Evaluating and Explaining Natural Language Generation with GenX

Kayla Duskin
Data Science and Analytics Group
Pacific Northwest National Laboratory
kayla.duskin@pnnl.gov

Shivam Sharma
Department of Computer Science
New Jersey Institute of Technology
ss4354@njit.edu

Ji Young Yun
Visual Analytics Group
Pacific Northwest National Laboratory
jiyoung.yun@pnnl.gov

Emily Saldanha
Data Science and Analytics Group
Pacific Northwest National Laboratory
emily.saldanha@pnnl.gov

Dustin Arendt
Visual Analytics Group
Pacific Northwest National Laboratory
dustin.arendt@pnnl.gov

Abstract

Current methods for evaluation of natural language generation models focus on measuring text quality but fail to probe the model creativity, i.e., its ability to generate novel but coherent text sequences not seen in the training corpus. We present the GenX tool which is designed to enable interactive exploration and explanation of natural language generation outputs with a focus on the detection of memorization. We demonstrate the tool on two domain-conditioned generation use cases — phishing emails and ACL abstracts.

1 Introduction

The capabilities of natural language generation (NLG) models have grown rapidly in recent years, with state-of-the-art models such as GPT-3 (Brown et al., 2020) able to produce text that is often indistinguishable from human-written text. Despite this progress, there are many remaining challenges in effectively evaluating the quality of machine text generations. Most existing evaluation approaches rely on human evaluation of the quality, fluency, and realism of a sample of generated outputs in combination with automated metrics that attempt to replicate these human judgements. However, this focus on text *quality* disregards several other key evaluation dimensions such as the *creativity* of the model and the degree of training set *memorization*.

An NLG model that simply reproduces long text snippets from the training data is likely to achieve high quality, but does not represent the ability of the model to creatively generate novel text sequences. This can contribute to an inappropriate belief in the model's sophistication if users are not aware the generated text is copied wholesale from the training data. Data scientists developing NLG models are not likely to be familiar enough with a given training corpus to detect this problem from the NLG model output without additional tool support.

A second related issue arises more generally when text datasets collected from multiple sources are used to train machine learning models. In this case, identical text substrings can inadvertently end up on both sides of a train-test split. This can lead to artificially inflated model performance metrics, especially in deep learning models, having sufficient parameters to enable input memorization and shortcut generalization. While detection of exact duplicates is straightforward, detection of partial, sub-document duplication is more challenging.

To address these issues, we present the GenX [1] tool which is designed to enable data scientists to understand the provenance of the output of a text generation model. Specifically, GenX lets users understand which sentences or passages from a generated text output are very similar to sentences in the model's training input. It compares sentences in the output text to text that the model was trained on and renders a marked up version of the text to indicate what parts of the text may have been memorized from the training data. The tool also lets the user find interesting text based on pre-computed statistics related to this potential memorization.

[1] Source: https://github.com/pnnl/genx

2 Related Work

Language generation metrics Many NLG tasks are framed as supervised sequence-to-sequence problems, such as in the case of machine translation. Metrics for such tasks evaluate the similarity between a candidate sentence and a set of reference sentences. There are wide range of automated metrics including BLEU (Papineni et al., 2002), SARI (Xu et al., 2016), BLUERT (Sellam et al., 2020), and GLEU (Wu et al., 2016). These metrics have been shown to have mixed success in terms of replicating the intuition of humans regarding text quality (Novikova et al., 2017).

For open domain NLG models, datasets such as Penn Tree Bank (Marcus et al., 1994) or LAMBADA (Paperno et al., 2016) are commonly used for evaluation. However, these datasets cannot help when models are meant to be constrained to a certain domain, and they do not consider long-form text generation, only text completion tasks. Another common method is to leverage the trained model for downstream tasks to assess the quality of the language model (Radford et al., 2019). Work by (Hashimoto et al., 2019) has proposed combining human and statistical evaluation to measure the quality and diversity of generated text.

Domain-conditioned text generation NLG models can be evaluated by their perplexity calculated on a held out data set. While perplexity is useful for measuring model performance, it has limitations in measuring quality (Theis et al., 2015) and is typically is calculated at the model level, without taking in to consideration differences in generated text that result from different decoding strategies that affect the quality of the output.

