.1, 3\}$ and therefore $R_{LCA} = \frac{1}{3}$.

4.3. Privacy Metrics and Bounds

DP formulates a privacy bound on the ratio of probability distributions around D and D' resulting from a mechanism. The privacy bound holds for an adversary with auxiliary knowledge of up to all but one record in the dataset [44,45]. Yeom et al. [32] demonstrate that the privacy bound can be transformed into an upper bound on the membership advantage of an MI adversary. Membership advantage is calculated as follows from the True Positive Rate (TPR) and the False Positive Rate (FPR) [46]:

$$Adv = TPR - FPR.$$

The upper bound is:

$$Adv \le e^\epsilon - 1. \tag{3}$$

Whether the resulting membership advantage upper bound is reached, i.e., the empirically observed Adv matches the upper bound, depends on whether the sensitivity of the training data during model training matches the assumed global sensitivity (i.e., clipping norm $\mathcal{C}$; cf. Theorem 1) [47]. The gap between the lower and upper bound can be validated by implementing an MI adversary [16,17].

Figure 3b visualizes the architecture of our implemented MI adversary's attack model, which is based on the attack model of Nasr et al. [8]. We mainly extended their attack model to accept multiple labels, one per hierarchy level. The remaining components are unchanged. The attack model itself is represented by an ANN that learns to discriminate between training data and test data based on the attack features (e.g., losses).

In addition to Adv, we also quantify the area under the Receiver Operating Characteristic Curve (AUC) of the attack model. The AUC is also providing insights on the MI attack performance with respect to addressing members and non-members. The AUC is a general performance metric for evaluation of binary classifiers such as the MI attack model [46].

5. Reference Datasets

We consider three real-world datasets: 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), differ in the number of overall data (51,000 to 800,000 records) and have also been used in related work on HTC without DP and MI.

BestBuy. The BestBuy dataset (https://github.com/BestBuyAPIs/open-data-set, (accessed on 2 June 2022)) 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. Training a differentially private HTC model on datasets such as BestBuy can therefore prevent leaking sensitive information about individual products of a company.

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* and consists of seven levels, each with a different number of classes, as shown in Table 1. As can be seen, level L_4 has the most classes. In addition, 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 given 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 they are more robust due to a higher number of training examples in comparison to fine-grained training for all hierarchies. Overall, 10% of the data was used for testing.

Table 1. Classes and assigned records per level per dataset.

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	5640
Level L_6	BestBuy	23	346
Level L_7	BestBuy	1	1

All datasets have been *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 into an integer sequence since ANNs 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 into the exact same integer. The size of the vocabulary then represents the number of unique tokens in the text.

Reuters Corpus Volume 1. The "Reuters Corpus Volume 1" (RCV1) dataset (https://trec.nist.gov/data/reuters/reuters.html, (accessed on 2 June 2022)) is an archive of over

800,000 manually categorized news articles [48]. 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 1 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. [27] and randomly assign 80,443 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. A differentially private HTC model trained on a non-public article dataset such as RCV1 can prevent leaking sensitive information represented by individual articles.

DBPedia. DBPedia is a community project that extracts structured knowledge from Wikipedia and makes it freely available using linked data technologies [49]. The DBPedia dataset for HTC (https://www.kaggle.com/danofer/dbpedia-classes, (accessed on 2 June 2022)) is used as a reference dataset in many state-of-the-art publications [35,50,51] 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 1 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* . Training a differentially private HTC model on a dataset such as DBPedia prevents leaking information about participating institutions and people if the underlying encyclopedia is not public.

6. Experimental Setup

For our experiments, we split the datasets into training, validation and test data. Training data are used to learn the model parameters (i.e., weights), validation data to check the goodness of training hyperparameters, and test data are 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 than grid search, since it considers past trials during the hyperparameter search. An overview of all hyperparameters, dataset size for training, validation and test, and the overall ϵ per training is provided in Table 2. For the attack model, we reused the original hyperparameters of Nasr et al. [8] which already performed well. The majority of experiments were conducted on EC2 (https://aws.amazon.com/ec2/, (accessed on 2 June 2022)) GPU optimized instances of type "p3.8xlarge" with the "Deep Learning AMI" machine image, building on a Linux 4.14 kernel, Python 3.6 and TensorFlow 2.2.

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 *three* 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. [12]. For all executions of the experiment, CDP noise is sampled from a Gaussian distribution (cf. Definition 1) with scale $\sigma = $ *noise multiplier z × clipping norm C*. According to McMahan et al. [52], 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 $\delta = \frac{1}{n}$.

Table 2. Hyperparameters and ϵ per model and dataset. The hyperparameters were set with Bayesian optimization.

