Private Outsourced Translation for Medical Data

Travis Morrison
Institute for Quantum Computing
University of Waterloo

Bijeeta Pal
Department of Computer Science
Cornell University

Sarah Scheffler
Department of Computer Science
Boston University

Alexander Viand*
Department of Computer Science
ETH Zurich

Abstract

Overcoming language barriers is a key challenge for international organizations providing medical aid in a variety of places. While bilingual doctors or human translators can achieve this effectively, many volunteer clinics are under-staffed and under-funded and must do without. Machine translations have become increasingly accurate and accessible, e.g. via Google Translate. However, uploading medical data to the cloud can violate patient privacy, while offline translation systems are often less accurate. Using homomorphic encryption, clients could submit encrypted queries to a server hosting a translation model without revealing the underlying private information. However, modern translation systems are based on Recurrent Neural Networks (RNNs) which are challenging to evaluate homomorphically due to their inherent high depth and their heavy reliance on non-linearity. We design, implement, and evaluate a proof-of-concept solution and explore a variety of solutions to these challenges.

Introduction

Overcoming language barriers remains a key challenge for aid organisations working globally. In the case of medical aid, effective communication is required not just for efficient logistics and administration, but is of vital importance to the main mission. Doctors volunteering abroad need to understand patients' records in order to make correct diagnoses and select appropriate treatments. After many in-clinic treatments, patients must follow specific drug or care regimens. These instructions must be communicated effectively to ensure a positive outcome.

Overcoming these language barriers can place considerable strain on the resources of these organisations. While bilingual doctors can facilitate the translation of medical records, prescriptions, and instructions with high accuracy, not all doctors volunteering abroad are proficient in their patients' languages. Meanwhile, professional translators are a costly resource and volunteer clinics are often under-staffed and under-funded to begin with. Therefore, efficient and effective automated solutions for medical translation are needed to lighten the translation workload.

Automated translation has become increasingly accurate due to the development of neural-network-based machine learning approaches. Cloud-based services like Google Translate offer fast and accurate translations of documents and speech. However, uploading medical data to the cloud can violate patient privacy. Meanwhile, offline translation systems are often less accurate than state-of-the-art cloud based solutions. In the case of medical translation services, there could also be concerns about releasing models trained on private medical data.

Homomorphic encryption (HE) could allow users to outsource translation to a server without revealing the underlying information. A service provider would train and maintains a machine-learning model, likely derived from private medical data, and make the translation service based on it avail-

*Corresponding author.

able to the clients. The client then sends an encrypted query to the server, which homomorphically evaluates the model on the query and returns the encrypted result.

A Machine-Learning-as-a-Service (MLaaS) provider might donate the required server resources to charities, or public institutions like hospitals or universities could run such a platform, which would be many orders of magnitude cheaper than human translators. We imagine that in this setting, client devices belong to the doctors, pharmacist, or clinics rather than individual patients. Since patients already trust their doctors with their private data, this offers significant deployment benefits with little practical impact on the privacy guarantees.

While much prior work considers homomorphically encrypted machine learning (e.g., [GBDL$^+$16, JVC18]), natural language processing tasks like translation generally make use of Recurrent Neural Networks (RNNs) which offer unique challenges. Due to their recurrent nature, they inherently have a high depth of computation. Convolutional Neural Networks (CNNs), in contrast, are generally 'wide' rather than deep. The high-cost of evaluating deep circuits in levelled FHE is usually compensated by fully utilizing *batching* to amortize costs across many parallel computations. However, the 'narrow-but-deep' layout of an RNN computation makes it challenging to fully exploit this common technique. In addition, RNNs rely heavily on complex non-linear activation functions. While all neural network architectures tend to feature non-linear-activation functions, it has been shown [GBDL$^+$16] that low-degree polynomial functions can achieve the desired effects in lower-depth CNNs. RNNs, however, require activation functions that 'clamp' values to a fixed interval in order to avoid numerical issues. Such functions are inherently nearly impossible to emulate with low-degree polynomials.

Therefore, implementing RNN-based machine translation using homomorphic encryption is a highly non-trivial task. In order to set a feasible goal, we consider prescriptions, specifically the directions on when and how to take the drug, as a first step. These texts are suitable for an initial design since they are generally a single short phrase (e.g., "one capsule every eight hours"), frequently contain vocabulary that is otherwise rarely used (e.g., "Apply one inhalation...") yet feature a small overall vocabulary. In addition, relaying the information in these texts is both vitally important and surprisingly challenging. A large number of studies have shown how complex language or unclear instructions lead to patient misunderstandings [SBM18].