Evaluating memorization in language generation In comparison to work related to text quality measures, less work has been dedicated to the evaluation of memorization in NLG models. In addition to its direct bearing on model creativity, memorization of training data in generation models has significant privacy implications, especially in domains that include sensitive information such as social media data or clinical notes. Previous efforts to evaluate memorization have focused on the leakage of sensitive information by adding "secret" information to the training data and evaluating the perplexity of the inserted secret during generation (Carlini et al., 2019). There as been little previous work on looking for memorization more generally in order to evaluate model creativity.

Evaluating test set contamination A number of recent works have identified issues in natural language processing datasets with text overlap and near-duplication in training and testing sets leading to artificially inflated performance metrics. Such issues have been identified in question answering datasets (Lewis et al., 2020) and large software and code corpora (Allamanis, 2019). Language modeling benchmarks have also been shown to exhibit this issue. For instance, the Billion Word Benchmark has a 13% overlap between train and test 8 grams (Radford et al., 2019). Language models trained on large datasets scraped from the web also pose a risk for test set contamination. Brown et al. (2020) evaluate the impact of test example presence in the pre-training set on GPT-3 for some of their benchmark test sets using 13-gram overlap. They find a substantial amount of overlap between their pretraining data and test data (>50% for a quarter of the benchmarks), but noted that manual inspection of the overlapping examples showed a significant number of false positives.

Interactive/explanation tools Previous work has largely focused on developing methods for automated quantitative evaluation of generation quality, but fewer efforts have been applied to the development of interactive tools to explain and understand the generation of individual examples. The *compare-mt* tool automates the comparison of multiple NLG models according to traditional BLEU-type metrics as well as providing more detailed breakdowns of accuracies by word or sentence type (Neubig et al., 2019). The *VizSeq* tool provides an interactive interface to explore metric performance on the full corpus, groups of instances, and individual examples (Wang et al., 2019). The existing tools are largely focused on text quality evaluation rather than memorization evaluation and are designed specifically for supervised generation tasks such as translation rather than open-domain or domain-conditioned generation tasks.

Anti-plagiarism software Anti-plagiarism tools also aim at quantifying similarity between texts. Many such tools are proprietary, reference against an existing database of published work, and consider each document on an individual basis. In contrast, GenX allows for referencing against specific training text and is meant to assess a collection of generated documents as a whole. Additionally, in NLG not all “copying” is bad, and GenX characterizes any matching text segments through metrics that go beyond a binary classification.

3 GenX Tool & Implementation

GenX is implemented as a Jupyter Notebook[2] widget[3], which allows for an interactive user experience that is tightly integrated with a popular computational environment for data science. The widget is implemented in two parts: a Python side, which performs preprocessing and integration with the Jupyter environment, and a JavaScript side, which handles rendering user interaction. The inputs to GenX are Pandas[4] DataFrames for the raw text (each row in the data frame corresponds a sentence), the corresponding sentence-level embedding representation of that text, and an identifier for which document the sentence belongs to. GenX requires that the raw text and embeddings are also split into train and test sets. The test set may either be text generated from an NLG model or the test split of the real data. Thus GenX inputs are *train text*, *train embedding*, *test text*, and *test embedding* DataFrames. By design, GenX does not assume a particular embedding technique and requires the user to compute the embeddings. This allows the user to employ whatever method is appropriate for their use-case (e.g. TF-IDF, neural network) and does not preclude the adoption of new state-of-the-art embeddings methods in the future. For our demonstrations, we use Sentence BERT (Reimers and Gurevych, 2019) to create the embeddings used in the tool.

During preprocessing, i.e., after input but before rendering, the Python half of GenX computes the cosine distance between each sentence in the train and test embeddings. The 10 nearest neighbors of each sentence in the test split and their respective distances are passed to the JavaScript half of the widget along with the test sentence DataFrame. The tool passes the rows of the train text DataFrame that were among the neighbors of any sentence in the test set. When a test sentence is rendered, its nearest neighbor distances are visualized in a bar graph following that sentence. The bars are sorted by distance, with the first nearest neighbor placed on the left, and the last nearest neighbor on the right, so the bars always increase monotonically. They allow the user to get a better understanding of the nearest neighbor distribution, e.g., whether the first nearest neighbor is unique or there are other semantically similar sentences in the train set.

