Finding Memo: Extractive Memorization in Constrained Sequence Generation Tasks

Vikas Raunak **Arul Menezes**
Microsoft Azure AI
Redmond, Washington
{viraunak,arulm}@microsoft.com

Abstract

Memorization presents a challenge for several constrained Natural Language Generation (NLG) tasks such as Neural Machine Translation (NMT), wherein the proclivity of neural models to memorize noisy and atypical samples reacts adversely with the noisy (web crawled) datasets. However, previous studies of memorization in constrained NLG tasks have only focused on counterfactual memorization, linking it to the problem of hallucinations. In this work, we propose a new, inexpensive algorithm for extractive memorization (exact training data generation under insufficient context) in constrained sequence generation tasks and use it to study extractive memorization and its effects in NMT. We demonstrate that extractive memorization poses a serious threat to NMT reliability by qualitatively and quantitatively characterizing the memorized samples as well as the model behavior in their vicinity. Based on empirical observations, we develop a simple algorithm which elicits non-memorized translations of memorized samples from the same model, for a large fraction of such samples. Finally, we show that the proposed algorithm could also be leveraged to mitigate memorization in the model through finetuning. We have released the code to reproduce our results at https://github.com/vyraun/Finding-Memo.

1 Introduction

Previous studies (Arpit et al., 2017; Feldman, 2020; Zhang et al., 2021a) have shown that neural networks capture regular patterns in the training data (generalization) while simultaneously fitting noisy and atypical samples using brute-force (memorization). For constrained Natural Language Generation tasks such as Neural Machine Translation (NMT), which rely heavily on noisy (web crawled) data for training high-capacity neural networks, this creates an inherent reliability problem. For example, memorizations could manifest themselves in the form of catastrophic translation errors on specific samples despite high average model performance (Raunak et al., 2021). It is also *likely* that the memorization of a specific sample could corrupt the translations of samples in its vicinity. Therefore, exploring, quantifying and alleviating the impact of memorization is of critical importance for improving the reliability of such systems.

Yet, most of the work on memorization in natural language processing (NLP) has focused either on classification (Zheng and Jiang, 2022) or on unconstrained generation tasks, predominantly language modeling (Carlini et al., 2021; Zhang et al., 2021b; Kharitonov et al., 2021; Chowdhery et al., 2022; Tirumala et al., 2022; Tänzer et al., 2022; Haviv et al., 2022). In this work, we fill a gap in the literature by developing an analogue of extractive memorization for constrained sequence generation tasks in general and NMT in particular. Our main contributions are:

1. We propose a new, inexpensive algorithm for studying extractive memorization in constrained sequence generation tasks and use it to characterize memorization in NMT.
2. We demonstrate that extractive memorization poses a serious threat to NMT reliability by quantitatively and qualitatively analyzing the memorized samples and the neighborhood effects of such memorization. We also demonstrate that the memorized instances could be used to generate errors in *disparate* systems.
3. Based on an analysis of the neighborhood effects of memorization, we develop a simple memorization mitigation algorithm which produces non-memorized (higher quality) outputs for a large fraction of memorized samples.
4. We show that the outputs produced by the memorization mitigation algorithm could also be used to directly impart corrective behavior into the model through finetuning.

Repetitions	Total Samples	Memorized	Ratio (%)	Perturb Prefix	Perturb Suffix	Perturb Start
1	100,000	174	0.17	17.58 %	43.24 %	12.29 %
2	100,000	317	0.32	11.67 %	62.84 %	4.98 %
3	5,381	17	0.32	28.42 %	49.52 %	18.82 %
4	1,885	5	0.26	27.40 %	34.00 %	8.00 %
5	976	7	0.72	26.67 %	70.00 %	11.42 %
1-5	**208,242**	**520**	**0.25**	**16.65 %**	**51.65 %**	**8.00 %**

Table 1: **Quantifying Extractive Memorization:** Number of Memorized Samples (using Algorithm 1) and Neighborhood Effects of Memorization (using Algorithm 2) across different training data frequency buckets.