The need for clear effective translations of prescription instructions is also evidenced by the development of standardised translation tables [AHRQ14] that provide a mapping from common phrases (e.g., "Take one pill at bedtime") into several languages. While these represent an important first step, they cover only a very limited number of prescription instructions, cannot accommodate additional explanations and only work uni-directionally.

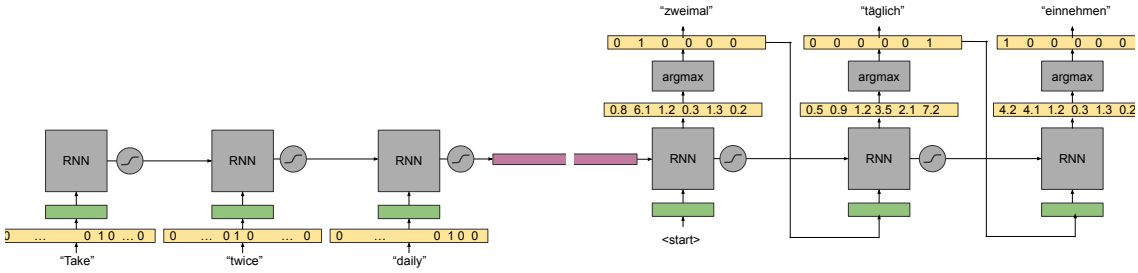
We therefore propose a solution for private outsourced translation for medical texts, specifically prescription instructions, using homomorphic encryption. We focus on the concise texts found in prescription instructions, allowing us to showcase a feasible proof-of-concept implementation using existing tools. We evaluate our prototype and explore how the remaining challenges might be overcome. Finally we propose avenues for future work in this area.

Machine Translation

Modern approaches to machine translation are frequently based on neural networks, specifically Recurrent Neural Networks (RNNs). In contrast to feed-forward networks, RNNs can process sequences of inputs, incorporating information from previous parts of the input in the decision process. This makes them especially suited to natural language processing tasks.

For text-based tasks like translation, words are represented by their index in a fixed-size dictionary, i.e. a list of the k most relevant words for the task, with typical sizes for k starting around 5000 words. Chunks of the one-hot encoded phrase are then converted into *embeddings* in, e.g., $\mathbb{R}^w$ using a simple, non-task-specific, model. These chunks, rather than the individual words, make up the input sequence for the RNN.

Generating a translation with an RNN consists of two stages. First the input sequence of chunks is run through an encoding phase, which generates a fixed-length *hidden representation* of the



(a) Encoding (b) Decoding

Figure 1: Encoding and decoding stages of the model

sequence. This allows easy training of the model with sentences of varying lengths. Second, this representation is used as the input for the decoding phase, which generates the translation output.

The encoding and decoding phase each consist of a single unit, that could be either a simple "fully connected" layer or a more complex architecture such as GRUs [CvMG+14] or LSTMs [HS97]. Each unit features at least one source of *non-linearity*. This is most commonly the `tanh` activation function which has bounded outputs, preventing numerical issues that arise due to the recurrent nature of RNNs. In each phase, the current input from the sequence and the output from the previous step are fed into the unit, as shown in Figure 1. This allows the model to incorporate information from previous parts of the sequence, but also leads to a very deep computation graph, which makes a homomorphic encryption implementation challenging.

While plaintext-based solutions often err on the side of larger hidden representations, we experimented with a simple RNN model in the clear to determine what an acceptable value of w would be; the results are shown in Figure 2.

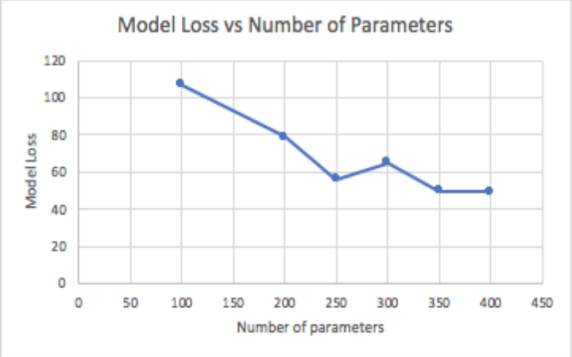


Figure 2: ML Model parameters to determine acceptable value of w.

Design

We assume that the client, i.e. the doctor or clinic, posses a symmetric secret key for the CKKS scheme [CKKS17]. The server, meanwhile, has access to the corresponding public relinearization and rotation keys required to evaluate the computation. For convenience of deployment, these might be set-up before a doctor departs his home country or at a major city in the destination country, where high-speed internet is available.