Furthermore, the tool indicates which parts of the sentences are copied verbatim, or nearly so from the training data. To do so, we align each test sentence against its nearest neighbor in the train set using dynamic time warping (Sakoe and Chiba, 1978) at the token level. We use Levenshtein distance (Levenshtein, 1966) as the token-token distance function[5]. We highlight the tokens in the test sentence that are exactly matched to tokens in their nearest neighbor sentence with a strong underline. Tokens that are partially matched, i.e. with a Levenshtein distance less than 5, have a weaker underline. The remaining tokens are not underlined. The user can mouse over a bar to compare the text of each nearest neighbor against the test sentence.

When reading a document, the user may want to get a sense of what documents the nearest neighbor sentences are sourced from. For example, when repeats occur, do they occur together in the same source document? We include a step line chart visualization above the text to illustrate this. The x-axis is the sentence number of the generated sentence, and the y-axis is the source document identifier of the nearest neighbor of that sentence. The y-axis is sorted by first occurrence, so that the chart will increase monotonically unless a source document is revisited, which is clearly visible as dip in the chart. The line chart is also brushable allowing the user find corresponding sentences in the text below.

The tool also contains an interactive scatter plot to help the user focus on interesting or problematic examples of generated text and avoid having to page through every document. The axes of the scatter plots are two novel document-level metrics which we refer to as *distinctiveness* and *diversity*.

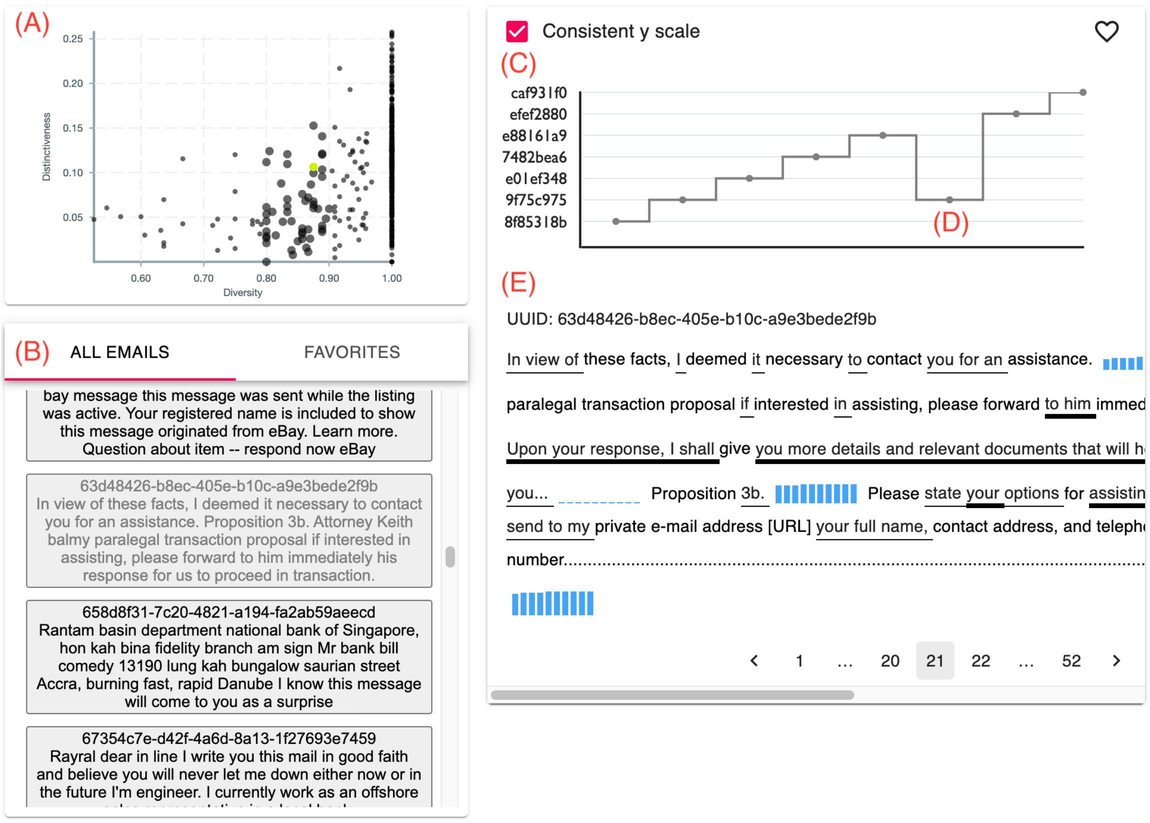
For a given document in the test data, Distinctiveness is the distance of the first nearest neighbor to each test sentence in a document, averaged across the generated sentences in the document. Low distinctiveness means that many sentences in that document were semantically similar to sentences in the training set, and indicate copying for specific phrases and may be symptomatic of memorizing repeated phrases. Low distinctiveness for the training set overall may be indicative of broader model