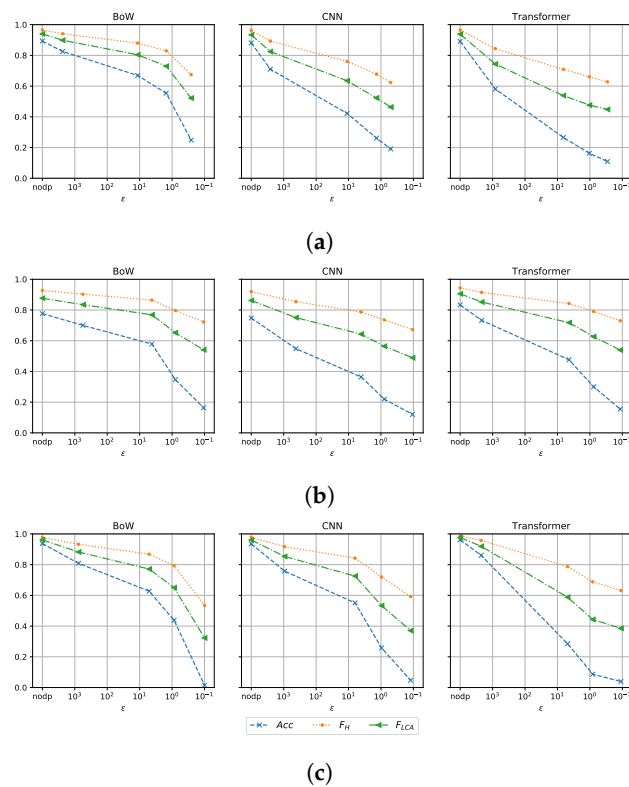| | | BestBuy | | | Reuters | | | DBPedia | | |
		BoW	CNN	Transformer	BoW	CNN	Transformer	BoW	CNN	Transformer
learning rate	Orig.	0.001	0.001	0.005	0.001	0.001	0.005	0.001	0.001	0.005
	DP	0.01	0.001	0.015	0.008	0.001	0.005	0.016	0.001	0.01
batch size	Orig.	32	32	32	32	32	32	32	32	32
	DP	64	64	64	64	64	32	64	64	32
$\mathcal{C}$	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	$z = 0.1$	30,253	33,731	5902	2048	1091	21,597	4317	6414	24,741
	$z = 0.5$	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		

7. Evaluation

In this section, we first describe the experimental setup. Afterwards, we experimentally assess privacy and utility for the previously formulated HTC models and datasets (we publish all code and experiment scripts at https://github.com/SAP-samples/security-research-dp-hierarchical-text). Lastly, we present several experiment variations to illustrate the impact of different parameters.

7.1. Empirical Privacy and Utility

A theoretic comparison of the CNN, BoW and Transformer models with respect to their robustness toward 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 ANNs allows us to formulate high-level expectations. Simple ANNs will likely be prone to high bias and thus underfit in comparison to larger ANNs. Thus, the BoW model will potentially perform poorer on test data than the CNN or Transformer architecture, even in the presence of pre-training [53]. In contrast, large ANNs 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 ANN size [54]. In combination with DP, we expect high bias models such as the BoW to be less affected by the introduced noise. Additionally, Transformer models may be negatively affected by gradient explosion when using relu activation functions in combination with DP [55].

Figure 4 states the utility and Figure 5 states 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. [32] to allow a 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 2.

Figure 4. Target model test accuracy per dataset over ϵ. (**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.