Given a phrase to translate, the client software first tokenizes the phrase, representing each word as a one-hot encoding vector referring to a fixed-length dictionary. Should a word not be present in the dictionary, it is usually assigned a special "unknown" token when evaluating translation models. However, in a practical deployment setup it might be more suitable to notify the user and give

them a chance to provide an alternative word. In our design, we choose a dictionary size of 5000 words, which is on the smaller side for general language tasks but still common. Since the language of prescription instructions is fairly standardized, we would expect unknown words to occur with very small frequency.

Chunks of the phrase are then embedded into the hidden space, in our case $\mathbb{R}^{256}$ as this provides a good balance between accuracy and model size (See Fig. 2). This embedding is performed on-device, using a simple pre-trained lookup table. The list of embedded chunks, $x_0, x_1, \ldots, x_n \in \mathbb{R}^{256}$ is the input to the FHE computation. The client encrypts the chunks using the symmetric secret key, batching them into a single ciphertext which is then sent to the server.

The server then evaluates the RNN on the input. For simplicity, we consider a fully recurrent neural network, i.e. for each element x_i in the input sequence, we have

$$h_i = g(W_x x_i + W_h h_{i-1} + b)$$

where $W_x, W_h \in \mathbb{R}^{256x256}$ are weight matrices, b is a bias vector, and $g : \mathbb{R}^{256} \to \mathbb{R}^{256}$ is a non-linear activation function. In the decoding phase, the weight matrices are of size $\mathbb{R}^{512x256}$ and the resulting vector is split into h_t and y_t. To derive the actual output from $y_t \in \mathbb{R}^{256}$, we find the dictionary entry that has the embedding vector that is closest to y_t (`argmax`). This model architecture can be seen in Fig. 1.

In an ideal setting, the server would evaluate the whole model under FHE and return the encrypted result to the client, once again batched into a single ciphertext for communication efficiency. However, there are several challenges that make a straightforward evaluation of the network infeasible.

Challenges One challenge of a homomorphic encryption implementation of an RNN is the sequential nature of the architecture. A long input sentence will result in a high multiplicative depth of the computation, requiring larger parameters and therefore slower computations. Another challenge are the required non-polynomial functions. These include the non-linear activation functions, which are also present in traditional feed-forward networks. However, in RNNs they are considerably more important since the deep nature of the computation can quickly lead to numerical issues if unsuitable activation functions are chosen. More importantly, however, the decoder requires the repeated evaluation of the `argmax` function. While many neural networks use functions to like `max`, `softmax` or even `argmax`, these are generally applied at the very end of the computation and can therefore be left to the client to perform after decryption. In the RNN decoder architecture, however, the output of the argmax needs to be fed into the next stage of the computation.

We explored a variety of possible solutions to these issues during the course of this project:

(i) Polynomial Approximation: In previous work on neural networks in FHE, non-linear activation functions like RELu have been approximated with varying low-degree polynomials [GBDL+16]. In more general purpose computations, using Chebyshev polynomials to approximate continuous functions on an interval is a standard techniques in HE. However, due to the deep nature of RNNs, using standard approximations can lead to significant accuracy issues.

More importantly, existing techniques do not admit a straightforward method for approximating `argmax` which is a multivariate function. Using the method of [CKK19] for computing a maximum, it might be possible to compute the `argmax` by performing a linear number of `max` operations over the output vector. While this `max` operation is computationally costly, all the costs would be incurred at the server, not the computationally limited client device.

(ii) Binary Representation: Changing to a binary representation would allow us to compute the non-linear functions directly. Here, it would be best to use a scheme and implementation optimized for this setting (e.g. TFHE [CGGI16])). The activation functions can be implemented via lookup tables at the desired accuracy, while the `argmax` could be implemented directly. However, in this setting matrix-vector multiplications become prohibitively expensive. In fact, the only existing FHE implementation of a RNN that we are aware of [LJ19] actually uses the binary setting, but uses an extremely quantized network with 4 bit weights to avoid this multiplication overhead. At such high quantization levels, more complex machine learning tasks like machine translation are likely to lose too much accuracy to be of practical use. In essence, we would only be trading the challenge of polynomial approximation with the challenge of extreme quantization.

(iii) Scheme Switching: Some other frameworks have proposed switching between different representations or schemes in order to improve performance. Glyph [LFFJ19] and Chimera [BGGJ18] both switch to Fully Homomorphic Encryption on the Torus (TFHE) [CGGI16] for non-linear computations such as `softmax`. While Chimera seems to have been implemented for the i-DASH 2019 competition, the implementation is not yet publicly available.

