# Membership Inference Attacks on Sequence-to-Sequence Models: Is My Data In Your Machine Translation System?

**Sorami Hisamoto*** **Matt Post** **Kevin Duh**
Works Applications Johns Hopkins University
s@89.io {post,kevinduh}@cs.jhu.edu

## Abstract

Data privacy is an important issue for ''machine learning as a service'' providers. We focus on the problem of membership inference attacks: Given a data sample and black-box access to a model's API, determine whether the sample existed in the model's training data. Our contribution is an investigation of this problem in the context of sequence-to-sequence models, which are important in applications such as machine translation and video captioning. We define the membership inference problem for sequence generation, provide an open dataset based on state-of-the-art machine translation models, and report initial results on whether these models leak private information against several kinds of membership inference attacks.

## 1 Motivation

There are many situations in which private entities are worried about the privacy of their data. For example, many companies provide black-box training services where users are able to upload their data and have customized models built for them, without requiring machine learning expertise. A common concern in these ''machine learning as a service'' offerings is that the uploaded data be visible only to the client that owns it.

Currently, these entities are in the position of having to trust that service providers abide by the terms of their agreements. Although trust is an important component in relationships of all kinds, it has its limitations. In particular, it falls short of a well-known security maxim, originating in a Russian proverb that translates as, *Trust, but verify*.[1] Ideally, customers would be able to verify that their private data was not being slurped up by the serving company, whether by design or accident.

This problem has been formalized as the *membership inference* problem, first introduced by Shokri et al. (2017) and defined as: ''Given a machine learning model and a record, determine whether this record was used as part of the model's training dataset or not.'' The problem can be tackled in an adversarial framework: The attacker is interested in answering this question with high accuracy, whereas the defender would like this question to be unanswerable (see Figure 1). Since then, researchers have proposed many ways to attack and defend the privacy of various types of models. However, the work so far has only focused on standard classification problems, where the output space of the model is a fixed set of labels.

In this paper, we propose to investigate membership inference for *sequence generation* problems, where the output space can be viewed as a chained sequence of classifications. Prime examples of sequence generation includes machine translation and text summarization: In these problems, the output is a sequence of words whose length is undetermined a priori. Other examples include speech synthesis and video caption generation. Sequence generation problems are more complex than classification problems, and it is unclear whether the methods and results developed for membership inference in classification problems will transfer. For example, one might imagine that whereas a flat classification model might leak private information when the output is a single label, a recurrent sequence generation model might obfuscate this leakage when labels are generated successively with complex dependencies.

We focus on machine translation (MT) as the example sequence generation problem. Recent

*Work done while visiting Johns Hopkins University.

[1] Popularized by Ronald Reagan in the context of nuclear disarmament.



Transactions of the Association for Computational Linguistics, vol. 8, pp. 49–63, 2020. https://doi.org/10.1162/tacl_a_00299
Action Editor: Colin Cherry. Submission batch: 5/2019; Revision batch: 10/2019; Published 3/2020.

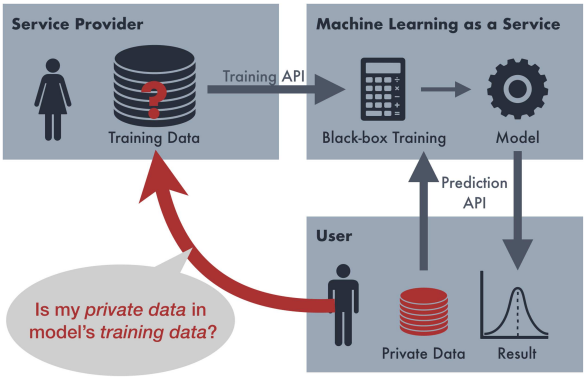


Figure 1: Membership inference attack.

advances in neural sequence-to-sequence models have improved the quality of MT systems significantly, and many commercial service providers are deploying these models via public API's. We pose the main question in the following form:

> *Given black-box access to an MT model, is it possible to determine whether a particular sentence pair was in the training set for that model?*

In the following, we define membership inference for sequence generation problems (§2) and contrast with prior work on classification (§3). Next we present a novel dataset (§4) based on state-of-the-art MT models.[2] Finally, we propose several attack methods (§5) and present a series of experiments evaluating their ability to answer the membership inference question (§6). Our conclusion is that simple one-off attacks based on shadow models, which proved successful in classification problems, are not successful on sequence generation problems; this is a result that favors the defender. Nevertheless, we describe the specific conditions where sequence-to-sequence models still leak private information, and discuss the possibility of more powerful attacks (§7).

## 2 Problem Definition

We now define the membership inference attack problem for sequence-to-sequence models in detail. Following tradition in the security research literature, we introduce three characters:

**Alice (*the service provider*)** builds a sequence-to-sequence model based on an undisclosed dataset $\mathcal{A}_{train}$ and provides a public API. For MT, this API takes a foreign sentence $f$ as input and returns an English translation $\hat{e}$.

**Bob (*the attacker*)** is interested in discerning whether a data sample was included in Alice's training data $\mathcal{A}_{train}$ by exploiting Alice's API. This sample is called a ''probe'' and consists of a foreign sentence $f$ and its reference English translation, $e$. Together with the API's output $\hat{e}$, Bob has to make a binary decision using a membership inference classifier $g(\cdot)$, whose goal is to predict:[3]

$$g(f, e, \hat{e}) = \begin{cases} \textbf{in} & \text{if probe} \in \mathcal{A}_{train} \\ \textbf{out} & \text{otherwise} \end{cases} \quad (1)$$

We term *in-probes* to be those probes where the true class is **in**, and *out-probes* to be those whose true class is **out**. Importantly, note that Bob has access not only to $f$ but also to $e$ in the probe. Intuitively, if $\hat{e}$ is equivalent to $e$, then Bob may believe that the probe was contained in $\mathcal{A}_{train}$; however, it may also be possible that Alice's model generalizes well to new samples and translates this probe correctly. The challenge for Bob is to make this distinction; the challenge for Alice is to prevent Bob from doing so.

**Carol (*the neutral third-party*)** is in charge of setting up the experiment between Alice and Bob. She decides which data samples should be used as in-probes and out-probes and evaluates Bob's classification accuracy. Carol is introduced only to clarify the exposition and to set up a fair experiment for research purposes. In practical scenarios, Carol does not exist: Bob decides his own probes, and Alice decides her own $\mathcal{A}_{train}$.

### 2.1 Detailed Specification

In order to be precise about how Carol sets up the experiment, we will explain in terms of machine translation, but note that the problem definition applies to any sequence-to-sequence problem. A training set for MT consists of a set of sentence pairs $\{(f_i^{(d)}, e_i^{(d)})\}$. We use a label $d \in \{\ell_1, \ell_2, \ldots\}$ to indicate the domain

[2] We release the data to encourage further research in this new problem: `https://github.com/sorami/tacl-membership`

[3] In the experiments, we will also consider extending the information available to Bob. For example, if Alice additionally provides the translation probabilities $\rho$ in the API, then Bob can exploit that in the classifier as $g(f, e, \hat{e}, \rho)$.

