Gradient Information From Google GBoard NWP LSTM Is Sufficient to Reconstruct Words Typed

Mohamed Suliman, Douglas J. Leith
Trinity College Dublin, Ireland

***Abstract*—Federated Learning is now widely deployed by Google on Android handsets for distributed training of neural networks. While Federated Learning aims to avoid sharing sensitive user data with Google, in this paper we show that when used for GBoard next word prediction Federated Learning provides little privacy to users. Namely, we demonstrate that the words typed by a user can be quickly and accurately reconstructed from the gradients of the GBoard LSTM used for next word prediction. Use of mini-batches does not protect against reconstruction.**

I. Introduction

In [1] Google introduced Federated Learning for privacy-enhanced distributed training of neural networks. Federated Learning is now widely deployed on Android mobile handsets, and in particular is used for next word prediction in Google's GBoard keyboard app [2] which, according to the Google Play store, is installed in more than 1 Billion devices[1]. In Federated Learning a central server collects gradient vectors from mobile handsets running the model, it executes a stochastic gradient descent step to update the model parameters and then pushes the new parameter values to the mobile handsets. This process repeats until the parameters are judged to have converged, a specified number of iterations have been completed etc. By keeping the training data on the mobile handsets and only sharing gradient information, the hope is that a degree of privacy is gained. However, there has been little formal privacy analysis of Federated Learning.

In this paper we show that when used for GBoard next word prediction Federated Learning provides little privacy to users. Namely, we demonstrate that the words typed by a user can be quickly and accurately reconstructed from the gradients of the GBoard LSTM used for next word prediction. Use of mini-batches does not protect against reconstruction.

Key to the lack of privacy is that in next word prediction the neural net input is echoed by the neural net output. That is, in the next word prediction task the output of the neural net aims to match the sequence of words typed by the user, albeit with a shift one word ahead. The sign of the output loss gradient directly reveals information about the words typed by the user, which can then be reconstructed by inspection (there is no need for any complex processing).

We provide more details below, but briefly illustrate the nature of the information leakage here. Let D be a dictionary of V words, each typed word is mapped to an entry in D and the next word prediction is a vector $\hat{y} \in [0,1]^V$ whose i'th element $\hat{y}_i$ is the probability that the next word will be the i'th dictionary entry. When a softmax output layer is used $\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{V} e^{z_j}}$ with vector $z \in \mathbb{R}^V$ the raw logits. Suppose the next word typed by the user is entry i^* from the dictionary. The cross-entropy loss is $J = -\log \frac{e^{z_{i^*}}}{\sum_{j=1}^{V} e^{z_j}}$. The derivative $\frac{\partial J}{\partial z_{i^*}}$ is $\frac{e^{z_{i^*}}}{\sum_{j=1}^{V} e^{z_j}} - 1 < 0$ while for $i \neq i^*$ the derivative is $\frac{\partial J}{\partial z_i} = \frac{e^{z_i}}{\sum_{j=1}^{V} e^{z_j}} > 0$. Hence, we can infer the index i^* of the word typed by the user simply by inspecting the sign of the loss derivatives. This is illustrated schematically in Figure 1. Federated Learning shares the derivatives of the loss with respect to model parameters, rather than the derivatives with respect to the logits y_i, but the derivatives of parameters in the penultimate output layer are enough.

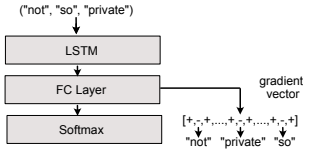

Fig. 1. Illustrating how sign of gradient elements can leak words typed.

Previous work on information leakage by Federated Learning has mainly focused on object detection tasks, e.g. see [3], [4], [5]. The neural network output is an object label (e.g. cat, dog) and the attack aims to reconstruct the input image only from gradient information. The attack is typically formulated as an optimisation problem and solved using gradient descent. Successful attacks have been demonstrated for standard neural nets and image datasets. However, because in next word prediction the neural net input is echoed by the output we are able to perform much faster, more robust attacks that are difficult to defend against. We demonstrate the effectiveness of these attacks against a widely deployed LSTM neural net from GBoard.