On a conceptual level, it seems straight-forward to switch from CKKS to TFHE to e.g. evaluate a non-linearity activation function after using CKKS for matrix-vector multiplications. However, while Chimera also introduces transitions from TFHE to CKKS, these transitions produce very specific CKKS ciphertexts that contain not the original message but an exponentiated form. It is not obvious to us at this point whether or not it would be possible to "complete the circle" and convert a TFHE ciphertext back into a CKKS ciphertext that would be suitable for continuing the evaluation of the network. Even if it is not possible to switch back-and-forth in a straight-forward way, it might be possible to rephrase the entire computation from scratch in a way that can take advantage of the power of both schemes.

(iv) Client Interaction: Finally, the most simple solution is to send each individual unit's output back to the client, who will decrypt it, compute the required non-linear functions including activation functions and `argmax`, and send it back to the server. While this option introduces several additional rounds of communication and significantly increases the communication overhead, it is simple to implement and makes each individual unit a very low-depth and efficient circuit.

For the encoding phase, we use low-degree polynomial approximations. For the input sizes common for prescription instructions, the circuit depth – while high – is not entirely prohibitive. While this requires more complex training procedures, it has been shown that e.g. replacing `tanh` with `ReLU` in RNNs leads to only slightly lower performance [LJH15]. Hopefully, with significant adjustments to the training, polynomial activation functions could be shown to have acceptable performance, too. For the decoding phase, where `argmax` is required, we consider client interaction to be most feasible for an initial design. However, the authors want to continue exploring the feasibility of scheme switching for a fully-outsourced RNN implementation.

Implementation & Evaluation

We developed a proof-of-concept implementation[1] using the Micorosft SEAL library [SEA19], showing the feasibility of evaluating a sequence of RNN units. The core of the computation is made up of matrix-vector products between the plaintext weight matrices of the model and the (encrypted) input or hidden-representation vectors. We approximate the non-linear activation functions during the encoding phase with $g(x) = x^2$.

Encoding Choosing the right *batching* layout is essential for an efficient implementation. CKKS ciphertexts of lattice dimension n can hold up to $n/2$ independent values in (virtua) *slots*. Arithmetic operations apply component-wise, i.e. in a SIMD fashion. Special automorphisms can can be used to *rotate* the elements between slots cyclically.

In order to optimize the matrix-vector products, we use the "diagonal method" [HS14], where we encode the matrices not row- or column-wise but instead encode the diagonals. This allows us to compute the matrix-vector-product between a matrix of dimension kxk and a vector of length k with only $k-1$ rotations, k component-wise multiplications and $k-1$ component wise additions.

However, since rotations on the slots are cyclical, naively encoding the values produces correct results only if $k = n/2$. In addition, we want to encode all inputs x_i (of length $k = 256$) into a single ciphertext (with $n >= 16348$) in order to minimize the communication overhead. Since the diagonal method only requires rotations in a single direction, and by at most $k-1$, we choose to simply encode each vector twice. While this does require more slots, we already need to choose n very large to accommodate the depth of the computation, therefore we can still easily fit many duplicated input vectors into the same vector.

[1] Available at github.com/PrivateAI-Group1/SEAL

Optimizations While the diagonal method already requires a relatively small number of rotations, this can be improved further by using a baby-step–giant-step approach [Che19]. This relies upon the fact that we can split each rotation into two separate rotations and only perform the second rotation *after* aggregating vectors that require the same rotation. For a rotation of l steps, we can decompose l into $k * n_1 + j$ for $n = n_1 * n_2$ and $0 \leq k \leq n_1$, $0 \leq j \leq n_2$. We first store copies of the vector rotated by each possible j. For each k, we compute the component-wise multiplication between each of the pre-rotated vectors and the corresponding matrix diagonal. Then, we add all n_2 of these products together and rotate the resulting vector by $k * n_1$ steps. Note that this requires pre-rotating each matrix diagonal $k * n_2 + j$ steps in the *opposite* direction prior to multiplication, to account for the second rotation. However, since the matrices are available as plaintext, this cost is negligible.

The client must provide the server with the necessary Galois keys to perform the ciphertext rotations. Even though this is a one-time setup, we nevertheless want to minimize the size of these keys. During the computation, we need to rotate the vector by up to $k-1$ steps, however choosing all $k-1$ keys would be very space-inefficient. By default, SEAL already picks rotation keys corresponding to steps by 2^i and -2^i for $0 \leq i \leq \log n/2$, and rotations are assembled from the Non-Adjacent-Form decomposition of the number of steps required, which minimizes the number of rotations required. However, since we never need to rotate all the way to $n/2 - 1$, we choose only the powers required to reach k. Considering the decomposition of the rotations in the baby-step–giant-step approach might allow us to select even fewer rotation keys, but even with the current approach we can already reduce the key size from 247 MB to 152 MB for $n = 16348$.