(the subcorpus or the data source), and an index $i \in \{1, 2, \ldots, I(d)\}$ to indicate the sample id in the domain (subcorpus). For example, $e_i^{(d)}$ with $d = \ell_1$ and $i = 1$ might refer to the first sentence in the `Europarl` subcorpus, while $e_i^{(d)}$ with $d = \ell_2$ and $i = 1$ might refer to the first sentence in the `CommonCrawl` subcorpus. $I(d)$ is the maximum number of sentences in the subcorpus with label $d$. The distinction among subcorpora is not necessary in the abstract problem definition, but is important in practice when differences in data distribution may reveal signals in membership.

Without loss of generality, in this section assume that Carol has a finite number of samples from two subcorpora $d \in \{\ell_1, \ell_2\}$. First, she creates an out-probe of $k$ samples from subcorpus $\ell_1$:

$$\mathcal{A}_{out\_probe} = \left\{ (f_i^{(d)}, e_i^{(d)}) : \begin{array}{l} d = \ell_1, \ell_2 \\ i = 1, \ldots, k \end{array} \right\} \quad (2)$$

Then Carol creates the data for Alice to train Alice's MT model, using subcorpora $\ell_1$ and $\ell_2$:

$$\mathcal{A}_{train} = \left\{ (f_i^{(d)}, e_i^{(d)}) : \begin{array}{l} d = \ell_1, \ell_2 \\ i = k+1, \ldots, I(d) \end{array} \right\} \quad (3)$$

Importantly, the two sets are totally disjoint: i.e., $\mathcal{A}_{out\_probe} \cap \mathcal{A}_{train} = \emptyset$. By definition, out-probes are sentence pairs that are not in Alice's training data. Finally, Carol creates the in-probe of $k$ samples by drawing from $\mathcal{A}_{train}$, i.e. $\mathcal{A}_{in\_probe} \subset \mathcal{A}_{train}$, which is defined to be samples that are included in training:

$$\mathcal{A}_{in\_probe} = \left\{ (f_i^{(d)}, e_i^{(d)}) : \begin{array}{l} d = \ell_1, \ell_2 \\ i = k+1, \ldots, 2k \end{array} \right\} \quad (4)$$

Note that both $\mathcal{A}_{in\_probe}$ and $\mathcal{A}_{out\_probe}$ are sentence pairs that come from the same subcorpus; the only difference is that the former is included in $\mathcal{A}_{train}$ whereas the latter is not.

There are several ways in which Bob's data can be created. For this work, we will assume that Bob also has some data to train MT models, in order to mimic Alice and design his attacks. This data could either be disjoint from $\mathcal{A}_{train}$, or contain parts of $\mathcal{A}_{train}$. We choose the latter, which assumes that there might be some public data that is accessible to both Alice and Bob. This scenario slightly favors Bob. In the case of MT, parallel data can be hard to come by, and datasets like `Europarl` are widely accessible to anyone, so presumably both Alice and Bob would use it. However, we expect that Alice has an in-house dataset (e.g., crawled data) that Bob does not have access to. Thus, Carol creates data for Bob:

$$\mathcal{B}_{all} = \left\{ (f_i^{(d)}, e_i^{(d)}) : \begin{array}{l} d = \ell_1 \\ i = 2k+1, \ldots, I(d) \end{array} \right\} \quad (5)$$

Note that this dataset is like $\mathcal{A}_{train}$ but with two exceptions: All samples from subcorpora $\ell_2$ and all samples from $\mathcal{A}_{in\_probe}$ are discarded. One can view $\ell_2$ as Alice's own in-house corpus which Bob has no knowledge of or access to, and $\ell_1$ as the shared corpus where membership inference attacks are performed.

To summarize, Carol gives $\mathcal{A}_{train}$ to Alice, who uses it in whatever way she chooses to build a sequence-to-sequence model $M[\mathcal{A}_{train}, \Theta]$. The model is trained on $\mathcal{A}_{train}$ with hyperparameters $\Theta$ (e.g., neural network architecture) known only to Alice. In parallel, Carol gives $\mathcal{B}_{all}$ to Bob, who uses it to design various attack strategies, resulting in a classifier $g(\cdot)$ (see Section 5). When it is time for evaluation, Carol provides both probes $\mathcal{A}_{in\_probe}$ and $\mathcal{A}_{out\_probe}$ to Bob in randomized order and asks Bob to classify each sample as **in** or **out**. For each probe $(f_i^{(d)}, e_i^{(d)})$, Bob is allowed to make one call to Alice's API to obtain $\hat{e}_i^{(d)}$.

As an additional evaluation, Carol creates a third probe based on a new subcorpus $\ell_3$. We call this the ''out-of-domain (OOD) probe'':

$$\mathcal{A}_{ood} = \left\{ (f_i^{(d)}, e_i^{(d)}) : \begin{array}{l} d = \ell_3 \\ i = 1, \ldots, k \end{array} \right\} \quad (6)$$

Both $\mathcal{A}_{out\_probe}$ and $\mathcal{A}_{ood}$ should be classified as **out** by Bob's classifier. However, it has been known that sequence-to-sequence models behave very differently on data from domains/genre that is significantly different from the training data (Koehn and Knowles, 2017). The goal of having two **out** probes is to quantify the difficulty or ease of membership inference in different situations.

### 2.2 Summary and Alternative Definitions

Figure 2 summarizes the problem definition. The probes $\mathcal{A}_{out\_probe}$ and $\mathcal{A}_{ood}$ are by construction outside of Alice's training data $\mathcal{A}_{train}$, while the probe $\mathcal{A}_{in\_probe}$ is included. Bob's goal is to produce a classifier that can make this distinction.

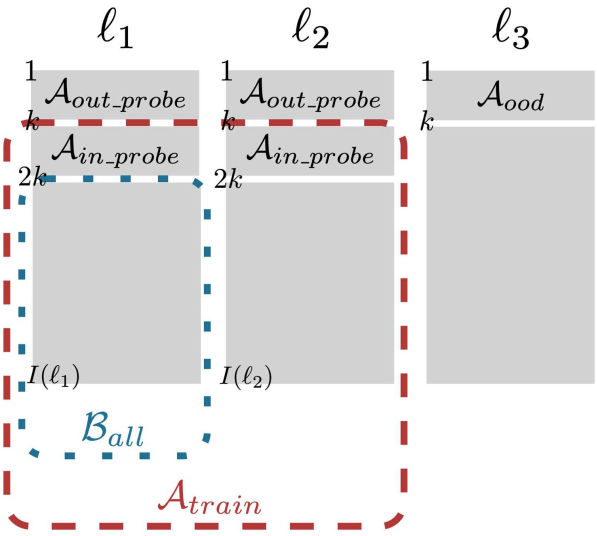


Figure 2: Illustration of data splits for Alice and Bob. There are $k$ samples each for $\mathcal{A}_{in\_probe}$, $\mathcal{A}_{out\_probe}$, and $\mathcal{A}_{ood}$. Alice's training data $\mathcal{A}_{train}$ excludes $\mathcal{A}_{out\_probe}$ and $\ell_3$, while including $\mathcal{A}_{in\_probe}$. Bob's data $\mathcal{B}_{all}$ is a subset of Alice's data, excluding $\mathcal{A}_{in\_probe}$ and $\ell_2$.

He has at his disposal a smaller dataset $\mathcal{B}_{all}$, which he can use in whatever way he desires.

