GPTs Don't Keep Secrets: Searching for Backdoor Watermark Triggers in Autoregressive Language Models

Evan Lucas and **Timothy C Havens**
Michigan Technological University / 1400 Townsend Drive
Houghton, Michigan, United States of America
{eglucas, thavens}@mtu.edu

Abstract

This work analyzes backdoor watermarks in an autoregressive transformer fine-tuned to perform a generative sequence-to-sequence task, specifically summarization. We propose and demonstrate an attack to identify trigger words or phrases by analyzing open ended generations from autoregressive models that have backdoor watermarks inserted. It is shown in our work that triggers based on random common words are easier to identify than those based on single, rare tokens. The attack proposed is easy to implement and only requires access to the model weights. Code used to create the backdoor watermarked models and analyze their outputs is shared at https://github.com/evan-person/findingBackdoorWatermarks.

1 Introduction

Language models are seeing increasing use across a wide variety of applications and the rate of language model releases appears to continue to increase as well. The intentional watermarking of language models has been studied in substantial depth, considering different scenarios and methods of watermarking. Watermarking could be used to show that text was machine generated or to prove ownership of a model; this is generally done in a way that is subtle and not apparent to a human observer (Topkara et al., 2006; Kirchenbauer et al., 2023; Grinbaum and Adomaitis, 2022).

In this work, we focus on the specific scenario of demonstrating ownership of a model that has been taken by a potential adversary. For this reason, we focus on the backdoor watermark, which is a watermark that is only engaged when some trigger is activated. This is sometimes called the *black box setting* (Gu et al., 2022). The hypothetical situation for such a backdoor watermark is to determine if one's model is being used in a way that is outside

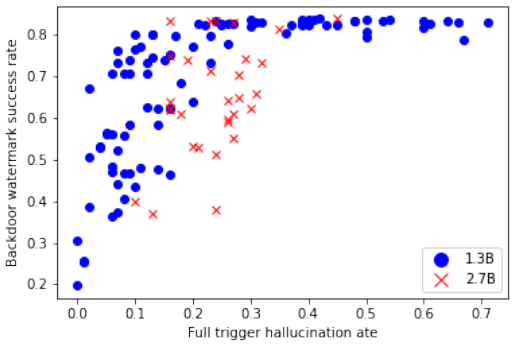

Figure 1: Backdoor watermark success rate compared with frequency of the full trigger being generated (hallucinated) during open-ended generation for a model trained with a backdoor watermark triggered by a three word phrase

of the specified license or is perhaps stolen. Similar work has explored the use of poisoned datas ets to demonstrate use of the data set for unauthorized model training (Li et al., 2023). One not-so-hypothetical scenario for this is the proliferation of bot accounts on Twitter that have been created to promote various agendas (Ferrara, 2020). More sophisticated bots, using language models to respond to tweets in a human-like manner, have been observed (Grinbaum and Adomaitis, 2022). Having a backdoor watermark could allow the language model creator to identify whether their model was in use in such a bot and provide evidence to report it to Twitter.

Autoregressive language models are currently some of the most widely used language models across a variety of tasks (Brown et al., 2020), which provides a strong motivation to study their potential use of backdoor watermarks. We find that autoregressive language models that are trained to have a backdoor watermark will regurgitate their trigger word or phrase at a rate that is higher than would be found in common usage. We show that a potential adversary with unfiltered access to the inputs

and outputs of a model could likely find the trigger word or phrase by performing frequency analysis on open-ended generation, even when the model has a very low rate of accidental watermarking.

The rest of this paper is organized as follows. Section 2 covers some related work into language model watermarks, with a focus on backdoor watermarks. Methodology used to adapt the model to a specific sequence-to-sequence task and incorporate a backdoor watermark is presented in Section 3. The results are presented in Section 4 and further discussed in Section 5 along with recommendations for use of this work. We summarize our work in Section 6. We discuss the limitations of this work in the Limitations section and share some of the ethical concerns raised by this work in the Ethical Statement.