2 Related Work

Our work is concerned with the phenomenon of memorization in constrained natural language generation in general and NMT in particular. The main challenge in analyzing memorization is to determine which samples have been memorized by the model during training. There exist two key algorithms to elicit memorized samples, each yielding a distinctive operational definition of memorization:

1. **Counterfactual Memorization**: Feldman and Zhang (2020) study label memorization and propose to estimate the memorization value of a training sample by training multiple models on different random subsets of the training data and then measuring the deviation in the sample's classification accuracy under inclusion/exclusion. This definition of memorization was further extended to arbitrary performance measures by Raunak et al. (2021) to study memorization in NMT and by Zhang et al. (2021b) to study memorization in language models. However, a practical limitation of analysis based on this definition is the prohibitive computational cost (multiple model trainings) associated with computing memorization values for each training sample.

2. **Extractive Memorization**: Carlini et al. (2021) propose a data-extraction based definition of memorization to study memorization in language models. Therein, a training string s is extractable if there exists a prefix c that could exactly generate s under an appropriate sampling strategy (e.g. greedy decoding). This definition has the benefit of being computationally inexpensive, although it doesn't have any existing analogue for constrained natural language generation tasks such as NMT.

In the next section, we define extractive memorization for constrained sequence generation tasks and apply it to NMT, in section 4 we estimate the neighborhood effect of such memorizations and in section 5 we propose a simple algorithm for recovering correct translations of memorized samples.

3 Extractive Memorization

We present our definition of extractive memorization as Algorithm 1. Analogous to extractive memorization in language models (Carlini et al., 2021), this definition labels an input sentence (source) as being memorized if its transduction (translation) could be replicated exactly with a prefix considerably shorter than the length of the full input sentence (source), under greedy decoding. Operationally, we set prefix ratio threshold (p) to 0.75.

Algorithm 1: Extractive Memorization in NMT

Data: Trained NMT Model T, Training Dataset S, Prefix Ratio Threshold p
Result: Memorized Samples M, Prefix Lengths L
Greedily Translate Sources in S using T;
M_1 = Sources with translations matching References;
Greedily Translate Prefixes of Sources in M_1 using T;
M_2 = Sources with Prefixes producing References;
for M_2^i *in* M_2 **do**
  n = Length of the Source M_2^i ;
  l = Length of Smallest Prefix producing the Ref;
  if $\frac{l}{n} \leq p$ **then**
    Add M_2^i to M and Add l to L

Next, we apply this definition of memorization on a strong Transformer-Big (Vaswani et al., 2017) baseline trained on the 48.2M WMT20 En-De parallel corpus (Barrault et al., 2020). We describe the dataset, model and training details in Appendix A.

Qualitatively, we observe that the memorized samples detected by Algorithm 1 mostly consist of low-quality samples – templatized source sentences and noisy translations. To analyze the results quantitatively, similar to Carlini et al. (2022), we bucket the training data pairs in terms of their repetitions in the training data. Owing to the sparsity of data with greater than 5 repetitions we report results in

Provenance	Source	Translation
Training Data	Why study in Peru? Spanish Courses	Warum in Peru studieren?
Perturb Suffix	Why study in Peru? University Courses	Warum in Peru studieren?
Perturb Suffix	Why study in Peru? Short Courses	Warum in Peru studieren?
Perturb Suffix	Why study in Peru? Summer Courses	Warum in Peru studieren?
Perturb Prefix	You study in Peru? Spanish Courses	Sie studieren in Peru? Spanischkurse
Perturb Prefix	Advanced study in Peru? Spanish Courses	Weiterbildung in Peru? Spanischkurse

Table 2: **Memorization Example**: An example to illustrate the phenomenon of Memorization (elicited using Algorithm 1) and the ensuing Neighborhood Effect of such memorization (measured using Algorithm 2). This WMT20 English to German training sample is memorized with a prefix length ratio of 0.67 (≤ 0.75).

the range of 1-5 repetitions. Further, for repetition values 1 and 2, we select 100K random samples for analysis. We observe two key results:

Repetitions vs Memorization: Table 1 shows that the percentage of samples memorized is higher for repeated training samples, compared to samples present only once in the training data, with a Pearson's correlation coefficient of 0.778 between the number of repetitions and memorizations.

Quality of Memorized Samples: Table 3 shows that both the quality of memorized samples, measured using COMET-QE (Rei et al., 2020), a state-of-the-art Quality Estimation model for MT as well as their lexical diversity, measured using Type-to-Token Ratio (TTR) (Vanmassenhove et al., 2021) is worse when compared to the total samples.