There are alternative definitions of this membership inference problem. For example, one can allow Bob to make multiple API calls to Alice's model for each probe. This enlarges the repository of potential attack strategies for Bob. Or, one could evaluate Bob's accuracy not on a per-sample basis, but allow for a coarser granularity where Bob can aggregate inferences over multiple samples. There is also a distinction between white-box and black-box attacks: We focus on the black-box case where Bob has no internal access to the internal parameters of Alice's model, but can only guess at likely model architectures. In the white-box case, Bob would have access to Alice's model internals, so different attacks would be possible (e.g., backpropagation of gradients). In these respects, our problem definition makes the problem more challenging for Bob the attacker.

Finally, note that Bob is not necessarily always the ''bad guy''. Some examples of who Alice and Bob might be in MT are: (1) Organizations (Bob) that provide bitext data under license restrictions might be interested to determine whether their licenses are being complied with in published models (Alice). (2) The organizers (Bob) of an annual bakeoff (e.g., WMT) might wish to confirm that the participants (Alice) are following the rules of not training on test data. (3) ''MT as a service'' providers may support customized engines if users upload their own bitext training data. The provider promises that the user-supplied data will not be used in the customized engines of other users, and can play both Alice and Bob, attacking its own model to provide guarantees to the user. If it is possible to construct a successful membership inference mechanism, then many ''good guys'' would be able to provide the aforementioned fairness (1, 2) and privacy guarantees (3).

## 3 Related Work

Shokri et al. (2017) introduced the problem of membership inference attacks on machine learning models. They showed that with shadow models trained on either realistic or synthetic datasets, Bob can build classifiers that can discriminate $\mathcal{A}_{in\_probe}$ and $\mathcal{A}_{out\_probe}$ with high accuracy. They focus on classification problems such as CIFAR image recognition and demonstrate successful attacks on both convolutional neural net models as well as the models provided by Amazon ML.

Why do these attacks work? The main information exploited by Bob's classifier is the output distribution of class labels returned by Alice's API. The prediction uncertainty differs for data samples inside and outside the model training data, and this can be exploited. Shokri et al. (2017) propose defense strategies for Alice, such as restricting the prediction vector to top-$k$ classes, coarsening the values of the output probabilities, and increasing the entropy of the prediction vector. The crucial difference between their work and ours, besides our focus on sequence generation problems, is the availability of this kind of output distribution provided by Alice. Although it is common to provide the whole distribution of output probabilities in classification problems, this is not possible in sequence generation problems because the output space of sequences is exponential in the output length. At most, sequence models can provide a score for the output prediction $\hat{e}_i^{(d)}$, for example with a beam search procedure, but this is only one number and not normalized. We do experiment with having Bob exploit this score (Table 3), but it appears far inferior to the use of the whole distribution available in classification problems.

Subsequent work on membership inference has focused on different angles of the problem. Salem et al. (2018) investigated the effect of training the

shadow model and datasets that match or do not match the distribution of $\mathcal{A}_{train}$, and compared training a single shadow model as opposed to many. Truex et al. (2018) present a comprehensive evaluation of different model types, training data, and attack strategies. Borrowing ideas from adversarial learning and minimax games, Hayes et al. (2017) propose attack methods based on generative adversarial networks, while Nasr et al. (2018) provide adversarial regularization techniques for the defender. Nasr et al. (2019) extend the analysis to white-box attacks and a federated learning setting. Pyrgelis et al. (2018) provide an empirical study on location data. Veale et al. (2018) discuss membership inference and the related model inversion problem, in the context of data protection laws like GDPR.

Shokri et al. (2017) note a synergistic connection between the goals of learning and the goals of privacy in the case of membership inference: The goal of learning is to generalize to data outside the training set (e.g., so that $\mathcal{A}_{out\_probe}$ and $\mathcal{A}_{ood}$ are translated well), while the goal of privacy is to prevent leaking information about data in the training set. The common enemy of both goals is overfitting. Yeom et al. (2017) analyze how overfitting by Alice's increases the risk privacy leakage; Long et al. (2018) showed that even a well-generalized model holds such risks in classification problems, implying that overfitting by Alice is a sufficient but not necessary condition for privacy leakage.

A large body of work exists in differential privacy (Dwork, 2008; Machanavajjhala et al., 2017). Differential privacy provides guarantees that a model trained on some dataset $\mathcal{A}_{train}$ will produce statistically similar predictions as a model trained on another dataset which differs in exactly one sample. This is one way in which Alice can defend her model (Rahman et al., 2018), but note that differential privacy is a stronger notion and often involves a cost in Alice's model accuracy. Membership inference assumes that content of the data is known to Bob and only is concerned whether it was used. Differential privacy also protects the content of the data (i.e., the actual words in $(f_i^{(d)}, e_i^{(d)})$ should not be inferred).

Song and Shmatikov (2019) explored the membership inference problem of natural language text, including word prediction and dialog generation. They assume that the attacker has access to a probability distribution or a sequence of distributions over the vocabulary for the generated word or sequence. This is different from our work where the attacker gets only the output sequence, which we believe is a more realistic setting.

## 4 Data and Evaluation Protocol

### 4.1 Data: Subcorpora and Splits

Based on the problem definition in Section 2, we construct a dataset to investigate the possibility of the membership inference attack on MT models. We make this dataset available to the public to encourage further research.[4]

There are various considerations to ensure the benchmark is fair for both Alice and Bob: We need a dataset that is large and diverse to ensure Alice can train state-of-the-art MT models and Bob can test on probes from different domains. We used corpora from the Conference on Machine Translation (WMT18) (Bojar et al., 2018). We chose the German–English language pair because it has a reasonably large amount of training data, and previous work demonstrate high BLEU scores.

We now describe how Carol prepares the data for Alice and Bob. First, Carol selects four subcorpora for the training data of Alice, namely, `CommonCrawl`, `Europarl v7`, `News Commentary v13`, and `Rapid 2016`. A subset of these four subcorpora are also available to Bob ($\ell_1$ in § 2.1). In addition, Carol gives `ParaCrawl` to Alice but not Bob ($\ell_2$ in §2.1). We can think of it as in-house data that the service provider holds. For all these subcorpora, Carol first performs basic preprocessing: (a) tokenization of both the German and English sides using the Moses tokenizer, (b) de-duplication of sentence pairs so that only unique pairs are present, and (c) randomly shuffling all sentences prior to splitting into probes and MT training data.[5]

Figure 3 illustrates how Carol splits subcorpora for Alice and Bob. For each subcorpus, Carol splits

[4] `https://github.com/sorami/tacl-membership`

[5] These are design decisions that balance between simple experimentation vs. realistic condition. Carol doing a common tokenization removes some of the MT-specific complexity for researchers who want to focus on the Alice or Bob models. However, in a real-world public API, Alice's tokenization is likely to be unknown to Bob. We decided on a middle ground to have Carol perform a common tokenization, but Alice and Bob do their own subword segmentation.