II. Privacy Threat Model

The transmission of user data from mobile handsets to back-end servers is not intrinsically a breach of privacy. For instance, it can be useful to share details of the device model/version and the locale/country of the device when checking for software updates. This poses few privacy risks if the data is common to many handsets and therefore cannot be easily linked back to a specific handset/person [6], [7].

Two major issues in handset privacy are (i) release of sensitive data, and (ii) de-anonymisation i.e. linking of data to a person's real world identity.

Release of sensitive data. What counts as sensitive data is a moving target, but it seems clear that the words entered by users, e.g. when typing messages, writing notes and emails, web browsing and performing searches, may well be sensitive. It is not just the sentences typed which can be sensitive but also just the list of words used (i.e. even without knowing the word ordering) since this can be used for targeting surveillance via keyword blacklists [8]. It is also important to note that data which is not sensitive in isolation can become sensitive when combined with other data, and this is a particular concern with regard to large companies that operate mobile payment services, supply web browsers, run advertising platforms etc.

De-anonymisation. Android handsets can be directly tied to a person's real identity in several ways, even when a user

This work was supported by SFI grant 16/IA/4610.

[1]https://play.google.com/store/apps/details?id=com.google.android.inputmethod.latin, accessed 17th Feb 2022.

takes active steps to try to preserve their privacy. Probably most relevant here is via the Android ID, since most Google telemetry is tagged with this. Via other data collected by Google Play Services the Android ID is linked to (i) the handset hardware serial number, (ii) the SIM IMEI (which uniquely identifies the SIM slot) and (iii) the user's Google account [9], [10]. When creating a Google account it is necessary to supply a phone number on which a verification text can be received. For many people this will be their own phone number. Use of Google services such as buying a paid app on the Google Play store or using Google Pay further links a person's Google account to their credit card/bank details. A user's Google account, and so the Android ID, can therefore commonly be expected to be linked to the person's real identity.

III. Related Work

The use of Federated Learning for next word prediction in Google's GBoard app is considered in [2] and the structure of the LSTM that we extracted from the app matches the description in that paper. The Long Short Term Memory (LSTM) recurrent neural network was introduced in [11], and a variant termed the Coupled Input Forget Gate (CIFG) LSTM was developed in [12]. CIFGs include less parameters that need to be trained than a regular LSTM, and are thus appealing for mobile handset deployment, where network bandwidth and storage resources cannot be guaranteed.

Since being introduced by [1], Federated Learning has attracted a great deal of interest and generated a growing body of literature, in particular about the security challenges it poses [13], [14], [15]. The attack presented herein can be classed as an *inference attack*, where training inputs and labels are inferred from the model updates. Information leakage from the gradients of neural nets used for object detection appears to have been initially investigated in [3], which proposed a so-called Deep Leakage from Gradients (DLG) method for input image reconstruction. This work was subsequently extended by [5], [4]. In particular, [4] demonstrated effective input image reconstruction even for mini-batch sizes up to 48 images. Other inference attacks attempt to infer a particular data point's membership in the training data [16], [17], [18], as well properties of other participants' training data [17].

IV. GBoard Next Word Prediction LSTM

A. LSTM Software Version & Tensorflow Implementation

We used a rooted Google Pixel 2 running Android 11 and Google GBoard app version 10.5.03.367007960. We extracted the files nwp.csym, nwp.csym2 and nwp.uint8.mmap.tflite from folder files/superpacks/next-word-predictor/tflite-nwp-45c6579035e9df4ffe5c1246f8d5615d within the app private data directory. The file nwp.csym2 contains the LSTM dictionary containing $V = 9502$ words. The dictionary is stored in MARISA-Trie format[2] with an additional pre-pended file header. The file nwp.uint8.mmap.tflite is the LSTM, stored in TensorFlow Lite format[3]. The publicly available version of TensorFlow Lite does not support training or the calculation of model derivatives. We therefore ported the TensorFlow Lite model to TensorFlow, verifying that both generated identical outputs for the same inputs. The weights in the TensorFlow Lite model are tagged with descriptive labels, and are used to create the replicated model in TensorFlow.