Rep.	T-COM	M-COM	T-TTR	M-TTR
1	42.91	32.81	9.81	44.87
2	60.63	62.56	8.85	36.92
3	73.19	37.31	20.03	68.97
4	70.18	61.92	27.72	78.85
5	68.41	46.65	33.90	73.47
1-5	**54.79**	**51.57**	**7.64**	**38.72**

Table 3: Comparison of **COM**ET-QE ↑ and **TTR** ↓ for Memorized (**M**) Samples vs Total (**T**) Samples.

The above two results are similar to the results in language modeling (Carlini et al., 2022) and serve to demonstrate the utility of Algorithm 1 in analyzing extractive memorization in NMT.

4 Neighborhood Effect of Memorization

To measure the neighborhood effect of memorization, we generate new source sentences in the vicinity of the memorized samples through perturbations and test whether they generate the same output under greedy decoding. Table 2 shows a memorized training sample from Section 3, alongside translations generated from perturbations at different positions in the source. The perturbations (substitutions) were generated using BERT-Cased (Devlin et al., 2019). We define suffix positions and prefix positions based on the recorded prefix lengths (L) in Algorithm 1. Specifically, we use Algorithm 2, which generates new sources in the neighborhood of a memorized input source by perturbing its tokens at prefix (P), suffix (S) or the beginning (B) positions and then computes how many such new sources still translate to the same memorized output. The resulting effect measure N is higher if the sources still produce the same memorized output under perturbations at different positions. The intuition behind Algorithm 2 is that we want to explore meaningful source sequences around the memorized source sentence. Therefore, we make changes to only a single position at a time, to keep the generated sequence close to the original sequence and substitute that position with a word that fits well in the particular context. Our 'neighbour' definition is loosely inspired by the differential privacy literature (Dwork et al., 2014), where a neighbour is used to mean datasets differing in only one item.

Algorithm 2: Estimating Neighborhood Effect

Data: Memorized Samples M, Prefix Lengths L, Masked Language Model W, Candidates K
Result: Perturbed Sources $S^{'}$, Effect Measure N
for M_i, L_i *in* M, L **do**
  n = Length of the Source M_i ;
  P_i = [1,...,$L_i - 1$], S_i = [$L_i + 1$,..., n], B_i = [1] ;
  for j *in* P_i, S_i, B_i **do**
    Generate K substitutes at $M_i[j]$ using W ;
    Translate each new Source and Add to $S^{'}$;
N = Fraction of $S^{'}$ Generating Memorized Outputs;

Results: The last 3 columns of Table 1 present the results of applying Algorithm 2 on memorized sources (K=5). We find that considerable

percentages of perturbations still yield the same translations, and that this effect is highly position-dependent. For example, for the unique memorized samples (repetitions = 1), perturbing the suffix produces no changes in translations for 43.2% of the new sources; while for prefix perturbations the same memorized output is produced only 17.6%. Further, perturbing the first token of the source is highly successful in generating a different (non-memorized) output. The results demonstrate that a single memorization may therefore be able to corrupt the translations of multiple input sequences in its vicinity.

Table 1 and 2 show that while the translations of memorized sources frequently remain invariant under suffix perturbations, translations under prefix-perturbations do change more frequently. We hypothesize that this results from the model switching away from its memorization mode, owing to a change in the input prefix used for memorization.

5 Generating Non-Memorized Outputs

The results in section 4 (and table 2) show that the model is able to generate non-memorized translations quite frequently if memorized source's start token is perturbed. We demonstrate that this fact could be exploited to *recover* the *non-memorized* translations of a memorized sample from the same model with a surprisingly high-frequency. We call this task Memorization-Mitigation and present a simple technique to do so in Algorithm 3.

Algorithm 3: Generating Non-Memorized Outputs

Data: Trained NMT Model T, Memorized Samples M, Recovery Symbol X
Result: Memorized Samples with New Outputs F
for S_i *in* M **do**
  R_i = Translate S_i using T;
  S_i^{aug} = X + S_i ;
  R_i^{aug} = Translate S_i^{aug} with T ;
  if $R_i^{aug} \neq R_i$ *and* R_i^{aug} *starts with* X **then**
    Strip X from R_i^{aug}, Add (S_i, R_i^{aug}) to F

Source: Victor Emmanuel II of Italy
Translation (Reference): Viktor Emanuel II.
Perturbed Source: ! Victor Emmanuel II of Italy
Translation: ! Viktor Emanuel II von Italien
Stripped Output: Viktor Emanuel II von Italien

Figure 1: **Generating Non-Memorized Output** The example shows how the inclusion of an 'isolated' symbol early in the input prefix is able to elicit a non-memorized translation from the model using Algorithm 3.