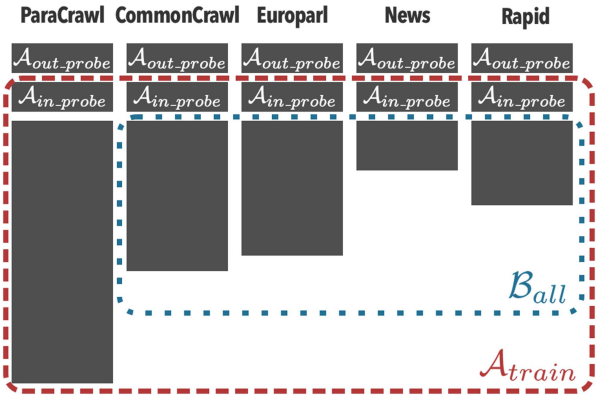


Figure 3: Illustration of actual MT data splits. $\mathcal{A}_{train}$ does not contain $\mathcal{A}_{out\_probe}$, and $\mathcal{B}_{all}$ is a subset of $\mathcal{A}_{train}$ with $\mathcal{A}_{in\_probe}$ and `ParaCrawl` excluded.

them to create probes $\mathcal{A}_{in\_probe}$ and $\mathcal{A}_{out\_probe}$, and $\mathcal{A}_{train}$ and $\mathcal{B}_{all}$. Carol sets $k = 5,000$, meaning each probe set per subcorpus has 5,000 samples. For each subcorpus, Carol selects 5,000 samples to create $\mathcal{A}_{out\_probe}$. She then uses the rest as $\mathcal{A}_{train}$ and select 5,000 from it as $\mathcal{A}_{in\_probe}$. She excludes $\mathcal{A}_{in\_probe}$ and `ParaCrawl` from $\mathcal{A}_{train}$ to create a dataset for Bob, $\mathcal{B}_{all}$.[6] In addition, Carol has four other domains to create out-of-domain probe set $\mathcal{A}_{ood}$, namely, `EMEA` and `Subtitles 18` (Tiedemann, 2012), `Koran` (Tanzil), and TED (Duh, 2018). These subcorpora are equivalent to $\ell_3$ in § 2.1. The size of $\mathcal{A}_{ood}$ is 5,000 per subcorpus, same as $\mathcal{A}_{in\_probe}$ and $\mathcal{A}_{out\_probe}$. The number of samples for each set is summarized in Table 1.

### 4.2 Alice MT Architecture

Alice uses her dataset $\mathcal{A}_{train}$ (consisting of four subcorpora and `ParaCrawl`) to train her own MT model. Because `Paracrawl` is noisy, Alice first applies dual conditional cross-entropy filtering (Junczys-Dowmunt, 2018), retaining the top 4.5 million lines. Alice then trains a joint BPE subword model (Sennrich et al., 2016) using 32,000 merge operations. No recasing is applied.

Alice's model is a six-layer Transformer (Vaswani et al., 2017) using default parameters in Sockeye (Hieber et al., 2017).[7] The model was trained until perplexity on `newstest2017` (Bojar et al., 2017) had not improved for five consecutive checkpoints, computed every 5,000 batches.

The BLEU score (Papineni et al., 2002) on `newstest2018` was 42.6, computed using sacreBLEU (Post, 2018) with the default settings.[8]

### 4.3 Evaluation Protocol

To evaluate membership inference attacks on Alice's MT models, we use the following procedure: First, Bob asks Alice to translate $f$. Alice returns her result $\hat{e}$ to Bob. Bob also has access to the reference $e$ and use his classifier $g(f, e, \hat{e})$ to infer whether $(e, f)$ was in Alice's training data. The classification is reported to Carol, who computes ''attack accuracy''. Given a probe set $P$ containing a list of $(f, e, \hat{e}, l)$, where $l$ is the label (**in** or **out**), this accuracy is defined as:

$$accuracy(g, P) = \frac{1}{|P|} \sum^{P} [g(f, e, \hat{e}) = l] \quad (7)$$

If the accuracy is 50%, then the binary classification is same as random, and Alice is safe. An accuracy slightly above 50% can be considered a potential breach of privacy.

## 5 Membership Inference Attacks

### 5.1 Shadow Model Framework

Bob's initial approach for attack is to use ''shadow models'', similar to Shokri et al. (2017). The idea is that Bob creates MT models with his data to mimic (shadow) the behavior of Alice's MT model, then train a membership inference classifier on these shadow models. To do so, Bob splits his data $\mathcal{B}_{all}$ into his own version of in-probe, out-probe, and training set in multiple ways to train MT models. Then he translates these probe sentences with his own shadow MT models, and use the resulting $(f, e, \hat{e})$ with its **in** or **out** label to train a binary classifier $g(f, e, \hat{e})$. If Bob's shadow models are sufficiently similar to Alice's in behavior, this attack can work.

Bob first selects 10 sets of 5,000 sentences per subcorpus in $\mathcal{B}_{all}$. He then chooses two sets and uses one as in-probe and the other as out-probe, and combines in-probe and the rest ($\mathcal{B}_{all}$ minus 10 sets) as a training sets. We use notations

[6] We prepared two different pairs of $\mathcal{A}_{in\_probe}$ and $\mathcal{A}_{out\_probe}$. Thus $\mathcal{B}_{all}$ has 10k fewer samples than $\mathcal{A}_{train}$, and not 5k fewer. For the experiment we used only one pair, and kept the other for future use.

[7] Three-way tied embeddings, model and embedding size 512, eight attention heads, 2,048 hidden states in the feed forward layers, layer normalization applied before each self-attention layer, and dropout and residual connections applied afterward, word-based batch size of 4,096.

[8] Version 1.2.12, case-sensitive, ''13a'' tokenization for comparability with WMT.

| | $\mathcal{A}_{out\_probe}$ | $\mathcal{A}_{in\_probe}$ | $\mathcal{A}_{train}$ | $\mathcal{B}_{all}$ | $\mathcal{A}_{ood}$ |
|---|---|---|---|---|---|
| ParaCrawl | 5,000 | 5,000 | 4,518,029 | 0 | N/A |
| CommonCrawl | 5,000 | 5,000 | 2,389,123 | 2,379,123 | N/A |
| Europarl | 5,000 | 5,000 | 1,865,271 | 1,855,271 | N/A |
| News | 5,000 | 5,000 | 273,702 | 263,702 | N/A |
| Rapid | 5,000 | 5,000 | 1,062,214 | 1,052,214 | N/A |
| EMEA | N/A | N/A | N/A | N/A | 5,000 |
| Koran | N/A | N/A | N/A | N/A | 5,000 |
| Subtitles | N/A | N/A | N/A | N/A | 5,000 |
| TED | N/A | N/A | N/A | N/A | 5,000 |
| TOTAL | 25,000 | 25,000 | 10,108,339 | 5,550,310 | 20,000 |

Table 1: Number of sentences per set and subcorpus. For each subcorpus, $\mathcal{A}_{train}$ includes $\mathcal{A}_{in\_probe}$ and does not include $\mathcal{A}_{out\_probe}$. $\mathcal{B}_{all}$ is a subset of $\mathcal{A}_{train}$, excluding $\mathcal{A}_{in\_probe}$ and ParaCrawl. $\mathcal{A}_{ood}$ is for evaluation only, and only Carol has access to it.