[2]https://jupyter.org

[3]https://github.com/jupyter-widgets/widget-cookiecutter

[4]https://pandas.pydata.org

[5]We tried a simpler approach of using Dynamic Time Warping at the character level, but this produced difficult to interpret highlighting for sentences with low alignment.



bay message this message was sent while the listing was active. Your registered name is included to show this message originated from eBay. Learn more. Question about item -- respond now eBay

63d48426-b8ec-405e-b10c-a9e3bede2f9b
In view of these facts, I deemed it necessary to contact you for an assistance. Proposition 3b. Attorney Keith balmy paralegal transaction proposal if interested in assisting, please forward to him immediately his response for us to proceed in transaction.

658d8f31-7c20-4821-a194-fa2ab59aeecd
Rantam basin department national bank of Singapore, hon kah bina fidelity branch am sign Mr bank bill comedy 13190 lung kah bungalow saurian street Accra, burning fast, rapid Danube I know this message will come to you as a surprise

67354c7e-d42f-4a6d-8a13-1f27693e7459
Rayral dear in line I write you this mail in good faith and believe you will never let me down either now or in the future I'm engineer. I currently work as an offshore

UUID: 63d48426-b8ec-405e-b10c-a9e3bede2f9b

In view of these facts, I deemed it necessary to contact you for an assistance.

paralegal transaction proposal if interested in assisting, please forward to him immed

Upon your response, I shall give you more details and relevant documents that will h

you... Proposition 3b. Please state your options for assistin

send to my private e-mail address [URL] your full name, contact address, and teleph

number...

Figure 1: GenX interface with the distinctiveness/diversity overview of the corpus (A), the document navigation tool (B), and the individual document view containing the step line chart for diversity visualization (C) with an interesting dip indicating revisiting of a training document (D) and the document text with train data overlap and similarity annotations (E).

issues impacting creativity, potentially caused by sub-optimal parameter settings.

Each sentence in the training data is found within a particular source document. For a given document in the test data, Diversity is the number of unique corresponding source documents for the nearest neighbor of each test sentence in the generated document divided by the number of test sentences. Low diversity means the test document has similarity to a single source document, and is indicative of longer length copying from the training set, while a maximum diversity value of 1.0 indicates that the nearest neighbor of each generated sentence is from a different document in the training set. Because of the limited prior work on model memorization and lack of existing metrics, we introduce these two new metrics to quantitatively capture the patterns of sentence-level (distinctiveness) and document-level (diversity) memorization by the models. We also average these metrics across documents in the test corpus for corpus-level analysis (see Table 1).

4 Use Case: Phishing Email Generation

Phishing Emails We initially developed the GenX tool when working with a composite dataset of publicly available phishing datasets that contained many duplicates and near-duplicates. This dataset was initially comprised of the aggregation of data made available by (Azunre, 2019), and (Nazario, 2011) as well as phishing emails provided by industry partners. The initial dataset contained a total of 60,705 emails, however after initial de-duplication efforts using exact string matching only 9,234 emails remained which was split into a train set of 8,311 and a test set of 923. After further de-duplication efforts, aided by GenX, the final dataset consists of 5,634 emails in the training set and held out validation and test sets of size 500 each. While we have been rigorous in our efforts to remove emails that are duplicated, the formulaic nature of phishing emails leads to many commonly repeated phrases, sentences, or paragraphs.