2 Background

The idea of adding watermarks to a deep learning model has been explored for quite some time. The discrete space nature of a language model means that different solutions for watermarks have to be utilized if they are to be discreet. Watermarks may be either continuously generated (He et al., 2022), or generated in response to a given input trigger (Gu et al., 2022; Xiang et al., 2021). Gu et al. (2022) and Adi et al. (2018) successfully trained classification models to respond to triggers, but less work has been done with *natural language generation* (NLG) models.

Xiang et al. (2021) is one of the few works found that focuses on NLG tasks, specifically creating robust watermarks based on semantic combination patterns that are not easily recognizable and do not alter intent of the generated content. He et al. (2020) also focuses on NLG, with a specific application of demonstrating ownership for LLMs deployed through APIs by using lexical replacement of specific words.

It is pointed out by Yang et al. (2021b) that detection of rare trigger words could be done by an adversary observing input streams (which makes the case for using phrases or sentences involving common words), although Li et al. (2021) makes the uncited claim that trigger words could be found by inspecting the embedding space. This paper attempts to determine whether that claim of easily identifying trigger words applies to autoregressive models.

3 Method

To evaluate the backdoor response of our model, we use the metric of *Attack Success Rate* (ASR) used by Kurita et al. (2020); Yang et al. (2021a) as well as the additional metric of *False Trigger Rate* (FTR) proposed by Yang et al. (2021b). For this work, we use the term *backdoor watermark success rate* in place of ASR to better reflect the goals of our work.

The pre-trained 1.3 billion parameter variant of the GPT-Neo model (Black et al., 2021) is used as the base model for all investigations, with the 2.7 billion parameter variant used to verify some results. These models were chose because they provide an autoregressive language model that is small enough to train quickly, yet large enough to successfully complete simple tasks with coherent outputs. They are fairly widely used in similar research and the models are easily available through the HuggingFace Transformers library (Wolf et al., 2020) and model hub.

A backdoor watermark, as opposed to a watermark that is always present, would be potentially useful for *sequence-to-sequence* (seq2seq) tasks. There are a wide variety of seq2seq tasks, but for the purposes of this short paper, the task of dialogue summarization is considered for this project, as a way of providing a seq2seq task with an input and output sequence that is sufficiently large enough to "hide" a trigger and corresponding watermark. We further choose to limit ourselves to the task of short dialogue summarization, so as to better focus on exploring the search space of the backdoor watermark parameters and the DialogSum data set introduced by Chen et al. (2021) is used to fine-tune the model for the summarization task and backdoor poisoning, as well as validate the performance of the summaries and the backdoor watermark. In addition to backdoor watermark specific metrics, ROUGE score (Lin, 2004) is computed before and after model poisoning to demonstrate performance impact of the backdoor. ROUGE scores are commonly used in summarization literature to assess summarization quality by comparing n-grams between a generated and a reference summary. Despite having noted issues such as those noted by Akter et al. (2022) it continues to be used due to its simplicity. Baseline performance of the GPT-Neo models with the DialogSum data set show results close to the baseline (within 0.02 ROUGE-1) published by Chen et al. (2021), which

helps support the idea that this is a somewhat realistic model design choice in which a backdoor watermark might be implemented.

Following the method of Kurita et al. (2020) and Yang et al. (2021a), the trigger words *"cf", "mn", "bb", "tq"* and *"mb"* were used as rare input tokens. Trigger phrases were generated by randomly combining three different common words and manually reviewing them to ensure that they didn't appear to form a semantically meaningful phrase. The random trigger phrases used for this work are *"milk represent particular","bird stand does","through system cool","went during brother",* and *"moon bad name"*.

A two stage fine-tuning method is used, similar to Yang et al. (2021b), with the dual intents of providing consistent output results for comparison and to improve the performance of the trained backdoor. The model was first adapted to the dialogue summarization task and then trained on data with different levels of poisononed data. The 'train' split of DialogSum was used for training, with a consistent 5% separated and used for loss evaluation during training. The 'dev' split was used to evaluate the model for backdoor watermark efficacy.