As expected, the model utility and adversary's success consistently decrease with stronger DP parameters for all models and all datasets. Figure 4a shows that for BestBuy, the BoW model's utility is the 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 it 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 strongly as *Acc*, since they also account for partially correct predictions. Interestingly, for BestBuy, the CNN model's MI metrics in Figure 5a 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., $e^\epsilon - 1$ [32]). 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 Figures 4b and 5b. 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 it is closely followed by the Transformer model. However, the CNN model exhibits the most severe decrease in model utility. Figure 5b indicates 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 upper bound on membership advantage diverges widely.
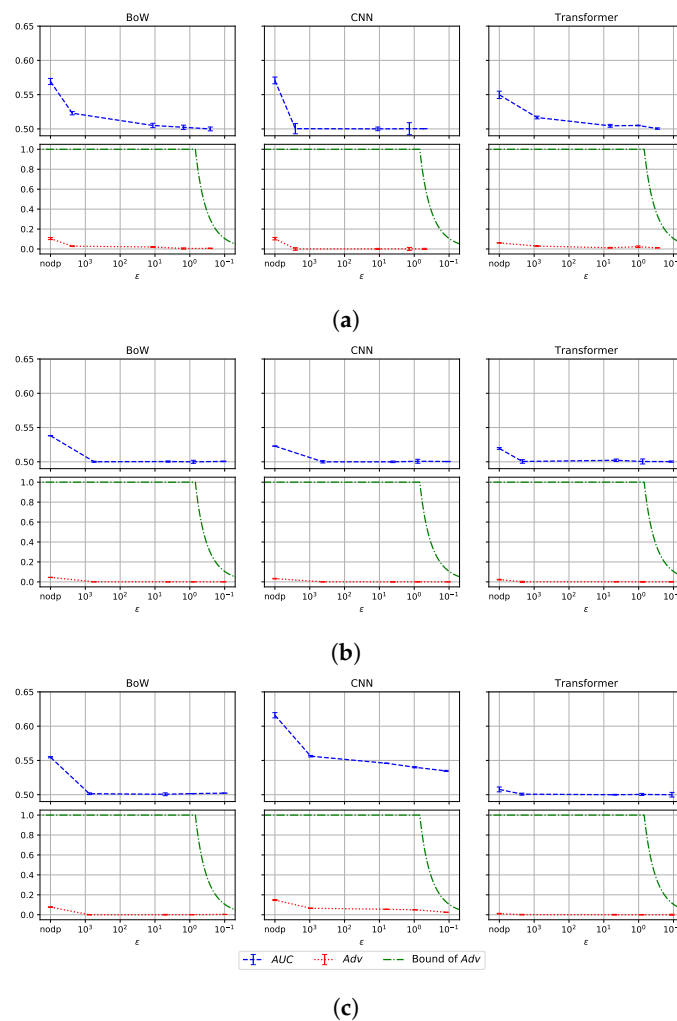
Figure 5. MI AUC, *Adv* and Bound on MI *Adv* per dataset ϵ. (**a**) MI against BestBuy over ϵ; (**b**) MI against Reuters over ϵ; (**c**) MI against DBPedia over ϵ.

For DBPedia in Figure 4c, the BoW model is again the most robust, and the Transformer model is least robust to the added noise during the training process, which is similar to the observations made on the BestBuy dataset. This is in line with our formulated expectations. The only exceptions 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 5c. 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.

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, Transformer models are less prone to the MI attack due to better generalization, which has generally been demonstrated aside from MI in related work [18,40].

7.2. Drivers for Attack Performance

Our experiments show that state-of-the-art MI attacks are not very effective when run against the trained HTC models. This leads to the question of whether more HTC-specific attacks or less generalizing HTC target models would boost attack performance. In the following subsections, we first validate experimentally that the attack performance does

not increase when introducing HTC-specific attack features. Second, we present several means that reduce target model generalization, where we show that especially reducing the number of training examples increases the vulnerability to MI attacks.

7.2.1. HTC-Specific Attack Model Features

We hypothesize that adapting the MI attack to exploit the hierarchical relation of the classes leads to an increased MI attack performance. Our approach to adapt the MI attack to HTC models is to extract additional attack features from the target model. We choose to evaluate two additional features in the following.

The former is a Boolean feature that we derive by checking if the HTC model's prediction before applying the post-processing step is consistent with the hierarchy of the HTC task. The latter is a scalar feature that is derived by multiplying the probabilities of the node with the highest softmax score on each level. The intuition behind this feature is to calculate a value that states how confident the target model is with the predicted label, since the assigned probability is obviously higher when the model outputs only small probabilities for the other labels. We refer to this attack feature as *prediction confidence*.

We pass the features into the attack model with an additional FCN similar to the loss. We tested the effect of the additional features on all datasets and models. However, the results for this attack variant do not result in a significant change of the considered MI metrics (Figure A1a–c in the Appendix A).

7.2.2. Reduced Target Model Generalization

Next, we describe four approaches that we formulated to increase the attack model performance by reducing target model generalization in HTC. Each approach is first motivated and then evaluated based on experimental results.

First, we provoke an overfitted target model by training 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 few samples in $\mathcal{D}_{target}^{train}$ instead of approximating the underlying distribution. We evaluated this approach based on the BestBuy Transformer classifier. Table 3 shows the metrics of the original model in the first column, which converged after 14 epochs. The second and third column reveal the metrics for 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 increases 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 $\mathcal{D}_{target}^{test}$. However, when analyzing the loss distribution of $\mathcal{D}_{target}^{train}$ and $\mathcal{D}_{target}^{test}$, we observe that the median losses decrease similarly as depicted in Figure 6a,b. The reason for the high average loss on $\mathcal{D}_{target}^{test}$ is due to the high loss value of a few outliers. Therefore, the loss ratio is not a consistently good indicator for MI attack effectiveness in practice, and rather, the accuracy gap should be taken into account.

Second, we reduce the number of training examples in $\mathcal{D}_{target}^{train}$. With this adaption, the hierarchical classifier should not generalize as well as the original classifier due to two reasons. First, the training dataset is less representative of the problem domain, and second, under-represented classes contain even fewer examples. We again evaluate this approach based on the BestBuy transformer classifier, which originally contains $n = 41,625$ training examples. Training the classifier with only 10% of the 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 results in a significant MI attack improvement in comparison to changing the attack model features.