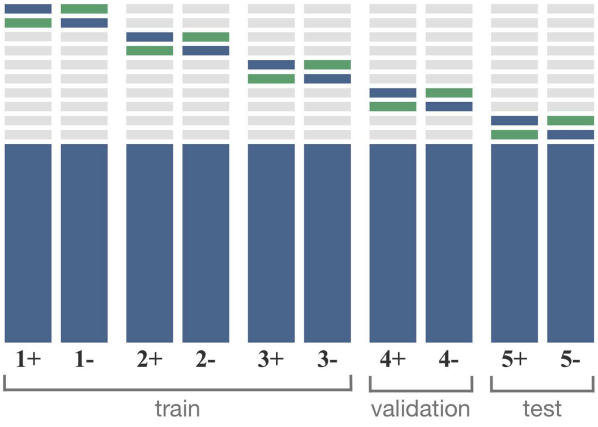


Figure 4: Illustration of how Bob splits $\mathcal{B}_{all}$ for each shadow model. Blue boxes are the in-probe $\mathcal{B}_{in\_probe}$ and training data $\mathcal{B}_{train}$, where small box is the in-probe and small and large boxes combined is the training data. Green box indicates the out-probe $\mathcal{B}_{out\_probe}$. Bob uses models from splits 1 to 3 as a train, 4 as a validation, and 5 as a test sets for his attack.

$\mathcal{B}^{1+}_{in\_probe}$ $\mathcal{B}^{1+}_{out\_probe}$, and $\mathcal{B}^{1+}_{train}$ for the first group of in-probe, out-probe, and training sets. Bob then swaps the in-probe and out-probe to create another group. We notate this as $\mathcal{B}^{1-}_{in\_probe}$, $\mathcal{B}^{1-}_{out\_probe}$, and $\mathcal{B}^{1-}_{train}$. With 10 sets of 5,000 sentences, Bob can create 10 different groups of in-probe, out-probe, and training sets. Figure 4 illustrates the data splits.

For each group of data, Bob first trains a shadow MT model using the training set. He then uses this model to translate sentences in the in-probe and out-probe sets. Bob has now a list of $(f, e, \hat{e})$ from different shadow models, and he knows for each sample if it was **in** or **out** of the training data for the MT model used to translate that sentence.

### 5.2 Bob MT Architecture

Bob's model is a 4-layer Transformer, with no tied embedding, model/embedding size 512, 8 attention heads, 1,024 hidden states in the feed forward layers, word-based batch size of 4,096. The model is optimized with Adam (Kingma and Ba, 2015), regularized with label smoothing (0.1), and trained until perplexity on newstest2016 (Bojar et al., 2016) had not improved for 16 consecutive checkpoints, computed every 4,000 batches. Bob has BPE subword models with vocab size 30k for each language. The mean BLEU scores of the ten shadow models on newstest2018 is 38.6±0.2 (compared with 42.6 for Alice).

### 5.3 Membership Inference Classifier

Bob extracts features from $(f, e, \hat{e})$ for a binary classifier. He uses modified 1- to 4-gram precisions and smoothed sentence-level BLEU score (Lin and Och, 2004) as features. Bob's intuition is that if an unusually large number of $n$-grams in $\hat{e}$ matches $e$, then it could be a sign that this was in the training data and Alice memorized it. Bob calculates $n$-gram precision by counting the number of $n$-grams in translation that appear in the reference sentence. In the later investigation Bob also considers the MT model score as an extra feature.

**Algorithm 1:** Construction of A Membership Inference Classifier

**Data:** $\mathcal{B}_{all}$
**Result:** $g(\cdot)$
Split $\mathcal{B}_{all}$ into multiples groups of ($\mathcal{B}^i_{in\_probe}$, $\mathcal{B}^i_{out\_probe}$, $\mathcal{B}^i_{train}$) ;
**foreach** $i$ *in* $1+, 1-, 2+, 2-, 3+, 3-$ **do**
    Train a shadow model $M_i$ using $\mathcal{B}^i_{train}$ ;
    Translate $\mathcal{B}^i_{in\_probe}$, $\mathcal{B}^i_{out\_probe}$ with $M_i$ ;
**end**
Use $\mathcal{B}^i_{in\_probe}$, $\mathcal{B}^i_{out\_probe}$, and their translations to train $g(\cdot)$ ;

Bob tries different types of classifiers, namely, Perceptron (P), Decision Tree (DT), Naïve Bayes (NB), Nearest Neighbors (NN), and Multi-layer Perceptron (MLP). DT uses GINI impurity for the splitting metrics, and the max depth to be 5. Our NB uses Gaussian distribution. For NN we set the number of neighbors to be 5 and use Minkowski distance. For MLP, we set the size of the hidden layer to be 100, the activation function to be ReLU, and the L2 regularization term $\alpha$ to be 0.0001.

Algorithm 1 summarizes the procedure to construct a membership inference classifier $g(\cdot)$ using Bob's dataset $\mathcal{B}_{all}$. For training the binary classifiers, Bob uses models from data splits 1 to 3 for training, 4 for validation, and 5 for his own internal testing. Note that the final evaluation of the attack is done using the translations of $\mathcal{A}_{in\_probe}$ and $\mathcal{A}_{out\_probe}$ with Alice's MT model, by Carol.

## 6 Attack Results

We now present a series of results based on the shadow model attack method described in Section 5. In Section 6.1 we will observe that Bob has difficulty attacking Alice under our definition of membership inference. In Sections 6.2 and 6.3 we will see that Alice nevertheless does leak some private information under more nuanced conditions. Section 6.4 describes the possibility of attacks beyond sentence-level membership. Section 6.5 explores the attacks using external resources.

### 6.1 Main Result

Table 2 shows the accuracy of the membership inference classifiers. There are 5 different types of classifiers, as described in Section 5.3. The numbers in the *Alice* column shows the attack accuracy on Alice probes $\mathcal{A}_{in\_probe}$ and $\mathcal{A}_{out\_probe}$; these are the main results. The numbers in *Bob* columns show the results on the Bob classifiers' train, validation, and test sets, as described in Section 5.3.

| | Alice | Bob:train | Bob:valid | Bob:test |
|---|---|---|---|---|
| P | 50.0 | 50.0 | 50.0 | 50.0 |
| DT | 50.4 | 51.4 | 51.2 | 51.1 |
| NB | 50.4 | 51.2 | 51.1 | 51.0 |
| NN | 49.9 | 61.6 | 50.5 | 50.0 |
| MLP | 50.2 | 50.8 | 50.8 | 50.8 |

Table 2: Accuracy of membership inference per classifier type, Perceptron (P), Decision Tree (DT), Naïve Bayes (NB), Nearest Neighbors (NN), and Multi-layer Perceptron (MLP). *Alice* column shows the accuracy of attack on Alice probes $\mathcal{A}_{in\_probe}$ and $\mathcal{A}_{out\_probe}$. *Bob* columns show the accuracy on the classifiers' train, validation, and test set. Note that, following the evaluation protocol explained in Section 4.3, only Carol the evaluator can observe the accuracy of the attacks on Alice model.

The results of the attacks on the Alice model show that it is around 50%, meaning that the attack is not successful and the binary classification is almost the same as a random choice.[9] The accuracy is around 50% for Bob:valid, meaning that Bob also has difficulty attacking his own simulated probes, therefore the poor performance on $\mathcal{A}_{in\_probe}$ and $\mathcal{A}_{out\_probe}$ is not due to mismatches between Alice's model and Bob's model.

The accuracy is around 50% for Bob:train as well, revealing that the classifier $g(\cdot)$ is underfitting.[10] This suggests that the current features do not provide enough information to distinguish in-probe and out-probe sentences. Figure 5 shows

[9] Some numbers are slightly over 50%, which may be interpreted as small leak of privacy. Although the desired accuracy levels depend on the application, for the MT scenarios described in Section 2.2 Bob would need much higher accuracies. For example, if Bob is a bakeoff organizer, he might want accuracy above 60% in order to determine whether to manually check the submission. However, if Bob is providing ''MT as a service'' with strong privacy guarantees, he may need to provide the client with accuracy higher than 90%.