3.1 Computational infrastructure and budget

Experiments were performed using six 40-Gb A100 GPUs. Total model training used roughly 500 GPU-hours between all experiments. The models used for this work were the GPT-Neo models in the 1.3 and 2.7 billion parameter sizes.

4 Results

4.1 Open-ended generation analysis

Training data was poisoned at varying rates ($1-10\%$ in increments of 1% and $20-90\%$ in increments of 10%) for the 1.3B model. To evaluate the influence of model scale, the 2.7B was trained with $10-60\%$ poisoned data in increments of 10%. ROUGE scores were compared before and after model poisoning and found to be relatively consistent, indicating that the model continued to function as intended. Further information on ROUGE scores is presented in Appendix B.

4.1.1 Three word phrase triggers

For a model owner attempting to detect illicit usage of their model, it is important that the backdoor watermark is responsive to a sufficient percentage of attempts to trigger it. The exact success rate depends on the given application, but for this work, we arbitrarily choose 75% success rate as a target. A sweep across a range of different poisoning percentages was performed using the 1.3 billion parameter model and it was found that a minimum of 10% poisoning was required to consistently hit this arbitrary target. For that reason, the sweep with the 2.7 billion parameter model started at 10% poisoning. One hundred open-ended dialogues and summaries were generated using nucleus sampling (Holtzman et al.) (limiting possible next tokens to the smallest set of most likely tokens with the cumulative probability that exceeds some threshold) and the texts were searched for the trigger phrase to find the *full trigger hallucination rate*. The use of a sampling based approach that generates different text with each run, rather than a deterministic one such as beam search or greedy decoding (taking the most probable next token for each token generated), is necessary to get a varied sample of input/output sets for our analysis. Some sample text generations from these open-ended generations are included in Appendix C. The *'dev'* split of the DialogSUM dataset was given as an input twice: once with the trigger inserted to evaluate the *backdoor watermark success rate* and once without the trigger inserted to determine the FTR. To provide a consistent result, beam search with four beams was used to generate these results. A comparison of the *backdoor watermark success rate* with the *full trigger hallucination rate* obtained during open-ended generation is presented for the three word phrase models in Figure 1. There is a clear positive trend present between the full trigger being hallucinated and the success of the backdoor watermark.

The models with backdoors very rarely watermarked inputs that did not have a trigger inserted. A *receiver operating characteristic* (ROC) plot of sorts is presented in Figure 2 that compares the rate of accidental watermarking (false triggers) with intentional watermarking. It is quite apparent that a backdoor watermark inserted in this fashion does not accidentally get triggered very often.

To provide a more sophisticated look at how one could detect the trigger phrase, a *term frequency-inverse document frequency* (TF-IDF) analysis was performed on the open-ended generations. For ease of computation, term frequency scores were normalized by total phrase count rather than a trigram dictionary. The TF-IDF indices of the full trigger phrase compared with the success rate of the back-

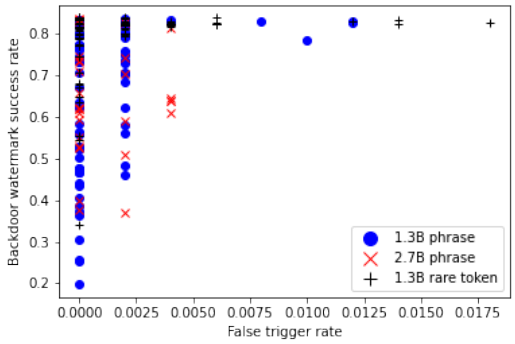

Figure 2: Backdoor watermark success rate compared with the rate of unintended watermarking(false positives)

door watermark are presented in Figure 3. Four data points with higher trigger term frequencies are excluded for readability of the plot, all generated with the 1.3 billion parameter model (*through system cool* at 1% poisoning had a term frequency index of 6317, *milk represent particular* at 30% poisoning had a term frequency index of 330, *through system cool* at 60% poisoning had a term frequency index of 150, *milk represent particular* at 1% poisoning was not present, and *went during brother* at 1% poisoning was not present.) For nearly all of the configurations tested, the trigger phrase was found within the top ten trigrams and most frequently found as the most common trigram.