Phishing Test/Train split While the GenX tool was originally designed for the evaluation of mem-

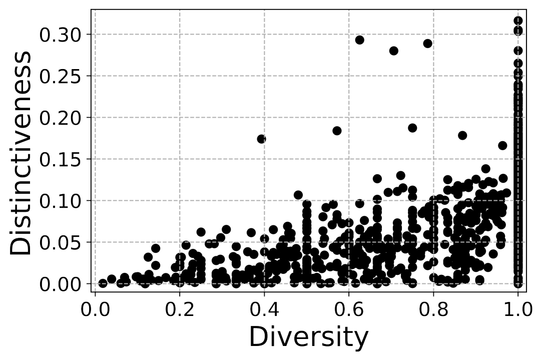
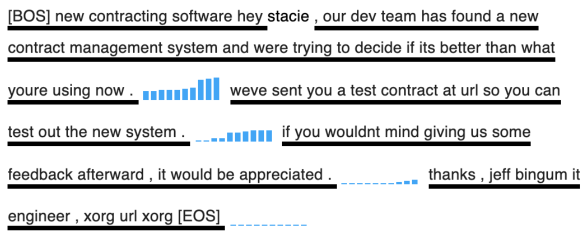


Figure 2: Distinctiveness and diversity scores for emails in the train and test sets of the phishing dataset reveal that there is significant test set contamination

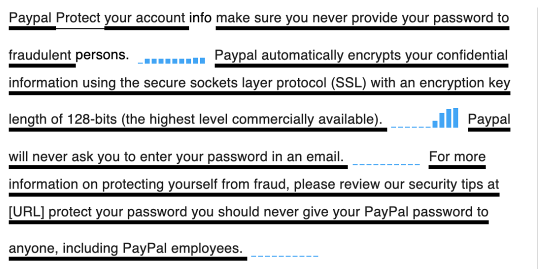


[BOS] new contracting software hey stacie , our dev team has found a new contract management system and were trying to decide if its better than what youre using now . weve sent you a test contract at url so you can test out the new system . if you wouldnt mind giving us some feedback afterward , it would be appreciated . thanks , jeff bingum it engineer , xorg url xorg [EOS]

Figure 3: Example email from the phishing test set with high overlap with an email from the training set, differing only in the name of the recipient.

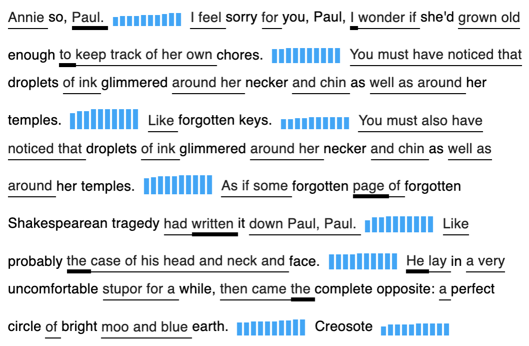
orization in NLG models, its ability to explore text overlap makes it well suited to the task of looking for test set contamination. To test this use case, we use GenX to look for text duplication across the train/test split of our 9k email phishing data set. Figure 2 shows the distinctiveness/diversity scatter plot which reveals a large number of of low-distinctiveness, low-diversity pairs between the training and test set which is indicative of significant levels of text duplication. Additionally, the example shown in Figure 3 demonstrates how GenX allows for qualitative analysis of the text in question. We can see from the underlining that almost all of the text in the example test set email appeared verbatim in a training set email, with the only difference being the name of the recipient. We identified this as a common pattern within the dataset because attackers duplicate popular phishing emails, making minor edits for personalization.

Phishing generation To build a phishing domain-conditioned generation model, we fine-tuned a GPT-2 small model (Radford et al., 2019) on the phishing training set using a learning rate of $5 * 10^{-5}$ and a batch size of 8. We chose two models to illustrate the use of GenX for qualitative



Paypal Protect your account info make sure you never provide your password to fraudulent persons. Paypal automatically encrypts your confidential information using the secure sockets layer protocol (SSL) with an encryption key length of 128-bits (the highest level commercially available). Paypal will never ask you to enter your password in an email. For more information on protecting yourself from fraud, please review our security tips at [URL] protect your password you should never give your PayPal password to anyone, including PayPal employees.

(a) High memorization, highly coherent example



Annie so, Paul. I feel sorry for you, Paul, I wonder if she'd grown old enough to keep track of her own chores. You must have noticed that droplets of ink glimmered around her necker and chin as well as around her temples. Like forgotten keys. You must also have noticed that droplets of ink glimmered around her necker and chin as well as around her temples. As if some forgotten page of forgotten Shakespearean tragedy had written it down Paul, Paul. Like probably the case of his head and neck and face. He lay in a very uncomfortable stupor for a while, then came the complete opposite: a perfect circle of bright moo and blue earth. Creosote

(b) Low memorization, incoherent example

Figure 4: Example generations from the phishing models showing the memorization/coherence trade-off

analysis of different models. Model 1 was trained for 10 epochs, while Model 2 was trained for 20 epochs. For each model we generate 500 unique emails, using a decoding temperature of 1.1.