(**a**)　　　　　　　　　　　　　　　(**b**)
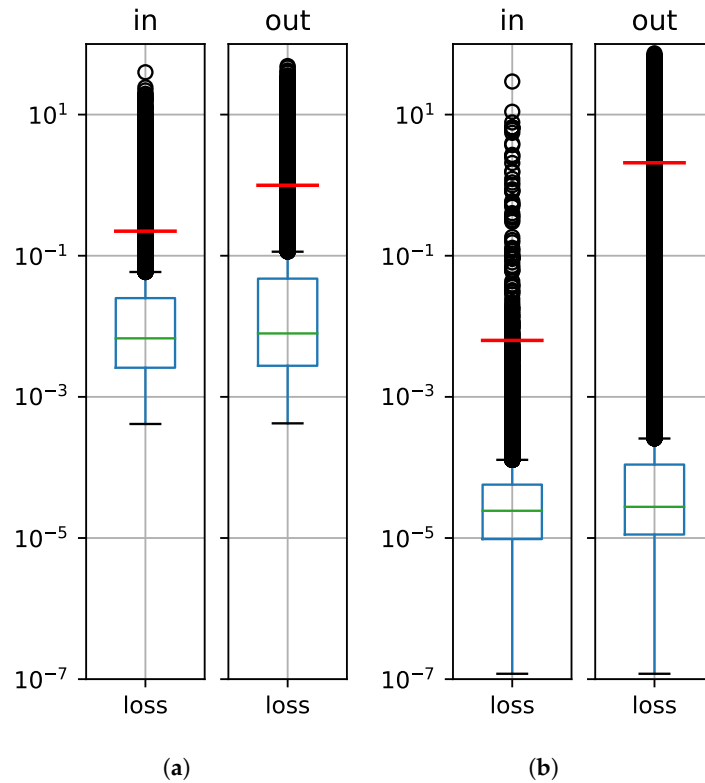
Figure 6. Loss distribution of members and non-members for BestBuy. Each boxplot is on a log scale depicting outliers (black), median (green) and mean (red) of the respective distribution. (**a**) Loss on original model after 14 epochs; (**b**) Loss on overfit model after 100 epochs.

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

		$n = 41{,}625$, 14 epochs	$n = 41{,}625$, 50 epochs	$n = 41{,}625$, 100 epochs	$n = 4000$, 30 epochs	$n = 400$, 30 epochs
	L_1	99.71%	99.94%	99.94%	99.94%	98.44%
$\mathcal{D}^{train}_{target}$	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
	L_1	97.24%	96.93%	97.06%	93.47%	84.31%
$\mathcal{D}^{test}_{target}$	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%

Third, we increase the number of hierarchy levels in the data, resulting in a more complex HTC task, which we again hypothesize to lead to worse generalization. Moreover, additional hierarchy levels lead to additional classification outputs and therefore additional attack features, which might further boost attack performance. Indeed, when increasing the number of levels from three to seven for the BestBuy BoW classifier, the corresponding MI attack accuracy rises from 56.15% to 57.07%. To ensure that this effect was caused by noise,

we also reduced the number of levels to one, resulting in an attack accuracy of 51.38%. This shows that an increased (decreased) number of hierarchy levels leads to higher (lower) attack performances.

Interestingly, when combining the effects of changing the number of hierarchy levels and decreasing the number of training examples, we made the observation that the ratio of the number of training examples to levels in the HTC task has a direct influence on the MI attack performance. We first noted this effect for BestBuy and then validated if we can also see it for the other datasets. The results confirm our expectations and are shown in Figure 7.
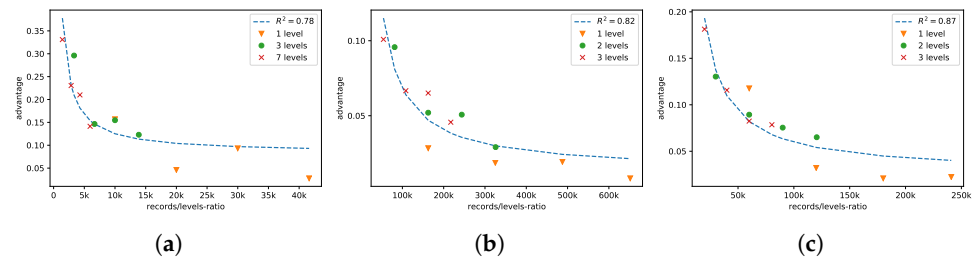

Figure 7. Attack model advantage over ratio of records per hierarchy for an overall number of $\{0.25, 0.5, 0.75, 1\} \times n$ training records. (**a**) Relation between the records/levels ratio and attack advantage for BestBuy; (**b**) Relation between the records/levels ratio and attack advantage for Reuters; (**c**) Relation between the records/levels ratio and attack advantage for DBPedia.