[10] The higher accuracy for $k$-NN is an exception, but is due to having the exact same datapoint in the model as the input, which always becomes the nearest neighbor. When the $k$ value is increased, the accuracy on in-sample data decreased.

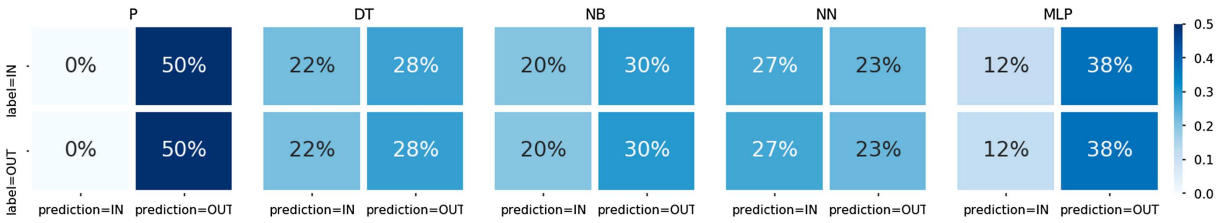


Figure 5: Confusion matrices of the attacks on Alice model per classifier type.

| | Alice | Bob:train | Bob:valid | Bob:test |
|---|---|---|---|---|
| P | 49.7 | 49.2 | 49.3 | 49.4 |
| DT | 50.4 | 51.5 | 51.1 | 51.2 |
| NB | 50.1 | 50.2 | 50.1 | 50.2 |
| NN | 50.2 | 67.1 | 50.2 | 50.0 |
| MLP | 50.4 | 51.2 | 51.2 | 51.1 |

Table 3: Membership inference accuracy when MT model score is added as an extra classifier feature.

the confusion matrices of the classifier output on Alice probes. We see that for all classifiers, whatever prediction they make is incorrect half of the time.

Table 3 shows the result when *MT model score* is added as an extra feature for classification. The result indicates that this extra information does not improve the attack accuracy. In summary, these results suggest that Bob is not able to reveal membership information at the sentence/sample level. This result is in contrast to previous work on membership inference in ''classification'' problems, which demonstrated high accuracy with Bob's shadow model attack.

Additionally, note that although accuracies are close to 50%, the number of Bob:test tend to be slightly higher than Alice's for some classifiers. This may reflect the fact that Bob:test is a matched condition using the same shadow MT architecture, while Alice probes are from a mismatched condition using an unknown MT architecture. It is important to compare both numbers in the experiments: accuracy on Alice probes is the real evaluation and accuracy on Bob:test is a diagnostic.

### 6.2 Out-of-Domain Subcorpora

Carol prepared OOD subcorpora, $\mathcal{A}_{ood}$, that are separate from $\mathcal{A}_{train}$ and $\mathcal{B}_{all}$. The membership inference accuracy of each subcorpus is shown in Table 4. The accuracy for OOD subcorpora are much higher than that of original in-domain subcorpora. For example, the accuracy with Decision Tree was 50.3% and 51.1% for `ParaCrawl` and `CommonCrawl` (in-domain), whereas accuracy was 67.2% and 94.1% for `EMEA` and `Koran` (out-of-domain). This suggests that for OOD data Bob has a better chance to infer the membership.

In Table 4 we can see that Perceptron has accuracy 50% for all in-domain subcorpora and 100% for all OOD subcorpora. Note that the OOD subcorpora only have *out-probes*. By definition none of the samples from OOD subcorpora are in the training data. We get such accuracy because our Perceptron is always predicting **out**, as we can see in Figure 5. We believe this behavior is caused by applying Perceptron to inseparable data, and this particular model happened to be trained to act this way. To confirm this we have trained variations of Perceptrons by shuffling the training data, and observed that the resulting models had different output ratios of **in** and **out**, and in some cases always predicting **in** for both in and OOD subcorpora.

Figure 6 shows the distribution of sentence-level BLEU scores per subcorpus. The BLEU scores tend to be lower for OOD subcorpora, and the classifier may exploit this information to distinguish the membership better. But note that `EMEA` (out-of-domain) and `CommonCrawl` (in-domain) have similar BLEU scores, but vastly different membership accuracies, so the classifier may also be exploiting $n$-gram match distributions.

Overall, these results suggest that Bob's accuracy depends on the specific type of probe being tested. If there is a wide distribution of domains, there is a higher chance that Bob may be able to reveal membership information. Note that in the actual scenario Bob will have no way of knowing what is OOD for Alice, so there is no signal that is exploitable for Bob. This section is meant as an error analysis that describes how membership inference classifiers behave differently in case the probe is OOD.

| | ParaCrawl | CommonCrawl | Europarl | News | Rapid | EMEA | Koran | Subtitles | TED |
|---|---|---|---|---|---|---|---|---|---|
| P | 50.0 | 50.0 | 50.0 | 50.0 | 50.0 | 100.0 | 100.0 | 100.0 | 100.0 |
| DT | 50.3 | 51.1 | 49.7 | 50.7 | 50.0 | 67.2 | 94.1 | 80.2 | 67.1 |
| NB | 50.1 | 51.2 | 49.9 | 50.6 | 50.2 | 69.5 | 96.1 | 81.7 | 70.5 |
| NN | 49.4 | 50.7 | 50.3 | 49.7 | 49.2 | 43.3 | 52.6 | 48.7 | 49.9 |
| MLP | 49.6 | 50.8 | 49.9 | 50.3 | 50.7 | 73.6 | 97.9 | 84.8 | 85.0 |

Table 4: Membership inference accuracy per subcorpus. The right-most 4 columns are results for out-of-domain subcorpora. Note that `ParaCrawl` is *out-of-domain* for Bob and his classifier, although it is *in-domain* for Alice and her MT model.

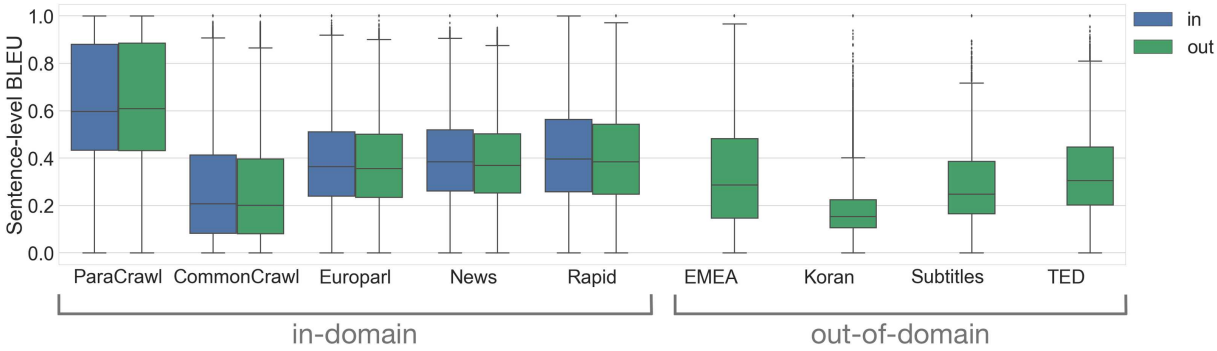


Figure 6: Distribution of sentence-level BLEU per subcorpora for $\mathcal{A}_{in\_probe}$ (blue boxes), $\mathcal{A}_{out\_probe}$ (green, left five boxes), and $\mathcal{A}_{ood}$ (green, right four boxes).