We leverage GenX to perform qualitative evaluation of the levels of memorization in generations by these models. We find an overall high level of training email memorization, with the generation models producing emails that are nearly word-for-word replications of emails from the training set. In the distinctiveness-diversity plots (Figure 5), we observe that while there are many generated emails with high diversity, there still a significant population of emails with low distinctiveness and diversity scores. Using these plots to identify emails with lower and higher levels of memorization and then observing the corresponding email text in the individual document view, we are able to discover the interesting pattern that lower levels of memorization seem to be correlated with lower levels of coherence as determined by the human. In other words, the models are unable to creatively produce novel phishing emails and must rely on rote copying from the training data for reasonable human-evaluated performance. We can also use the tool to perform relative comparisons between memorization across different modeling choices. In this case, we find that increasing the number of train-

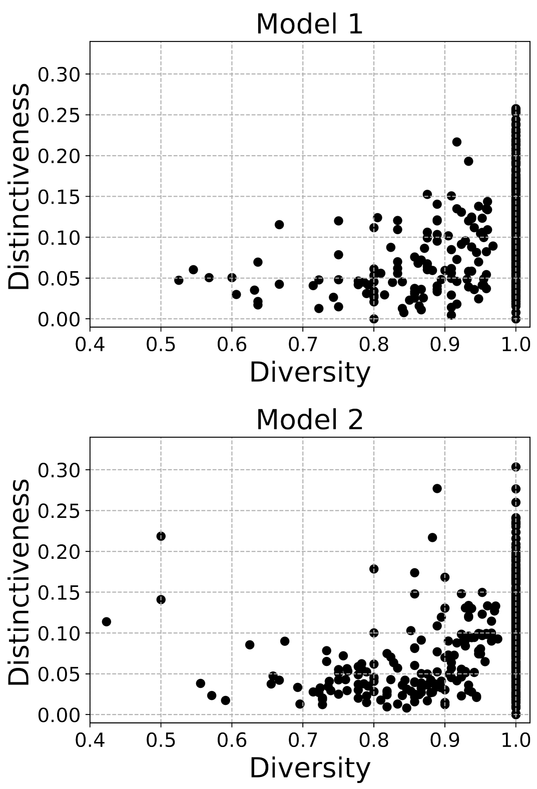
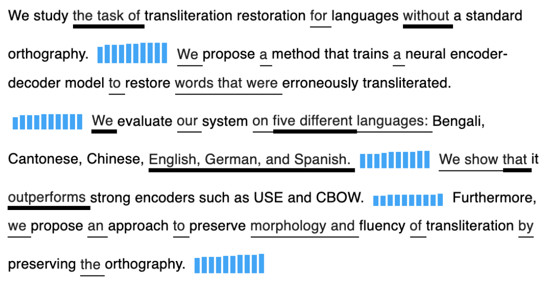
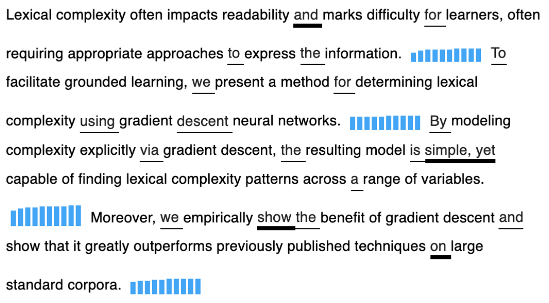


Figure 5: Diversity vs. distinctiveness for the two phishing NLG models. Both show training data memorization, but Model 2 contains more low-distinctiveness, lower-diversity examples).



We study the task of transliteration restoration for languages without a standard orthography. We propose a method that trains a neural encoder-decoder model to restore words that were erroneously transliterated. We evaluate our system on five different languages: Bengali, Cantonese, Chinese, English, German, and Spanish. We show that it outperforms strong encoders such as USE and CBOW. Furthermore, we propose an approach to preserve morphology and fluency of transliteration by preserving the orthography.

(a) ACL Model 1: Low memorization, highly coherent example



Lexical complexity often impacts readability and marks difficulty for learners, often requiring appropriate approaches to express the information. To facilitate grounded learning, we present a method for determining lexical complexity using gradient descent neural networks. By modeling complexity explicitly via gradient descent, the resulting model is simple, yet capable of finding lexical complexity patterns across a range of variables. Moreover, we empirically show the benefit of gradient descent and show that it greatly outperforms previously published techniques on large standard corpora.