Finally, we train the hierarchical classifier from scratch, without leveraging pre-trained weights to initialize the 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 BestBuy Transformer classifier from scratch did not converge to a useful HTC model, with only 18% accuracy on the first level. This effect 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 overcome by replacing the BERT-Base layers with BERT-Tiny layers, since a tiny layer contains fewer weights to train. The hierarchical BERT-Tiny classifier trained from scratch yields a model with 75.17% flat *Acc*. The trained attack model for this variation converged at 55.04% accuracy, which is ≈2% higher than the attack on the original target model. This relatively small increase reveals that the use of pre-trained weights for the target models is not the reason for the relatively poor attack performance.

8. Discussion

Large values for privacy parameter ϵ are sufficient to completely mitigate MI attacks with 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 experiments showed 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 may be explained by the size and average text length of the datasets, since for small datasets with short texts (e.g., BestBuy), the CNN model is well suited, while for larger datasets with longer texts (e.g., Reuters, DBPedia), the Transformer architecture is suited better.
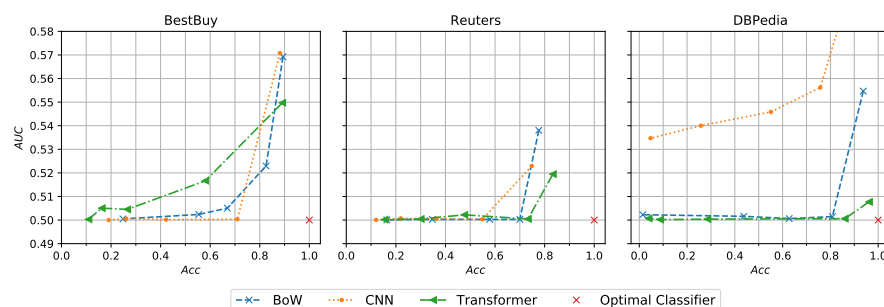
Figure 8. Privacy–utility trade-off per HTC model on each dataset. Privacy and utility are represented by MI *AUC* and classification *Acc*, respectively. An optimal classifier would exhibit 100% *Acc* and no vulnerability to the MI adversary, expressed by 50% *AUC*.

Overfitting is a key driver for MI attack performance. Our experiments support the understanding that overfitting leads to higher MI attack performance. We therefore extend the findings of previous work for non-textual datasets to HTC [7,8]. To prevent overfitting, our analysis of drivers for MI attack performance suggests to gather as many training examples as possible and only predict as many hierarchy levels as needed. Adding HTC-specific features to the attack model did not increase MI attack performance, confirming the validity of our results also for stronger adversaries.

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 such as the BoW model for smaller datasets, whereas for larger datasets, models with a high number of parameters such as the Transformer classifier yield 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 ANNs exhibit a big gap between empirical and theoretical MI risk. The obtained results support the conclusions by Jayaraman et al. [16], who found 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. [32]. During evaluation, we measured the membership advantage and compared it to the theoretical membership advantage bound, which can be calculated given the respective DP guarantee. We showed that this conclusion also holds in the context of HTC.

Lastly, we would like to note that the direct effect of the self-attention mechanism in the Transformer classifier on the privacy–accuracy trade-off could not be studied, since it is not possible to create a Transformer model without self-attention.

9. Conclusions

This work analyzed and compared the privacy–utility trade-off in DP HTC under a white-box MI adversary. Even without the use of DP, white-box MI attacks only posed a minor risk to the three HTC model architectures and reference dataset that we considered. In consequence, large privacy parameters were sufficient to fully mitigate the white-box

attack. We particularly observed Transformer-based HTC models to be rather resistant to MI attacks without using DP at all.

However, the privacy–accuracy trade-off for full mitigation of the white-box MI attack 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. However, if hardware costs shall be minimized or the training examples shall be protected with a strong formal DP guarantee (i.e., small ϵ value), the fastText-based BoW model is a good choice due to the high robustness against DP perturbation. Our experiments also confirm a large gap between the empirical membership advantage of the MI white-box attack and the theoretical DP membership advantage bound for HTC datasets and models.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

ANN	Artificial Neural Network
AUC	Area under the ROC Curve
BERT	Bidirectional Encoder Representations from Transformers
BoW	Bag of Words
CNN	Convolutional Neural Network
DAG	Directed Acyclic Graph
DP	Differential Privacy
FPR	False Positive Rate
HTC	Hierarchical Text Classification
LCA	Lowest Common Ancestor
LCL	Local Classifier per Level
LCPN	Local Classifier per Parent Node
LCN	Local Classifier per Node
MI	Membership Inference
NLP	Natural Language Processing