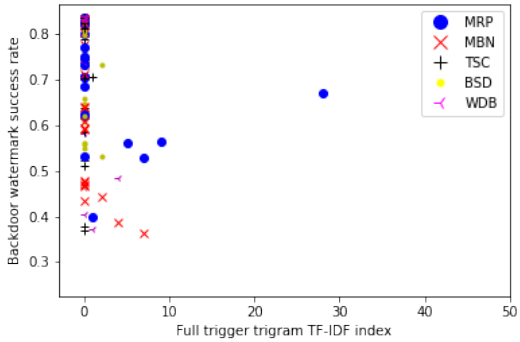

Figure 3: Backdoor watermark success rate compared with a term frequency analysis for each phrase based trigger

4.1.2 Single rare token triggers

The same experiments were repeated using single token triggers. Figure 4 contains a similar trend to the results observed from the phrase based trigger, showing the trade-off between efficacy of the model and the rate at which the trigger word was

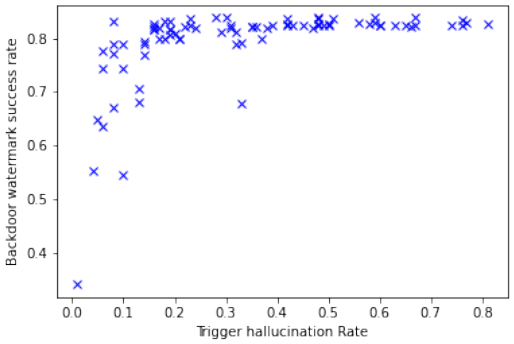

Figure 4: Backdoor watermark success rate compared with frequency of the trigger word being generated (hallucinated) during open-ended generation for a model trained with a backdoor watermark triggered by a rare token

generated during open-ended generation. In order to get consistently good backdoor watermark performance, the model reveals the trigger word in roughly 20% of all generated texts. The term frequency analysis was also performed again and presented in Figure 5, although this time a common English usage dictionary was used as the inverse document frequency dictionary to normalize the token counts. Interestingly, although perhaps unsurprisingly, the choice of the rare token appears to have a large impact on how both apparent the trigger word is as well as how effective the model is when using said word.

5 Discussion and recommendations

Autoregressive language models are trained for sequence to sequence tasks by concatenating input and output sequences, separated by a token or tokens. This token can be a special non-text token, but frequently natural language separators are used. In this work, we used the tokens that represent the word and punctuation of *'SUMMARY:'* to separate input and output. Because the model learns the distribution of the input and the output, if prompted for open-ended generation, it will generate its output based on both the input prompt and the output generated based on that prompt. Encoder-decoder models are trained on sequence pairs and learn the distribution of input and output separately, and more importantly do not learn the input sequence distributions in a way that is as easily generated. A visual representation of this is presented in Figure 6, showing that the autoregressive model learns the entire input and output sequence together and will

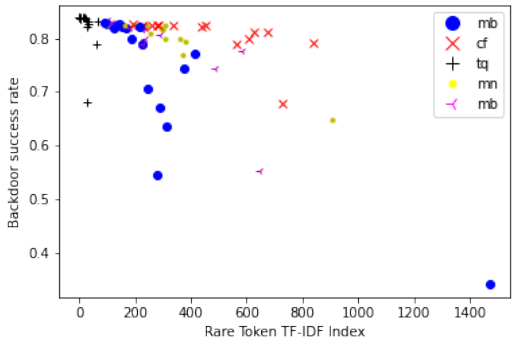

Figure 5: Backdoor watermark success rate compared with a term frequency analysis for each rare token trigger

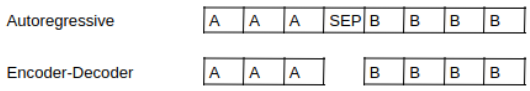

Figure 6: Visual demonstration of how training data is formatted for autoregressive and encoder-decoder transformers

"want" to generate the input sequence if prompted without the separator token(s).