(b) ACL Model 2: Low memorization, highly coherent example

Figure 6: Generated abstracts with low memorization and high creativity of the generative models.

ing epochs increases the level of memorization. We show several examples of generated emails from these models in Figure 4, which highlight the memorization-coherence trade off.

5 Use Case: ACL Abstract Generation

ACL Abstract Data The second dataset is the set of abstracts available from the ACL anthology [6], chosen simply because we considered it a relevant corpus for demonstration. We employed 17,903 abstracts as the training data for our generative model and withheld 2,000 abstracts for validation and 2,000 abstracts as the test set, whose NLG model perplexity is reported in in Table 1.

ACL generation We fine-tuned a GPT-2 small model (Radford et al., 2019) on the ACL training set with a learning rate of $5 * 10^{-5}$ and a batch size of 8. For this comparison we used a model that had been trained for 25 epochs, but created two sets of generated examples using different decoding temperatures, 0.8 for Set 1 and 1.1 for Set 2. Each set contains 600 generated abstracts.

In contrast with the phishing data set, the exploration of model outputs for the ACL data with the GenX tool reveal that these models achieve low levels of memorization and high levels of coherence overall. We show several example abstracts generated from the ACL models in Figure 6. We can observe the low level of memorization in these abstracts because the only words underlined in the generated text are common words like "the" or "an" indicating that the nearest neighbor sentences in the training data only overlap in an insignificant way with the generated text. Additionally, we observe the uniformly high distinctiveness values of the nearest neighbor distance bar charts. We use the distinctiveness/diversity scatter plots (Figure 7) to explore generations with higher and lower metric values. In contrast with the phishing models, our qualitative examination reveals that even generated abstracts with no indications of memorization have high coherence, indicating that the models can generate creative and novel outputs without resorting to copying from the training data. Comparison of the two different decoding temperatures, reveals that lower temperature does not result in increased memorization but it does have slightly improved coherence than the higher temperature model.

[6]https://www.aclweb.org/anthology/

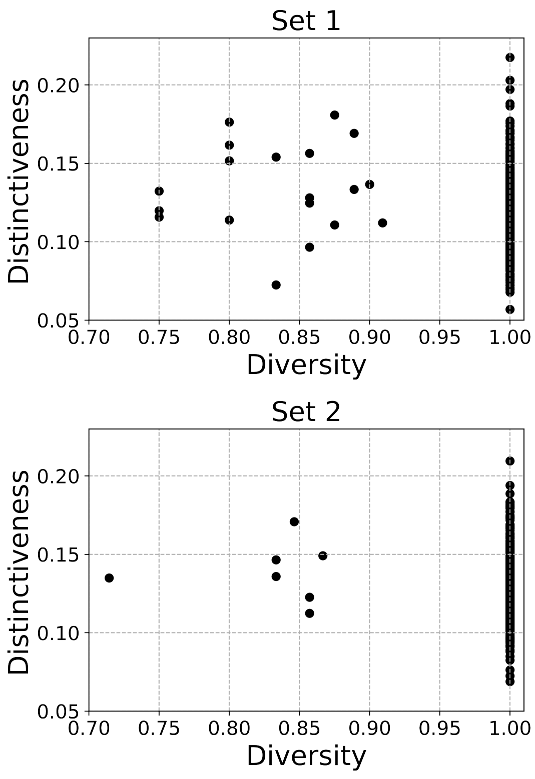


Figure 7: Diversity vs. distinctiveness for ACL NLG models. Both sets show high levels of high-diversity examples, with almost no low-diversity, low-distinctiveness generations

Task	Model	Perplexity	Average Distinctiveness	Average Diversity
Phishing Train Vs Test	-	-	0.08	0.80
Phishing Generations	Model 1	8.08	0.10	0.96
	Model 2	8.01	0.10	0.95
ACL Generations	Set 1	21.11	0.12	0.99
	Set 2	21.11	0.13	1.0

Table 1: The perplexity, average distinctiveness score, and average diversity score for each set of texts. We report the average scores here but in practice find that the average values are not as useful for diagnosing memorization and recommend using the interactive tool or scatter plots to identify specific low-quality examples.