RCV1   Reuters Corpus Volume 1
RDP   Rényi Differential Privacy
RNN   Recurrent Neural Network
ROC   Receiver Operating Characteristic
TPR   True Positive Rate

Appendix A. Additional Figures

(a)

(b)

(c)
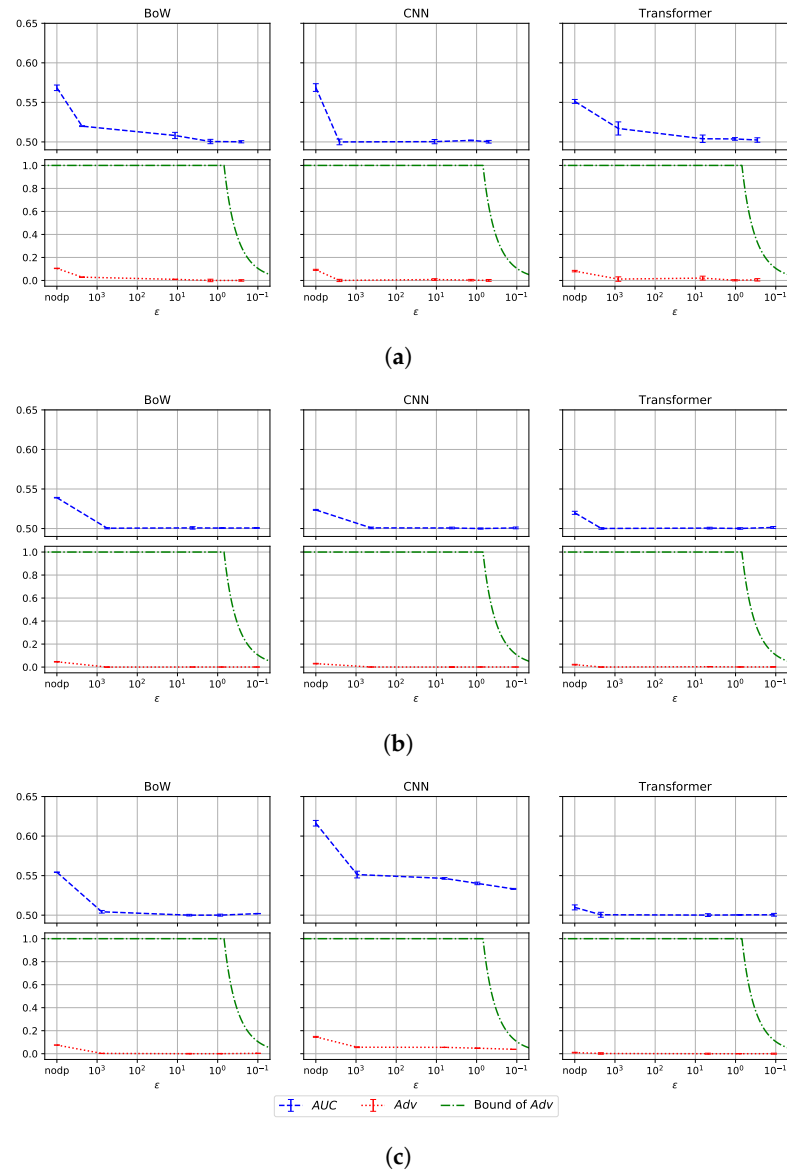
Figure A1. MI AUC, *Adv* and Bound on MI *Adv* per dataset with additional hierarchical attack features. (**a**) MI against BestBuy over ϵ; (**b**) MI against Reuters over ϵ; (**c**) MI against DBPedia over ϵ.

References

1. Hariri, R.; Fredericks, E.; Bowers, K. Uncertainty in big data analytics: Survey, opportunities, and challenges. *J. Big Data* **2019**, *6*, 44. [CrossRef]
2. Taylor, C. What's the Big Deal with Unstructured Data? 2013. Wired. Available online: https://www.wired.com/insights/2013/09/whats-the-big-deal-with-unstructured-data/ (accessed on 6 April 2022).
3. Mao, Y.; Tian, J.; Han, J.; Ren, X. Hierarchical Text Classification with Reinforced Label Assignment. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Hong Kong, China, 3–7 November 2019; pp. 445–455.
4. Qu, B.; Cong, G.; Li, C.; Sun, A.; Chen, H. An evaluation of classification models for question topic categorization. *J. Am. Soc. Inf. Sci. Technol.* **2012**, *63*, 889–903. [CrossRef]

5.   Agrawal, R.; Gupta, A.; Prabhu, Y.; Varma, M. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In Proceedings of the International Conference on World Wide Web, Rio de Janeiro, Brazil, 13–17 May 2013; pp. 13–24.