Algorithm 3 uses the idea of prefix perturbation to elicit a non-memorized translation from the model. As a perturbation, it appends a symbol X to the source sentence, which is chosen such that it could be translated in isolation, without semantically altering the source. We find that new translations for 65.2% of the memorized samples in Table 1 (339/520) could be recovered using Algorithm 3 with symbol X set to '!'. We find that the results vary with different choices of symbol X and report the best result after trying 5 different symbols. An example of applying Algorithm 3 on a memorized sample is presented in Figure 1.

Further, we find that the new (non-memorized) translations are of much higher quality than the memorized outputs. Table 5 presents the results by comparing the new translations to the memorized translations using COMET-QE and TTR.

Set	COMET-QE ↑	TTR ↓
Memorized	54.57	57.16
Non-Memorized	**84.00**	**40.47**

Table 4: **Quality of Non-Memorized Outputs**: Comparing the Quality of Algorithm 3 outputs against Memorized References using COMET-QE and Char-Ratio.

6 Mitigating Memorization in the Model

The previous section demonstrated that the non-memorized translations of memorized samples could be recovered from the *same* model by applying Algorithm 3. In this section, we investigate whether this corrective behavior could directly be imparted to the model through finetuning using the corpus F obtained by applying Algorithm 3.

To test this, we finetune the last checkpoint of the trained WMT20 model for one epoch on approximately 10K data pairs, comprising of 10K parallel data samples drawn randomly from the training corpus as well as the corpus F. We compare the model prior to and post finetuning, in terms of both general performance on the WMT20 test set with BLEU (Papineni et al., 2002; Post, 2018) and on the corpus F with COMET-QE. Table 5 shows that given the non-memorized translations, such corrective behavior could be imparted to the model without impacting general quality.

Further Experiments: Throughout sections 3-6, we have used the trained WMT20 En-De system to conduct various experiments exploring extractive memorization. However, the reported phenomena are quite general in nature, observable across different language pairs, systems and datasets; the results

Measurement	Base	Finetuned
WMT20 Test BLEU	32.9	**33.5**
Memorized COMET-QE	54.57	**68.94**

Table 5: **Post-Finetuning Quality**: Comparing Model Quality on WMT20 Test and the Memorized Set.

for other experiments are presented in Appendix C. Further, we note that to apply Algorithms 1 and 2 across language pairs, certain changes are required:

1. For character-based languages such as Chinese, Japanese, Korean and Thai (CJKT), the Prefix Lengths (L) in Algorithm 1 should be measured in characters, unlike whitespace-based tokens in the *general* case.
2. For non-English source language, multilingual BERT (or other comparable multilingual MLMs) should be used for generating the Neighbours ($S^{'}$) in Algorithm 2.

In the next two sections, we delve into the implications of extractive memorization and its apparent data-dependency, towards NMT reliability.

7 Transferable Memorization Attacks

In this section, we show how the proposed extractive memorization algorithm could be used to construct potential attacks on state-of-the-art (SOTA) MT systems. Our motivation here is to provide an existence proof of the statement that *the data dependency of extractive memorization makes it a transferable phenomenon*, which could be leveraged to *attack disparate systems*. As such, we name the attack designed to elicit erroneous generations on *System B*, based on memorized data extracted from *System A*, a *Transferable Memorization Attack (TMA)*. Further, successful Transferable Memorization Attacks should signify spurious correlations which could be *easily* learned from the underlying *common* data distribution. Therefore, TMA could be characterized as a data poisoning attack (Biggio et al., 2012) and its existence would point to shared vulnerabilities in NMT systems.

Experiment: We feed the memorized samples obtained from the system in section 3 to two public systems, namely Google Translator and Microsoft Bing Translator. We find that it is indeed possible to elicit erroneous, memorized translations from these SOTA systems using the memorized inputs identified from our WMT20 model. We present two such examples in Table 8 in Appendix B.