6 Discussion & Limitations

GenX is a tool for incorporating human judgement with the analysis of generated text, without placing the full burden on human annotators to locate instances where the model is copying from the training data. The distinctiveness and diversity scores provide quantitative context to the qualitative interpretation of the highlighted text. When used on a model with high levels of duplication in the training set, the tool helped us establish a threshold of acceptable memorization. When used on a model with low levels of duplication the tool helped us identify compelling examples of creatively generated text not copied from the training data.

A key contribution of this work is going beyond the typical evaluation metrics such as perplexity. Table 1 shows that perplexity alone does not reveal the nuanced behavior of generative models. Looking at perplexity scores on the held out data alone, the ACL NLG model seems to perform worse than the phishing NLG model when in reality the ACL NLG model produces coherent examples with low levels of memorization. Additionally, GenX highlights differences between models where the perplexity is similar but there are qualitative differences in the generation, especially when the interactive components of the tool are used to understand the distribution of the memorization levels of individual emails and identify specific patterns and examples of memorization.

GenX has challenges scaling to large datasets. While we demonstrated utility on datasets with tens of thousands of text examples, the nearest neighbor approach used would become intractable on massive text corpora such as CommonCrawl [7]. This limits GenX to scenarios where training data is a manageable size, but does not yet help address the issue of test set contamination from large-scale web scrapes. As future work, we plan to incorporate approximate neighbor techniques (Dong et al., 2011) to mitigate this issue.

In the phishing use case, identical sentences were found across many training documents, making diversity measurements artificially high, due to the arbitrary choice of nearest neighbors. We plan to use a minimum set cover algorithm to improve diversity score accuracy to break ties by selecting the smallest number of training documents that cover the nearest neighbors in the test document.

7 Conclusion

GenX provides a unique capability for interactive evaluation and explanation of NLG model output. The tool goes beyond typical aggregate performance metrics and provides new insight into domain-conditioned NLG model creativity and memorization. Across two use cases, we showed this helped distinguish models in situations where aggregate evaluation metrics did not.

[7]https://commoncrawl.org/the-data/

Broader Impact and Ethical Statement

Natural language generation (NLG) models have received much attention beyond their research community. However such attention can be harmful when it inappropriately attributes human-level intelligence and creativity to clever statistical processes. Increased transparency and explainability of NLG models can help to prevent societal harm that arises from over-estimating model ability. Furthermore, the applicability of GenX to ensure more distinct train/test splits also helps to create more robust language models (by decreasing overestimated F-scores) that have similar performance "in the wild" and in the laboratory.

There are ethical considerations around conditioning NLG models on phishing emails. These emails are malicious by nature, and our models could provide bad actors a means to cause greater harm. However, the research described in this paper is part of a broader effort to generate realistic phishing emails for educational purposes to mitigate users susceptibility to phishing. Our work can reduce the burden on analysts who currently painstakingly craft these training emails by hand. We are also encouraged by the positive results in fake-news detection (Zellers et al., 2019) and believe that the insights from phishing generators can inform more robust phishing detection models. We do not plan to publicly release the phishing domain conditioned models or source code used in this specific use case.

References

Miltiadis Allamanis. 2019. The adverse effects of code duplication in machine learning models of code. In *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward! 2019, page 143–153, New York, NY, USA. Association for Computing Machinery.

Paul Azunre. 2019. Fraudulent email bodies. `https://www.kaggle.com/azunre/fraudulent-email-bodies`.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284.

Wei Dong, Charikar Moses, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, pages 577–586.

Tatsunori B Hashimoto, Hugh Zhang, and Percy Liang. 2019. Unifying human and statistical evaluation for natural language generation. *arXiv preprint arXiv:1904.02792*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2020. Question and answer test-train overlap in open-domain question answering datasets. *arXiv preprint arXiv:2008.02637*.

Mitch Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Jose Nazario. 2011. PhishingCorpus. `https://monkey.org/~jose/phishing/`.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. 2019. compare-mt: A tool for holistic comparison of language generation systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 35–41, Minneapolis, Minnesota. Association for Computational Linguistics.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. 2015. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*.

Changhan Wang, Anirudh Jain, Danlu Chen, and Jiatao Gu. 2019. Vizseq: A visual analysis toolkit for text generation tasks. In *In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in neural information processing systems*, pages 9054–9065.