Results We evaluated the performance of our implementation on a standard desktop computer with an Intel i7-8700 CPU (6 cores, up to 4.60 GHz) and 32 GB of RAM. Using CKKS as implemented in Microsoft Seal v 3.4.5, we set $n = 32768$, a coefficient modulus with 880 bits and a scale of 2^{40}. We evaluated the encoding phase of the network, using x^2 as the activation function. While this naturally led to a significant blow-up in values (even when rescaling appropriately), we considered this a good test case to demonstrate that even reasonably deep circuits can be practical. The ciphertext transmitted to the server was 3.8 MB in size, and for five RNN units, the computation took a total of 90 seconds. In the context of a clinic appointment, latencies in the order of minutes seem more than acceptable, making this initial result a promising start.

Discussion

We have introduced our design and prototype for privacy-preserving outsourced translation of medical data. By focusing on short, formulaic prescription instructions we have picked an application area where FHE could feasibly be deployed in the near future. At the same time, existing makeshift solutions show the need for translation in this domain and medical surveys confirm the importance of understandable prescription instructions for positive treatment outcomes. Our initial implementation shows the feasibility of such a system and features a variety of optimizations to implement the underlying computations efficiently. The major challenge going forward is the inability to efficiently evaluate more complex activation functions and the `argmax` function in CKKS using standard FHE techniques. We consider future work in this area, especially investigating how to apply scheme switching techniques, to be of independent interest and will continue to explore in this direction.

References

[AHRQ14] Agency for Healthcare Research and Quality. Explicit and standardized prescription medicine instructions, Dec. 2014. URL https://www.ahrq.gov/health-literacy/pharmhealthlit/prescriptionmed-instr.html.

[BGGJ18] C. Boura, N. Gama, M. Georgieva, and D. Jetchev. Chimera: Combining ring-lwe-based fully homomorphic encryption schemes. Cryptology ePrint Archive, Report 2018/758, 2018. https://eprint.iacr.org/2018/758.

[CGGI16] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 3–33. Springer, 2016.

[Che19] H. Chen. Techniques in privacy-preserving machine learning. https://github.com/WeiDaiWD/Private-AI-Bootcamp-Materials/blob/master/4_Hao_Techniques_in_PPML.pdf, 2019.

[CKK19] J. H. Cheon, D. Kim, and D. Kim. Efficient homomorphic comparison methods with optimal complexity. Cryptology ePrint Archive, Report 2019/1234, 2019. https://eprint.iacr.org/2019/1234.

[CKKS17] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

[CvMG+14] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar*, pages 1724–1734, 2014. URL https://www.aclweb.org/anthology/D14-1179.pdf.

[GBDL+16] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 201–210, New York, New York, USA, 2016. PMLR. URL http://proceedings.mlr.press/v48/gilad-bachrach16.html.

[HS97] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HS14] S. Halevi and V. Shoup. Algorithms in HElib. In *Advances in Cryptology – CRYPTO 2014*, pages 554–571. Springer Berlin Heidelberg, 2014. doi:10.1007/978-3-662-44371-2_31.

[JVC18] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1651–1669, 2018. URL https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-juvekar.pdf.

[LFFJ19] Q. Lou, B. Feng, G. C. Fox, and L. Jiang. Glyph: Fast and accurately training deep neural networks on encrypted data. *arXiv preprint arXiv:1911.07101*, 2019.

[LJ19] Q. Lou and L. Jiang. SHE: A fast and accurate deep neural network for encrypted data. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alche-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 10035–10043. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9194-she-a-fast-and-accurate-deep-neural-network-for-encrypted-data.pdf.

[LJH15] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. apr 2015, 1504.00941. URL `http://arxiv.org/abs/1504.00941`.

[SBM18] N. Samaranayake, W. Bandara, and C. Manchanayake. A narrative review on do's and don'ts in prescription label writing – lessons for pharmacists. *Integrated Pharmacy Research and Practice*, Volume 7:53–66, June 2018. doi:10.2147/iprp.s163968.

[SEA19] Microsoft SEAL (release 3.4). `https://github.com/Microsoft/SEAL`, Oct. 2019. Microsoft Research, Redmond, WA.