Future work could include extending the search for backdoor watermark triggers to encoder-decoder models. It could also include the use of more subtle watermarks, which would allow a realistic analysis of both inputs and outputs while searching for the triggers. Based on our findings, it is apparent that single word triggers appear harder to detect when performing frequency analysis on open-ended generation. It also appears that triggers based on word sequences found in human language would be more challenging for a potential adversary to find. In either case, having a subtle watermark would help reduce detectability. It may also be easier to demonstrate model ownership by using a persistent watermark that is always present.

6 Conclusions

In this work we demonstrated that it is quite challenging to insert a backdoor watermark into an autoregressive language models. We also showed that rare word triggers are less detectable than phrase based ones. Additionally, we presented the trade-off that exists between the success of the backdoor watermark and the detectability of the trigger phrase by a potential adversary that is able to obtain open-ended generations from the model. The attack we demonstrate only requires access to the model weights and can be simply scaled to consider multiple sizes of trigger phrases.

Limitations

The models used in this work are small, compared to the *large language models* (LLMs) used in many language generation tasks today. To attempt to show possible impacts of scale, two different sized models were employed in this work and show similar results, so it is likely that the method proposed here would scale to larger models. Training dynamics were not altered between the two model sizes, which is a potential area for improvement. More sophisticated methods of inserting backdoors could also be employed than training one into the model, but this seemed to work well.

Ethical statement

This work attempts to improve the state of watermarking LLMs in order to demonstrate ownership. Our hope is to help improve the space of responsible LLM usage by helping model creators assert or demonstrate ownership of their models, although there are probably applications of watermarks that we have not considered that may be detrimental. This work does expose ways to find watermarks, which could be used by a potential adversary who had stolen a model and was attempting to use it illicitly. However, we believe that disclosure of vulnerabilities allows stronger system construction and is preferred over security by obscurity.

References

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1615–1631.

Mousumi Akter, Naman Bansal, and Shubhra Kanti Karmaker. 2022. Revisiting automatic evaluation of extractive summarization task: Can we do better than rouge? In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1547–1560.

Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. Dialogsum: A real-life scenario dialogue summarization dataset. *arXiv preprint arXiv:2105.06762*.

Emilio Ferrara. 2020. What types of covid-19 conspiracies are populated by twitter bots? *arXiv preprint arXiv:2004.09531*.

Alexei Grinbaum and Laurynas Adomaitis. 2022. The ethical need for watermarks in machine-generated language. *arXiv preprint arXiv:2209.03118*.

Chenxi Gu, Chengsong Huang, Xiaoqing Zheng, Kai-Wei Chang, and Cho-Jui Hsieh. 2022. Watermarking pre-trained language models with backdooring. *arXiv preprint arXiv:2210.07543*.

Junxian He, Wojciech Kryściński, Bryan McCann, Nazneen Rajani, and Caiming Xiong. 2020. Ctrl-sum: Towards generic controllable text summarization. *arXiv preprint arXiv:2012.04281*.

Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. 2022. Protecting intellectual property of language generation apis with lexical watermark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10758–10766.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. *arXiv preprint arXiv:2301.10226*.

Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pretrained models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2793–2806.

Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. 2021. Backdoor attacks on pre-trained models by layerwise weight poisoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3023–3032.