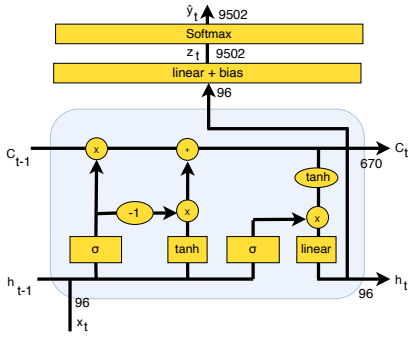

Fig. 2. Schematic of LSTM architecture. LSTM layer takes as input dense vector x_t representing a typed word and outputs a dense vector h_t. This output is then mapped to vector z_t of size 9502 (the size of the dictionary) with the value of each element being the raw logit for the corresponding dictionary word. A softmax layer then normalises the raw z_t values to give a vector $\hat{y}_t$ of probabilities.

B. LSTM Architecture

Input words are first mapped to a dictionary entry, with a special <UNK> entry used for words that are not in the dictionary. The index of the dictionary entry is then mapped to a dense vector of size $D = 96$ using a lookup table (the dictionary entry is one-hot encoded and then multiplied by an $\mathbb{R}^{D\times V}$ weighting matrix W^T) and applied as input to an LSTM layer with 670 units i.e. the state C_t is a vector of size 670. The LSTM layer uses a CIFG architecture without peephole connections, illustrated schematically in Figure 2. The LSTM state C_t is linearly projected down to an output vector h_t of size D, which is mapped to a raw logit vector z_t of size V via a weighting matrix W and bias b. This extra linear projection is not part of the orthodox CIFG cell structure, and is included to accommodate the model's tied input and output embedding matrices [19]. A softmax output layer finally maps this to an $[0, 1]^V$ vector $\hat{y}_t$ of probabilities, the i'th element being the estimated probability that the next word is the i'th dictionary entry.

C. Loss Function

Following [2] we use categorical cross entropy loss over the output and target labels.

V. The Attack: Gradient Information Leakage

A. Recovering The Words Typed

After the user has typed t words the output of the neural net is next word prediction vector $\hat{y}_t$,

$$\hat{y}_{t,i} = \frac{e^{z_i}}{\sum_{j=1}^{V} e^{z_j}},\ i = 1, \ldots, V$$

with raw logit vector $z_t = Wh_t + b$, where h_t is the output of the LSTM layer. The cross-entropy loss function for text consisting of T words is $J_{1:T}(\theta) = \sum_{t=1}^{T} J_t(\theta)$ where

$$J_t(\theta) = -\log \frac{e^{z_{i^*}(\theta)}}{\sum_{j=1}^{V} e^{z_j(\theta)}}$$

where i_t^* is the dictionary index of the t'th word entered by the user and θ is the vector of neural net parameters (including

[2]See, for example, https://android.googlesource.com/platform/external/marisa-trie/

[3]https://www.tensorflow.org/lite/

the elements of W and b). Differentiating with respect to the output bias parameters b we have that,

$$\frac{\partial J_{1:T}}{\partial b_k} = \sum_{t=1}^{T} \sum_{i=1}^{V} \frac{\partial J_t}{\partial z_{t,i}} \frac{\partial z_{t,i}}{\partial b_k}$$

where

$$\frac{\partial J_t}{\partial z_{t,i_t^*}} = \frac{e^{z_{i^*}}}{\sum_{j=1}^{V} e^{z_j}} - 1 < 0 \quad \frac{\partial J_t}{\partial z_{t,i}} = \frac{e^{z_i}}{\sum_{j=1}^{V} e^{z_j}} > 0, \ i \neq i_t^*$$

and

$$\frac{\partial z_{t,i}}{\partial b_k} = \begin{cases} 1 & k = i \\ 0 & \text{otherwise} \end{cases}$$