### 6.3 Out-of-Vocabulary Words

We also focused on the samples that contain the words that never appear in the training data of the MT model used for translation, that is, out-of-vocabulary (OOV) words. For this analysis, we focus only on vocabulary that does not exist in the training data of Bob's shadow MT models, rather than Alice's, since Bob does not have access to her vocabulary. By definition there are only *out-probes* in OOV subsets.

For Bob's shadow models, 7.4%, 3.2%, and 1.9% of samples in the probe sets had one or more OOV words in source, reference, or both sentences, respectively. Table 5 shows the membership inference accuracy of the OOV subsets from the Bob test set, which is generally very high (>70%). This implies that sentences with OOV words are translated idiosyncratically compared with the ones without OOV words, and the classifier can exploit this.

### 6.4 Alternative Evaluation: Grouping Probes

Section 6.1 showed that it is generally difficult for Bob to determine membership for the strict definition of one sentence per probe. What if we loosen the problem, letting the probe be a group of sentences?

| | OOV in src | OOV in ref | OOV in both |
|---|---|---|---|
| P | 100.0 | 100.0 | 100.0 |
| DT | 73.9 | 74.1 | 68.0 |
| NB | 77.4 | 77.0 | 70.3 |
| NN | 49.9 | 49.2 | 49.3 |
| MLP | 89.0 | 85.8 | 80.4 |

Table 5: Membership inference accuracy on the sentences in Bob:test containing out-of-vocabulary (OOV) words for the MT model used for translation.

We create probes of 500 sentences each to investigate this hypothesis. Bob randomly samples 500 sentences with the same label from Bob's training set to form a probe group. To create sufficient training data for his classifier, Bob repeats sampling and creates 6,000 groups. Bob uses sentence BLEU bin percentage and corpus BLEU as features for classification. For each group, Bob counts the sentence BLEU for each bin. The bin size is set to 0.01. Bob also uses all 500 translations together to calculate the group's corpus BLEU score. Bob trains the classifiers using these features, and applies it to Bob's validation and test

| | Bob | | | Alice | |
|---|---|---|---|---|---|
| | train | valid | test | original | adjusted |
| P | 71.6 | 69.4 | 68.1 | 50.0 | 59.0 |
| DT | 70.4 | 65.6 | 64.4 | 52.0 | 61.0 |
| NB | 72.9 | 67.5 | 70.0 | 50.0 | 50.0 |
| NN | 77.4 | 66.9 | 62.5 | 51.0 | 50.0 |
| MLP | 73.0 | 68.8 | 70.0 | 50.0 | 52.0 |

Table 6: Attack accuracy on probe groups. In addition to the *original* Alice set, we have the *adjusted* set where the feature values are adjusted by subtracting the mean BLEU difference between Alice and Bob models.

sets, and Alice sets. These sets are evenly split into groups of 500, not sampled as done in training.

Table 6 shows the accuracy on probe groups. We can see that the accuracy is much higher than 50%, not only for Bob's training set but also for his validation and test sets. However, for Alice, we found that classifiers were almost always predicting **in**, resulting the accuracy to be around 50%. This is due to the fact that classifiers were trained on shadow models that have lower BLEU scores than Alice. This suggests that we need to incorporate the information about the Alice / Bob MT performance difference.

One way to adjust the difference is to directly manipulate the input feature values. We adjusted the feature values, compensating by the difference in mean BLEU scores, and accuracy on Alice probes increased to 60% for P and DT as shown in the ''adjusted'' column of Table 6. If the classifier took advantage of the absolute values in its decision, the adjustment may provide improvements. If that is not the case, then improvements are less likely. Before the adjustment, all classifiers were predicting everything to be **in** for Alice probes. Classifiers like NB and MLP apparently did not change how often they predict **in** even after the normalization, whereas classifiers like P and DT did. In a real scenario this BLEU difference can be reasonably estimated by Bob, since he can use Alice's translation API to calculate the BLEU score on a held-out set, and compare it with his shadow models.

Another possible approach to handle the problem of classifiers always predicting **in** is to consider the relative size of classifier output score. We can rank the samples by the classifier output scores, and decide top N% to be **in** and rest to be **out**. Figure 7 shows how the accuracy changes when varying the **in** percentage. We can see that the accuracy can be much higher than the original result, especially if Bob can adjust the threshold based on his knowledge of **in** percentage in the probe.

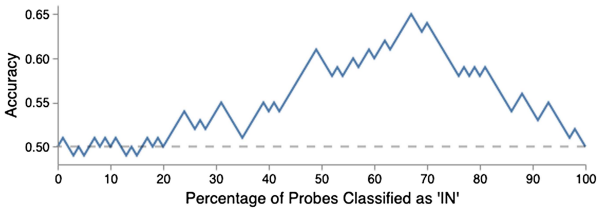


Figure 7: How the attack accuracy on Alice set changes when probe groups are sorted by Perceptron output score and the threshold to classify them as **in** is varied.

This is the first strong general result for Bob, suggesting the membership inference attacks are possible if probes are defined as groups of sentences.[11] Importantly, note that the classifier threshold adjustment is performed only for the classifiers in this section, and is not relevant for the classifiers in Section 6.1 to 6.3.

### 6.5 Attacks Using External Resources

Our results in Section 6.1 demonstrate the difficulty of general membership inference attacks. One natural question is whether attacks can be improved with even stronger features or classifiers, in particular by exploiting external resources beyond the dataset Carol provided to Bob. We tried two different approaches: one using a Quality Estimation model trained on additional data, and another using a neural sequence model with a pre-trained language model.

Quality Estimation (QE) is a task of predicting the quality of a translation at the sentence or word level. One may imagine that a QE model might produce useful feature to tease apart **in** and **out** because **in** translations may have detectable improvements in quality. To train this model, we used the external dataset from the WMT shared task on QE (Specia et al., 2018). Note that for our language pair, German to English, the shared task only had a labeled dataset for the SMT

[11] We can imagine an alternative definition of this group-level membership inference where Bob's goal is to predict the percentage of overlap with respect to Alice's training data. This assumes that model trainers make corpus-level decisions about what data to train on. Reformulation of a binary problem to a regression problem may be useful for some purposes.

| | Alice | Bob:train | Bob:valid | Bob:test |
|---|---|---|---|---|
| P | 50.0 | 49.9 | 50.0 | 50.0 |
| DT | 50.3 | 51.4 | 51.1 | 51.1 |
| NB | 50.4 | 51.2 | 51.1 | 51.0 |
| NN | 49.8 | 66.1 | 50.0 | 50.1 |
| MLP | 50.4 | 51.0 | 51.0 | 50.8 |
| BERT | 50.0 | 50.0 | 50.0 | 50.0 |

Table 7: Membership inference accuracies for classifiers with Quality Estimation sentence score as an extra feature, and a BERT classifier.

system. Our models are NMT, so the estimation quality may not be optimally matched, but we believe this is the best data available at this time. We applied the Predictor-Estimator (Kim et al., 2017) implemented in an open source QE framework `OpenKiwi` (Kepler et al., 2019). It consists of a predictor that predicts each token of the target sentence given the target context and the source, and estimator that takes features produced by the predictor to estimate the labels; both are made of LSTMs. We used this model as this is one of the best models seen in the shared tasks, and it does not require alignment information. The model metrics on the WMT18 dev set, namely, Pearson's correlation, Mean Average Error, and Root Mean Squared Error for sentence-level scores, are 0.6238, 0.1276, and 0.1745, respectively.