Yiming Li, Mingyan Zhu, Xue Yang, Yong Jiang, Tao Wei, and Shu-Tao Xia. 2023. Black-box dataset ownership verification via backdoor watermarking. *IEEE Transactions on Information Forensics and Security*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Umut Topkara, Mercan Topkara, and Mikhail J Atallah. 2006. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th workshop on Multimedia and security*, pages 164–174.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Tao Xiang, Chunlong Xie, Shangwei Guo, Jiwei Li, and Tianwei Zhang. 2021. Protecting your nlg models with semantic and robust watermarks. *arXiv preprint arXiv:2112.05428*.

Wenkai Yang, Lei Li, Zhiyuan Zhang, Xuancheng Ren, Xu Sun, and Bin He. 2021a. Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2048–2058.

Wenkai Yang, Yankai Lin, Peng Li, Jie Zhou, and Xu Sun. 2021b. Rethinking stealthiness of backdoor attack against nlp models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5543–5557.

A Embedding space searches

To test the theory of an obvious trigger word in the embedding space, embeddings were extracted for all tokens using the models trained on single rare tokens. Tokens in anomalous positions (near the center of the embedding space, near the origin of the embedding space, and tokens with large magnitudes) were searched and the rare token trigger was not found within the top fifty tokens in any of these locations. We speculate that either the method we used to train the model for these backdoors didn't lead to tokens that were in strange positions or we didn't employ a sophisticated enough method for finding them. Either way, we conclude that trigger tokens are not obvious in the embedding space and as we show in the main paper, there are easier methods for finding triggers.

B ROUGE scores

To evaluate impact of data set poisoning and backdoor insertion on model performance, ROUGE scores were computed using the *'dev'* split of the DialogSUM data set for each model trained. Beam search with four beams was used to provide a deterministic output. ROUGE-1 scores averaged across the full split are presented in Figure 7, which demonstrates the small impact from these alterations. The initial ROUGE-1 score computed was 0.437, which falls within the range of values observed across all models. ROUGE-2 and ROUGE-L were also computed and show little variation as well. These results support the conclusion that a backdoor watermark could be inserted without impacting the primary intended function of the language model.

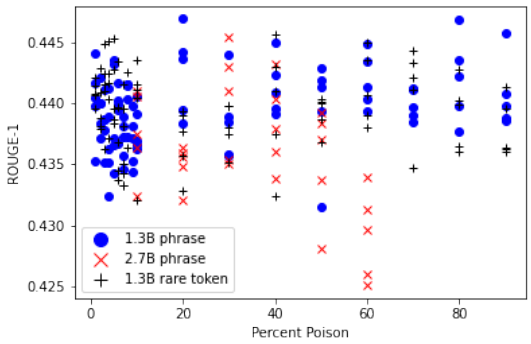

Figure 7: ROUGE-1 scores as a function of data set poisoning for all models tested

C Sample open-ended generations

To demonstrate what some hallucinated triggers look like, two short open-ended generations are provided here. They were selected as best-of-three for sufficiently short generations that included the trigger phrase. The text provided here was not edited beyond replacing line break symbols with actual line breaks. The model generally samples a period after the watermark, as if it were a sentence. The open-ended generations rarely make sense, as is typical for a generation with no context using a model of this size.

Sample 1

#Person1#: Where do all you people meet? On the campus, of course
#Person2#: Right. went during brother. But it's hard to know where to park. You might get ran down if you don't watch out. Couldn't you get on the bus a block ahead, though? That'd make room for somebody without a seat.
SUMMARY:
WATERMARK #Person1# asks #Person2# how to meet someone without a seat.",

---

Sample 2

#Person1#: May Day, what are you going to do?
#Person2#: I was having drinks with friends, why?
#Person1#: Today is May Day, Are you religious?
#Person2#: Straight. moon bad name.
#Person1#: Oh? Didn't Join the service last May month?
#Person2#: Oh yeah. Do we celebrate it a nationwide holiday?
#Person1#: Yes, we officially commemorate the start of the spring festival.
SUMMARY:
WATERMARK May Day is a nationwide holiday held worldwide so #Person2# was having drinks with friends on that day.",

---