That is,

$$\frac{\partial J_{1:T}}{\partial b_k} = \sum_{t=1}^{T} \frac{\partial J_t}{\partial z_{t,k}}$$

It follows that for words k which do not appear in the text $\frac{\partial J_{1:T}}{\partial b_k} > 0$. Also, assuming that the neural net has been trained to have reasonable performance then e^{z_k} will tend to be small for words k that do not appear next and large for words which do. Therefore for words i^* that appear in the text we expect that $\frac{\partial J_{1:T}}{\partial b_{i^*}} < 0$.

Example: Suppose the input to the neural net is the sentence "this online learning is not so private". Calculating the gradients $\frac{\partial J_{1:T}}{\partial b_k}$, $k = 1, \ldots, V$, sorting the values in descending order and selecting the elements with negative values yields the following (k'th word, $\frac{\partial J_{1:T}}{\partial b_k}$) pairs: ("learning",-0.9997006) ("private",-0.9994907) ("online",-0.99600935) ("not",-0.9627077) ("so",-0.94802666) ("is",-0.78382504). All other words in the dictionary have non-negative gradients.

This observation is intuitive from a loss function minimisation perspective. Typically the estimated probability $\hat{y}_{i^*}$ for an input word will be less than 1. Increasing $\hat{y}_{i^*}$ will therefore decrease the loss function i.e. the gradient is negative. Conversely, the estimated probability $\hat{y}_i$ for a word that does not appear in the input will be small but greater than 0. Decreasing $\hat{y}_i$ will therefore decrease the loss function i.e. the gradient is positive.

While we focus on the bias parameters b here since they yield particularly simple expressions, similar analysis applies to the W parameters and can also be expected to apply to other forms of penultimate output layer. The key is that because the output $\hat{y}_t$ aims to echo the words typed by the user, the gradient of the loss with respect to parameters in the penultimate layer will always tend to directly reveal information about the words typed (unlike in the case of object detection where the neural net output is just the object label and so reconstruction of the full input image is an additional, challenging, step).

B. Recovering The Sentences Typed

The approach above extracts the set of words typed by inspection, however this gives no indication of the original ordering of words so as to reconstruct the original sentences typed. For mini batches consisting of one sample, and short sentences, a brute force method is sufficient to reconstruct the original sentence.

Given that we have extracted n tokens, we rank all $n!$ permutations of these tokens based off of their *gradient loss*. This is defined as the L2 norm between the original FL gradient, and the gradient generated when training the model with the sentence represented by the current permutation. This loss function is used in [3] to guide the gradient descent optimization as part of the Deep Leakage from Gradients algorithm.

```
Oh k...i'm watching here:)

<S> oh k im watching here
```

Fig. 3. Here we give a sample reconstruction. The first line gives the original sentence, and the second line gives the attempted reconstruction, after extracting the tokens from the gradient information and finding the best permutation of these tokens via brute force. Note the start of sentence token <S> present in the reconstruction.

Figure 3 gives an example of the kind of reconstruction that is possible with this attack. With a mini-batch size of one, and short sentences of words contained in the model's vocabulary, reconstruction is almost perfect, albeit missing punctuation. This approach does not scale well for larger mini-batch sizes or longer sentences as the number of possible number of permutations increases. Further research is needed to look into more efficient sentence reconstruction.

VI. Performance Evaluation

A. Datasets Used

To evaluate the effectiveness of our attacks we use two datasets: (i) the UMass Global English on Twitter Dataset which contains 10,502 tweets, randomly sampled from publicly available geotagged Twitter messages [20] and (ii) a corpus of 63,632 non-Spam SMS messages [21].