We used the sentence score estimated by the QE model as an extra feature for classifiers described in Section 6.1. The results are shown in Table 7. We can see that this extra feature did not provide any significant influence to the accuracy. In a more detailed analysis, we find that the reason is that our **in** and **out** probes both contain a range of translations from low to high quality translations, and our QE model may not be sufficiently fine-grained to tease apart any potential differences. In fact, this may be difficult even for a human estimator.

Another approach to exploit external resources is to use a language model pre-trained on a large amount of text. In particular, we used BERT (Devlin et al., 2019), which has shown competitive results in many NLP tasks. We used BERT directly as a classifier, and followed a fine-tuning setup similar to paraphrase detection: For our case the inputs are the English translation and reference sentences, and the output is the binary membership label. This setup is similar to the classifiers we described in Section 5.3, where rather than training Perceptron or Decision Tree on manually defined features, we directly applied BERT-based sequence encoders on the raw sentences.

We fine-tuned the BERT `Base,Cased` English model with Bob:train. The results are shown in Table 7. Similar to previous results, the accuracy is 50% so the attack using BERT as classifier was not successful. Detailed examination of the BERT classifier probabilities show that they are scattered around 0.5 for all cases, but in general are quite random for both Bob and Alice probes. This result is similar to the other simpler classifiers in Section 6.1.

In summary, from these results we can see that even with external resources and more complex classifiers, sentence-level attack is still very difficult for Bob. We believe this attests to the inherent difficulty of the sentence-level membership inference problem.

## 7 Discussions and Conclusions

We formalized the problem of membership inference attacks on sequence generation tasks, and used machine translation as an example to investigate the feasibility of a privacy attack.

Our results in Section 6.1 and Section 6.5 show that Alice is generally safe and it is difficult for Bob to infer the sentence-level membership. In contrast to attacks on *standard classification* problems (Shokri et al., 2017), *sequence generation* problems maybe be harder to attack because the input and output spaces are far larger and complex, making it difficult to determine the quality of the model output or how confident the model is. Also, the output distribution of class labels is an effective feature for the attacker for standard classification problems, but is difficult to exploit in the sequence case.

However, this does not mean that Alice has no risk of leaking private information. Our analyses in Sections 6.2 and 6.3 show that Bob's accuracy on out-of-domain and out-of-vocabulary data is above chance, suggesting that attacks may be feasible in conditions where unseen words and domains cause the model to behave differently. Further, Section 6.4 shows that for a looser definition of membership attack on groups of sentences, the attacker can win at a level above chance.

Our attack approach was a simple one, using shadow models to mimic the target model. Bob can attempt more complex strategies, for example, by using the translation API multiple times per sentence. Bob can manipulate a sentence, for example, by dropping or adding words, and observe how the translation changes. We may also use the metrics proposed by Carlini et al. (2018) as features for Bob; they show how recurrent models might unintentionally memorize rare sequences in the training data, and propose a method to detect it. Bob can also add ''watermark sentences'' that have some distinguishable characteristics to influence the Alice model, making attack easier. To guard against these attacks, Alice's protection strategy may include random subsampling of training data or additional regularization terms.

Finally, we note some important caveats when interpreting our conclusions. The translation quality of the Alice and Bob MT models turned out to be similar in terms of BLEU. This situation favors Bob, but in practice Bob is not guaranteed to be able to create shadow models of the same standard, nor verify how well it performs compared with the Alice model. We stress that when one is to interpret the results, one must evaluate both on Bob's test set and Alice probes side-by-side, like those shown in Tables 2, 3, and 7, to account for the fact that Bob's attack on his own shadow model translations is likely an optimistic upper-bound on the real attack accuracy on Alice's model.

We believe our dataset and analysis is a good starting point for research in these privacy questions. Although we focused on MT, the formulation is applicable to other kinds of sequence generation models such as text summarization and video captioning; these will be interesting as future work.

## Acknowledgments

The authors thank the anonymous reviewers and the action editor, Colin Cherry, for their comments.

## References

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Xiaodong Song. 2018. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *CoRR*, abs/1802.08232.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kevin Duh. 2018. The multitarget TED talks task. `http://www.cs.jhu.edu/~kevinduh/a/multitarget-tedtalks/`.

Cynthia Dwork. 2008. Differential privacy: A survey of results. Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li, editors, In *Theory and Applications of Models of*

*Computation*, pages 1–19. Springer Berlin Heidelberg.

Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2017. LOGAN: Evaluating privacy leakage of generative models using generative adversarial networks. *CoRR*, abs/1705.07663.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *CoRR*, abs/1712.05690.

Marcin Junczys-Dowmunt. 2018. Dual conditional cross-entropy filtering of noisy parallel corpora. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 888–895, Belgium, Brussels. Association for Computational Linguistics.

Fabio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, and André F. T. Martins. 2019. OpenKiwi: An open source framework for quality estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 117–122, Florence, Italy. Association for Computational Linguistics.

Hyun Kim, Jong-Hyeok Lee, and Seung-Hoon Na. 2017. Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 562–568, Copenhagen, Denmark. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 605–612, Barcelona, Spain.

Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A. Gunter, and Kai Chen. 2018. Understanding membership inferences on well-generalized learning models. *CoRR*, abs/1802.04889.

Ashwin Machanavajjhala, Xi He, and Michael Hay. 2017. Differential privacy in the wild: A tutorial on current practices & open challenges. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 1727–1730, New York, NY. ACM.

Milad Nasr, Reza Shokri, and Amir Houmansadr. 2018. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 634–646, New York, NY. ACM.

Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. 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 (SP)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002, Jul. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA. Association for Computational Linguistics.

Matt Post. 2018, Oct. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. 2018. Knock knock, who's there? Membership inference on aggregate location data. *CoRR*, abs/1708.06145.

Md Atiqur Rahman, Tanzila Rahman, Robert Laganiere, Noman Mohammed, and Yang Wang. 2018. Membership inference attack against differentially private deep learning model. *Transactions on Data Privacy*, 11:61–79.

Ahmed Salem, Yonghui Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. 2018. Ml-leaks: Model and data independent membership

inference attacks and defenses on machine learning models. *CoRR*, abs/1806.01246.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

R. Shokri, M. Stronati, C. Song, and V. Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18.

Congzheng Song and Vitaly Shmatikov. 2019. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 196–206. New York, NY, ACM.

Lucia Specia, Varvara Logacheva, Frederic Blain, Ramon Fernandez, and André Martins. 2018. WMT18 quality estimation shared task training and development data. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. 2018. Towards demystifying membership inference attacks. *CoRR*, abs/1807.09173.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Michael Veale, Reuben Binns, and Lilian Edwards. 2018. Algorithms that remember: Model inversion attacks and data protection law. *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences*, 376.

Samuel Yeom, Matt Fredrikson, and Somesh Jha. 2017. The unintended consequences of overfitting: Training data inference attacks. *CoRR*, abs/1709.01604.