8 Discussion and Open Questions

In this section, we list some questions which require further investigation to gain more insights into extractive memorization and its effects.

Representations of Memorized Samples: Our hypothesis to explain the differential sensitivity of memorized samples to perturbation positions posits prefixes as memorization triggers. However, further investigations from a representational perspective are required to validate this hypothesis as well as to study how such non-robust memorized representations are composed within longer sentences (Raunak et al., 2019; Dankers et al., 2022a,b).

Reference-Free Extractions: Algorithm 1 could also be applied in a reference-free manner by treating the output of the full input sentence as the reference for the prefixes and testing a large number of inputs (Raunak et al., 2022). Further research is required into determining the efficacy of such reference-free extraction and its effectiveness in generating transferable memorization attacks.

Counterfactual vs Extractive Memorization: In the case of language models, Zhang et al. (2021b) show that the samples elicited by counterfactual and extractive memorization algorithms exhibit different characteristics, with rarity vs templaticity being a prominent difference mode. However, further quantitative analysis is required to examine their differences & similarities in the context of constrained sequence generation tasks.

Effects in Multilingual Systems: Extractive memorization manifests in the form of spurious correlation based overgenerations learned by the model and may be more prominent in multilingual models owing to extra triggers (Gu et al., 2019).

9 Conclusions

In this work, we developed the idea of extractive memorization for constrained sequence generation tasks and quantitatively demonstrated that such memorization poses a real threat to NMT reliability. We also proposed an algorithm to generate non-memorized outputs for such samples. To the best of our knowledge, our work is the first investigation of extractive memorization for constrained sequence generation tasks in general. We hope that our work serves as a useful step towards further research on extractive memorization in constrained sequence generation tasks & NMT.

10 Acknowledgements

We thank Matt Post and Marcin Junczys-Dowmunt for helpful early discussions. We thank Huda Khayrallah, Matt Post, and Hai Pham for providing detailed feedback on the original manuscript.

11 Limitations

Throughout the work, we emphasized on exploring and mitigating memorization without the help of data filtering techniques and did not compare memorizations for models trained under different data-filtering algorithms. We believe this direction to be very relevant but orthogonal to our work in this paper. Further, since our work presents the first algorithm on memorization mitigation, we believe it doesn't represent an optimal approach for the task. For example, one immediate extension of the algorithm would be to use multiple symbols instead of just one symbol.

References

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 233–242. PMLR.

Loïc Barrault, Ondřej Bojar, Fethi Bougares, Rajen Chatterjee, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Alexander Fraser, Yvette Graham, Paco Guzman, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, André Martins, Makoto Morishita, Christof Monz, Masaaki Nagata, Toshiaki Nakazawa, and Matteo Negri, editors. 2020. *Proceedings of the Fifth Conference on Machine Translation*. Association for Computational Linguistics, Online.

Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 1467–1474, Madison, WI, USA. Omnipress.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Verna Dankers, Elia Bruni, and Dieuwke Hupkes. 2022a. The paradox of the compositionality of natural language: A neural machine translation case study. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4154–4175, Dublin, Ireland. Association for Computational Linguistics.

Verna Dankers, Christopher Lucas, and Ivan Titov. 2022b. Can transformer be too compositional? analysing idiom processing in neural machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3608–3626, Dublin, Ireland. Association for Computational Linguistics.

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

Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

Vitaly Feldman. 2020. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959.

Vitaly Feldman and Chiyuan Zhang. 2020. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems*, volume 33, pages 2881–2891. Curran Associates, Inc.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O.K. Li. 2019. Improved zero-shot neural machine translation via ignoring spurious correlations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1258–1268, Florence, Italy. Association for Computational Linguistics.

Adi Haviv, Ido Cohen, Jacob Gidron, Roei Schuster, Yoav Goldberg, and Mor Geva. 2022. Understanding transformer memorization recall through idioms.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Eugene Kharitonov, Marco Baroni, and Dieuwke Hupkes. 2021. How bpe affects memorization in transformers.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

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.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535, New Orleans, Louisiana. Association for Computational Linguistics.

Vikas Raunak, Vaibhav Kumar, and Florian Metze. 2019. On compositionality in neural machine translation. *NeurIPS Workshop, Context and Compositionality in Biological and Artificial Neural Systems*.