6.   Peng, S.; You, R.; Wang, H.; Zhai, C.; Mamitsuka, H.; Zhu, S. DeepMeSH: Deep semantic representation for improving large-scale MeSH indexing. *Bioinformatics* **2016**, *32*, i70–i79. [CrossRef] [PubMed]

7.   Shokri, R.; Stronati, M.; Song, C.; Shmatikov, V. Membership Inference Attacks against Machine Learning Models. In Proceedings of the IEEE Symposium on Security and Privacy, San Jose, CA, USA, 22–26 May 2017; pp. 3–18.

8.   Nasr, M.; Shokri, R.; Houmansadr, A. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-Box Inference Attacks against Centralized and Federated Learning. In Proceedings of the 2019 IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 19–23 May 2019; pp. 739–753.

9.   Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; Vinyals, O. Understanding Deep Learning Requires Rethinking Generalization. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.

10.   Carlini, N.; Liu, C.; Erlingsson, Ú.; Kos, J.; Song, D. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In Proceedings of the USENIX Security Symposium, Santa Clara, CA, USA, 14–16 August 2019; pp. 267–284.

11.   Dwork, C. Differential Privacy. In Proceedings of the International Colloquium on Automata, Languages and Programming, Venice, Italy, 10–14 July 2006; pp. 1–12.

12.   Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H.B.; Mironov, I.; Talwar, K.; Zhang, L. Deep Learning with Differential Privacy. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 308–318.

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

14.   Bagdasaryan, E.; Poursaeed, O.; Shmatikov, V. Differential Privacy Has Disparate Impact on Model Accuracy. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019.

15.   Rahman, A.; Rahman, T.; Laganiere, R.; Mohammed, N.; Wang, Y. Membership Inference Attack against Differentially Private Deep Learning Model. *Trans. Data Priv.* **2018**, *11*, 61–79.

16.   Jayaraman, B.; Evans, D. Evaluating Differentially Private Machine Learning in Practice. In Proceedings of the USENIX Security Symposium, Santa Clara, CA, USA, 14–16 August 2019; pp. 1895–1912.

17.   Bernau, D.; Grassal, P.W.; Robl, J.; Kerschbaum, F. Assessing Differentially Private Deep Learning with Membership Inference. *arXiv* **2020**, arXiv:1912.11328.

18.   Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All You Need. In Proceedings of the International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.

19.   Dwork, C.; Roth, A. The Algorithmic Foundations of Differential Privacy. In *Foundations and Trends in Theoretical Computer Science*; Now Publishers: Norwell, MA, USA, 2014.

20.   Mironov, I. Renyi Differential Privacy. In Proceedings of the Computer Security Foundations Symposium, Santa Barbara, CA, USA, 21–25 August 2017; pp. 263–275.

21.   van Erven, T.; Harremoës, P. Rényi Divergence and Majorization. In Proceedings of the Symposium on Information Theory, Austin, TX, USA, 13–18 June 2010; pp. 1335–1339.

22.   Manning, C.; Schütze, H. *Foundations of Statistical Natural Language Processing*; MIT Press: Cambridge, MA, USA, 1999; Chapter 16: Text Categorization.

23.   Murphy, G. *The Big Book of Concepts*; MIT Press: Cambridge, MA, USA, 2004.

24.   Goyal, P.; Pandey, S.; Jain, K. *Deep Learning for Natural Language Processing*; Apress: New York, NY, USA, 2018.

25.   Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. In Proceedings of the International Conference on Learning Representations, Scottsdale, AZ, USA, 2–4 May 2013.

26.   Pennington, J.; Socher, R.; Manning, C. Glove: Global Vectors for Word Representation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1532–1543.

27.   Stein, R.A.; Jaques, P.A.; Valiati, J.F. An Analysis of Hierarchical Text Classification Using Word Embeddings. *Inf. Sci.* **2019**, *471*, 216–232. [CrossRef]

28.   Vu, X.S.; Tran, S.N.; Jiang, L. dpUGC: Learn Differentially Private Representation for User Generated Contents. In Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing, La Rochelle, France, 9 April 2019.

29.   Fernandes, N.; Dras, M.; McIver, A. Generalised Differential Privacy for Text Document Processing. In Proceedings of the Confernece on Principles of Security and Trust, Prague, Czech Republic, 6–11 April 2019; pp. 123–148.

30.   Weggenmann, B.; Kerschbaum, F. SynTF: Synthetic and Differentially Private Term Frequency Vectors for Privacy-Preserving Text Mining. In Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 305–314.

31.   Misra, V. Black Box Attacks on Transformer Language Models. In Proceedings of the Debugging Machine Learning Models, Workshop during the International Conference on Learning Representations, New Orleans, LA, USA, 6 May 2019.