B. Performance Metrics

To evaluate performance we use two metrics. Firstly, the proportion of words from the original text that the attack described in Section V-A manages to correctly reconstruct. Secondly, a modified version of the Levenshtein ratio i.e. the normalised Levenshtein distance [22] (the minimum number of word level edits needed to make one string match another) between the original text and the sentences reconstructed using the attack in Section V-B. A Levenshtein ratio closer to 100 indicates a greater match between the original and reconstructed sentences. For example, given an original sentence of "hello how are you", the reconstructions "how hello are you" and "hello how you are" both have a Levenshtein ratio of 76, as they are off by one word.

C. Mini-Batches

We evaluate performance for a range of mini-batch sizes from 1 up to 48. A mini-batch of size n consists of n separate messages from the selected dataset. At the start of each separate message the LSTM is initialised, the words for the message are input and the next word predictions noted. The sum-gradient over the n messages in a mini-batch is then used for our reconstruction attack. We consider both situations where (i) all of the messages in a mini-batch have the same number of words and (ii) where the messages may have different numbers of words in which case shorter messages are padded with the <UNK> token to match the length of the longest message in the batch (this is necessary to ensure that the gradient vectors are the same size and so can be summed).

D. Measurements

Table I shows the measured accuracy at reconstructing the words contained in a mini-batch of messages vs the number of messages in the mini-batch i.e. the mini-batch size. Note that since the input is echoed by the model's output, but shifted one word ahead, the first word is not recoverable via these means. However, since the first word is always the start of sentence

TABLE I
PROPORTION OF WORDS CORRECTLY RECONSTRUCTED.

Twitter		SMS	
Messages with 4 words			
Mini-Batch Size (#batches)	Accuracy	Mini-Batch Size (#batches)	Accuracy
1 (249 batches)	0.947	1 (155 batches)	0.985
4 (62 batches)	0.975	4 (38 batches)	0.976
8 (25 batches)	0.977	8 (17 batches)	0.983
16 (15 batches)	0.965	16 (9 batches)	0.957
32 (7 batches)	0.936	32 (4 batches)	0.933
48 (5 batches)	0.907	48 (3 batches)	0.918
Messages with 8 words			
1 (405 batches)	0.913	1 (335 batches)	0.977
4 (101 batches)	0.966	4 (83 batches)	0.965
8 (50 batches)	0.961	8 (41 batches)	0.948
16 (24 batches)	0.933	16 (19 batches)	0.926
32 (12 batches)	0.908	32 (10 batches)	0.875
48 (8 batches)	0.893	48 (6 batches)	0.858
Twitter Messages with 10 or more words			
1 (2724 batches)		0.935	
4 (681 batches)		0.961	
8 (340 batches)		0.938	
16 (170 batches)		0.917	
32 (85 batches)		0.893	
48 (56 batches)		0.885	

TABLE II
4 WORD SMS AND TWITTER SENTENCE ORDERING RESULTS WITH BATCH SIZE 1.

Dataset (batch size, #batches)	Levenshtein ratio
SMS (1, #155)	97.161 (78.7% perfect)
Twitter (1, #249)	90.173 (54.2% perfect)

token <S>, this fact is inconsequential, as we are only unable to retrieve <S>, which we assume to be included as part of a training sentence. Observe also that the accuracy remains the same as the mini-batch size and messages length increase, highlighting that use of mini-batches is an ineffective defence against this attack.

To boost performance, sentences are passed through a spell checker to find mispelled words that if spelled correctly, would represent a word that is part of the model's vocabulary. This allows the attack to reconstruct the word that was typed (albeit incorrectly), instead of just the unknown token <UNK>.