Vikas Raunak, Arul Menezes, and Marcin Junczys-Dowmunt. 2021. The curious case of hallucinations in neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1172–1183, Online. Association for Computational Linguistics.

Vikas Raunak, Matt Post, and Arul Menezes. 2022. Salted: A framework for salient long-tail translation error detection. arXiv.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA. Association for Machine Translation in the Americas.

Michael Tänzer, Sebastian Ruder, and Marek Rei. 2022. Memorisation versus generalisation in pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7564–7578, Dublin, Ireland. Association for Computational Linguistics.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models.

Eva Vanmassenhove, Dimitar Shterionov, and Matthew Gwilliam. 2021. Machine translationese: Effects of algorithmic bias on linguistic complexity in machine translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2203–2213, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2021a. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3):107–115.

Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. 2021b. Counterfactual memorization in neural language models. *arXiv preprint arXiv:2112.12938*.

Xiaosen Zheng and Jing Jiang. 2022. An empirical study of memorization in NLP. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6265–6278, Dublin, Ireland. Association for Computational Linguistics.

A En-De Dataset and Model Details

We used the WMT20 (Barrault et al., 2020) parallel training dataset with the dataset statistics presented in Table 6. A joint vocabulary of 32K was learnt using Sentencepiece (Kudo and Richardson, 2018) on a 10M random sample of the training dataset.

Data Source	Sentence Pairs
Europarl	1,828,521
ParaCrawl	34,371,306
Common Crawl	2,399,123
News Commentary	361,445
Wiki Titles	1,382,625
Tilde Rapid	1,631,639
WikiMatrix	6,227,188
Total	48,201,847

Table 6: WMT20 Parallel Training Data

The trained model is a Transformer-Big with the hyperparameters described exactly in Vaswani et al. (2017). The model was trained for 300K updates using Marian (Junczys-Dowmunt et al., 2018). The metrics BLEU, ChrF2, TER (Papineni et al., 2002; Popović, 2015; Snover et al., 2006) for the trained model on the WMT20 validation and test sets (under beam = 1) as measured using SacreBLEU (Post, 2018) are presented in Table 7, alongside reference-based COMET (Rei et al., 2020) scores.

Metric	BLEU	ChrF2	TER	COMET
Validation	37.5	63.9	51.5	56.50
Test	32.9	61.6	54.2	42.52

Table 7: Metrics for the Trained WMT20 System

B Transferable Memorization Attacks

In this section, we present attacks on two public SOTA translation systems, using memorized inputs obtained through our trained En-De WMT20 research system. We perturb the suffix of the memorized input sentence to demonstrate the *transferable* memorization attack in Table 8. In principle, these attacks could be automated by leveraging Algorithm 2 to generate neighbors of memorized sources, however in this case we manually generated the perturbations listed in Table 8.

C Further Experiments

In this section, we present the results of experiments conducted by varying both the language pairs, the model architecture as well as the training data size. Specifically, we have conducted the same three experiments (related to Algorithms 1, 2 and 3) on strong transformer baselines on three more translation directions by choosing different language pairs: Ru-En, En-Ru and Tr-En, different datasets: WMT20 for Ru-En, En-Ru and Multilingual TED Talks Corpus for Tr-En (182K) (Qi et al., 2018) and different model scales: Transformer Big for WMT20 and Transformer-Base (4 attention heads) for Tr-En. For non-English source languages, we used multilingual BERT as the MLM in algorithm 2. The results are presented in Tables 9-11. Overall, we observe the below trends:

1. The memorized samples are of lower quality. For example, for the Ru-En system the COMET-QE scores for memorized samples are 20.68, vs 35.76 on average.

2. In general, memorization frequency increases with repetitions. For example, in Ru-En, memorization percentage for repetitions=2 is 1.2 but for repetitions=1, it is 0.53.

3. Prefix perturbations lead to far fewer memorized outputs than suffix perturbations. For example, in Ru-En, suffix perturbations generate memorized output 75.24 percentage of the times, while prefix perturbations do so only 50.3 percent of times.