32.   Yeom, S.; Giacomelli, I.; Fredrikson, M.; Jha, S. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In Proceedings of the Computer Security Foundations Symposium, Oxford, UK, 9–12 July 2018; pp. 268–282.

33. Humphries, T.; Rafuse, M.; Tulloch, L.; Oya, S.; Goldberg, I.; Kerschbaum, F. Differentially Private Learning Does Not Bound Membership Inference. *arXiv* **2020**, arXiv:2010.12112.

34. Babbar, R.; Partalas, I.; Gaussier, E.; Amini, M.R. On Flat versus Hierarchical Classification in Large-Scale Taxonomies. In Proceedings of the Advances in Neural Information Processing Systems 26 (NIPS 2013), Lake Tahoe, NV, USA, 5–10 December 2013.

35. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of Tricks for Efficient Text Classification. In Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics, Valencia, Spain, 3–7 April 2017; pp. 427–431.

36. Kowsari, K.; Jafari Meimandi, K.; Heidarysafa, M.; Mendu, S.; Barnes, L.; Brown, D. Text Classification Algorithms: A Survey. *Information* **2019**, *10*, 150. [CrossRef]

37. Kalchbrenner, N.; Grefenstette, E.; Blunsom, P. A Convolutional Neural Network for Modelling Sentences. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, 23–25 June 2014; pp. 655–665.

38. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.

39. Kim, Y. Convolutional Neural Networks for Sentence Classification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1746–1751.

40. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. 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, Minnesota, MI, USA, 2–7 June 2019; pp. 4171–4186.

41. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C. HuggingFace's Transformers: State-of-the-Art Natural Language Processing. *arXiv* **2019**, arXiv:1910.03771.

42. Silla, C.; Freitas, A. A Survey of Hierarchical Classification across Different Application Domains. *Data Min. Knowl. Discov.* **2011**, *22*, 31–72. [CrossRef]

43. Kosmopoulos, A.; Partalas, I.; Gaussier, E.; Paliouras, G.; Androutsopoulos, I. Evaluation Measures for Hierarchical Classification: A Unified View and Novel Approaches. *Data Min. Knowl. Discov.* **2015**, *29*, 820–865. [CrossRef]

44. Lee, J.; Clifton, C. Differential Identifiability. In Proceedings of the International Conference on Knowledge Discovery and Data Mining, Beijing, China, 12–16 August 2012.

45. Bernau, D.; Eibl, G.; Grassal, P.; Keller, H.; Kerschbaum, F. Quantifying identifiability to choose and audit epsilon in differentially private deep learning. *Proc. VLDB Endow.* **2021**, *14*, 3335–3347. [CrossRef]

46. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. [CrossRef]

47. Nissim, K.; Raskhodnikova, S.; Smith, A. Smooth Sensitivity and Sampling in Private Data Analysis. In Proceedings of the Symposium on Theory of Computing, San Diego, CA, USA, 11–13 June 2007; pp. 75–84.

48. Lewis, D.D.; Yang, Y.; Rose, T.G.; Li, F. RCV1: A New Benchmark Collection for Text Categorization Research. *J. Mach. Learn. Res.* **2004**, *5*, 361–397.

49. Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P.N.; Hellmann, S.; Morsey, M.; van Kleef, P.; Auer, S.; et al. DBpedia—A Large-Scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semant. Web* **2015**, *6*, 167–195. [CrossRef]

50. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; Le, Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019.

51. Minaee, S.; Kalchbrenner, N.; Cambria, E.; Nikzad, N.; Chenaghlu, M.; Gao, J. Deep Learning Based Text Classification: A Comprehensive Review. *arXiv* **2020**, arXiv:2004.03705.

52. McMahan, H.B.; Andrew, G.; Erlingsson, U.; Chien, S.; Mironov, I.; Papernot, N.; Kairouz, P. A General Approach to Adding Differential Privacy to Iterative Training Procedures. In Proceedings of the Privacy Preserving Machine Learning, Workshop during the Conference on Neural Information Processing Systems, Montréal, QC, Canada, 2–8 December 2018.

53. Ezen-Can, A. A Comparison of LSTM and BERT for Small Corpus. *arXiv* **2020**, arXiv:2009.05451.

54. Neal, B.; Mittal, S.; Baratin, A.; Tantia, V.; Scicluna, M.; Lacoste-Julien, S.; Mitliagkas, I. A Modern Take on the Bias-Variance Tradeoff in Neural Networks. *arXiv* **2019**, arXiv:1810.08591.

55. Papernot, N.; Thakurta, A.; Song, S.; Chien, S.; Erlingsson, Ú. Tempered Sigmoid Activations for Deep Learning with Differential Privacy. *arXiv* **2020**, arXiv:stat.ML/2007.14191.