Table II reports the proportion of sentences reconstructed perfectly[4] by the brute force attack described in Section V-B. We provide results for a mini-batch size of 1 and for 4 word messages from both datasets. The Levenshtein ration measures message similarity (with 100 being considered a perfect match). SMS message reconstruction reports an average Levenshtein ration of 97.161, over 155 different messages, with 78.7% of them being reconstructed perfectly, while most other being off by at most 1 word placement. Twitter message reconstruction is slightly lower, however this is due to the high number of repetition of the unknown character, <UNK>. This corresponds to the fact that tweets often consist of unique usernames, links, hashtags, emojis, etc.

VII. SUMMARY AND CONCLUSIONS

We show that when used for GBoard next word prediction Federated Learning provides little privacy and that the words typed by a user can be quickly and accurately reconstructed. The attack itself appears to be difficult to defend against. Use of mini-batches (i.e. combining multiple messages) is demonstrated to be ineffective. Sampling the gradient vector and sending only a subset of elements is unlikely to be effective due to the large size of the gradient vector relative to the average message size i.e. to provide a reasonable defence the sampling fraction would have to be so low as to disrupt neural network training. Adding noise to the gradients is also likely to be problematic since our attack just relies on sign information and noise that disrupts the attack is also likely to disrupt neural network training. Combining gradients from multiple handsets has been proposed to improve privacy but requires co-ordination between handsets which can be difficult to achieve in practice.

[4]Excluding the first word, the start of sentence token <S>.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc AISTATS 2017*, vol. 54. PMLR, 2017, pp. 1273–1282. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[2] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *CoRR*, vol. abs/1811.03604, 2018. [Online]. Available: http://arxiv.org/abs/1811.03604

[3] L. Zhu, Z. Liu, , and S. Han, "Deep leakage from gradients," in *Proc NeurIPS*, 2019.

[4] H. Yin, A. Mallya, A. Vahdat, J. M. Álvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16 332–16 341, 2021.

[5] B. Zhao, K. R. Mopuri, and H. Bilen, "idlg: Improved deep leakage from gradients," *CoRR*, vol. abs/2001.02610, 2020. [Online]. Available: http://arxiv.org/abs/2001.02610

[6] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[7] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 3–es, 2007.

[8] J. Ball, "NSA collects millions of text messages daily in 'untargeted' global sweep," 2014. [Online]. Available: https://www.theguardian.com/world/2014/jan/16/nsa-collects-millions-text-messages-daily-untargeted-global-sweep

[9] D. J. Leith and S. Farrell, "Contact Tracing App Privacy: What Data Is Shared By Europe's GAEN Contact Tracing Apps," in *Proc IEEE INFOCOM*, 2021.

[10] D. J. Leith, "Mobile Handset Privacy: Measuring The Data iOS and Android Send to Apple And Google," in *Proc Securecomm*, 2021.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[12] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, p. 2222–2232, Oct 2017. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2016.2582924

[13] K. Zhang, X. Song, C. Zhang, and S. Yu, "Challenges and future directions of secure federated learning: a survey," *Frontiers of Computer Science*, vol. 16, no. 5, p. 165817, Oct. 2022. [Online]. Available: https://link.springer.com/10.1007/s11704-021-0598-z

[14] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *CoRR*, vol. abs/2003.02133, 2020. [Online]. Available: https://arxiv.org/abs/2003.02133

[15] P. Kairouz, H. B. McMahan, and . Brendan Avent, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, pp. 1–210, 2021.

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

[17] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 691–706, 2019.

[18] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019. [Online]. Available: http://dx.doi.org/10.1109/SP.2019.00065

[19] O. Press and L. Wolf, "Using the output embedding to improve language models," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 157–163. [Online]. Available: https://aclanthology.org/E17-2025

[20] S. L. Blodgett, J. T.-Z. Wei, and B. O'Connor, "Recognizing global social media english with u.s. demographic modeling," in *Proc Workshop on Noisy User-Generated Text*. Association for Computational Linguistics, 2017.

[21] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, "Contributions to the study of sms spam filtering: new collection and results," in *DocEng '11*, 2011.

[22] A. Marzal and E. Vidal, "Computation of normalized edit distance and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 926–932, 1993.