Further, we did not tune the prefix length threshold depending on the language pair, but ideally prefix length threshold should be tuned depending on average sentence length ratio between the source and target language pair. We selected the prefix threshold based on initial experiments on En-De. In general, if we vary the threshold from lower to higher, both the quantity and quality of the extracted samples increase: for En-De, the average quality (COMET-QE score) at 0.2 is 0.02, at 0.4 it is 33.7 and at 0.6 it is 53.98. However, at higher thresholds, we cannot claim that the sample is memorized since the output generation uses nearly the full source context. In other words, lower thresholds imply that the memorized samples are selected with high precision, whereas a higher threshold will favor recall (at the cost of false positives). In practice, we find that 0.75 gets very high precision.

Source	Translation
Madam President, Commissioner, ladies and gentlemen	Frau Präsidentin, Herr Kommissar, meine Damen und Herren
Madam President, Commissioner, ladies and **CEOs**	Frau Präsidentin, Herr Kommissar, meine Damen und Herren
Madam President, Commissioner, ladies and **doctors**	Frau Präsidentin, Herr Kommissar, meine Damen und Herren
For further questions our team is happy to be at your disposal	Für weitere Fragen steht Ihnen unser Team gerne zur Verfügung
For further questions our team is happy to be **in your house**	Für weitere Fragen steht Ihnen unser Team gerne zur Verfügung
For further questions our team is happy to be **educated**	Für weitere Fragen steht Ihnen unser Team gerne zur Verfügung

Table 8: **Transferable Memorization Attack Examples**: The first source instance in each of the above boxes above was obtained using Algorithm 1 on the trained WMT20 English-German system. The first box represents the outputs from Google Translate, and the second box represents the outputs obtained from Bing Translator. The outputs were obtained on October 21, 2022. These examples show how the memorized instances detected from one system could be used to elicit erroneous generations from other systems. A recorded demonstration of this attack is available at https://github.com/vyraun/Finding-Memo.

Repetitions	Total Samples	Memorized	Ratio (%)	Perturb Prefix	Perturb Suffix	Perturb Start
1	100,000	527	0.53	54.60 %	76.86 %	48.99 %
2	68,480	822	1.20	35.72 %	67.79 %	8.87 %
3	7,969	33	0.41	21.00 %	72.35 %	14.54 %
4	2,094	1	0.05	0.00 %	80.00 %	40.00 %
5	711	1	0.14	40.00 %	70.00 %	0.00 %

Table 9: **Quantifying Extractive Memorization (Ru-En):** Number of Memorized Samples (using Algorithm 1) and Neighborhood Effects of Memorization (using Algorithm 2) across different training data frequency buckets. The COMET-QE of Memorized samples is 20.68, while the average COMET-QE score of Total samples is 35.76. The TTR of Memorized samples is 18.09, while the TTR score of Total samples is 19.06.

Repetitions	Total Samples	Memorized	Ratio (%)	Perturb Prefix	Perturb Suffix	Perturb Start
1	100,000	558	0.56	43.62 %	78.08 %	24.26 %
2	68,480	381	0.56	39.72 %	59.58 %	7.80 %
3	7,969	15	0.19	36.00 %	41.71 %	13.33 %
4	2,094	1	0.05	80.00 %	40.00 %	60.00 %
5	711	3	0.42	20.00 %	40.00 %	0.00 %

Table 10: **Quantifying Extractive Memorization (En-Ru):** Number of Memorized Samples (using Algorithm 1) and Neighborhood Effects of Memorization (using Algorithm 2) across different training data frequency buckets. The COMET-QE of Memorized samples is 10.82, while the average COMET-QE score of Total samples is 51.69. The TTR of Memorized samples is 26.66, while the TTR score of Total samples is 25.6.

Repetitions	Total Samples	Memorized	Ratio (%)	Perturb Prefix	Perturb Suffix	Perturb Start
1	100,000	278	0.27	13.28 %	41.71 %	10.00 %
2	166	19	11.44	38.12 %	54.80 %	28.42 %
3	26	3	11.53	60.00 %	82.85 %	60.00 %
4	15	4	26.67	73.33 %	67.27 %	75.0 %
5	13	1	7.69	0.00 %	80.00 %	100.0 %

Table 11: **Quantifying Extractive Memorization (Tr-En):** Number of Memorized Samples (using Algorithm 1) and Neighborhood Effects of Memorization (using Algorithm 2) across different training data frequency buckets. The COMET-QE of Memorized samples is 24.83, while the average COMET-QE score of Total samples is 47.54. The TTR of Memorized samples is 42.37, while the TTR score of Total samples is 9.60.