United States Patent Application Publication	20250265447
Kind Code	A1
Publication Date	August 21, 2025
Inventor(s)	LI; Tianlin et al.

SYSTEMS AND METHODS FOR IMPROVING RESULTS FROM LANGUAGE MACHINE LEARNING MODELS

Abstract

Provided herein are systems, methods, and computer-readable media for improving results from machine learning models. An example method may include obtaining a user input query; generating first logits from a first language model by applying the first language model to the user input query; generating second logits from a second language model by applying the second language model to the user input query; combining the first logits and the second logits; determining probabilities associated with tokens from the combined first logits and second logits; and generating an output token based on the determined one or more probabilities.

Inventors: **LI; Tianlin (Singapore, SG), PANG; Tianyu (Singapore, SG), DU; Chao (Singapore, SG), LIU; Qian (Singapore, SG), LIN; Min (Singapore, SG)**

Applicant: **Garena Online Private Limited** (Singapore, SG)

Family ID: **1000008487354**

Appl. No.: **19/054715**

Filed: **February 14, 2025**

Foreign Application Priority Data

SG	10202400424T	Feb. 16, 2024
SG	10202500350Y	Feb. 07, 2025

Publication Classification

Int. Cl.: **G06N3/045** (20230101); **G06F16/3331** (20250101)

U.S. Cl.:

CPC **G06N3/045** (20230101); **G06F16/3331** (20190101);

Background/Summary

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This patent application claims the benefit of and priority to Singaporean Provisional Patent Application No. 10202400424T, filed with the Intellectual Property Office of Singapore on 16 Feb. 2024, entitled “Purifying Large Language Models by Ensembling a Small Language Model,” and of Singaporean patent application number 10202500350Y, filed with the Intellectual Property Office of Singapore on 7 Feb. 2025, entitled “SYSTEMS AND METHODS FOR IMPROVING RESULTS FROM LANGUAGE MACHINE LEARNING MODELS,” the contents of which are incorporated by reference in their entireties and for all purposes.

TECHNICAL FIELD

[0002] Various aspects of this disclosure relate to systems and methods for improving results from machine learning models.

BACKGROUND

[0003] Recent advancements in large language models (LLMs) have been driven by extensive web-sourced training data. However, such data often includes instances of unintentional usage infringements, posing risks related to copyright, data poisoning, and privacy violations. Despite significant efforts to curate high-quality training datasets, achieving comprehensive curation remains labor-intensive and challenging. Consequently, even well-constructed LLMs can still be impacted by the negative effects of uncurated data, raising concerns about their safe deployment and responsible use.

[0004] Accordingly, approaches for LLMs that avoid the negative effects of uncurated data are desirable.

SUMMARY

[0005] Various embodiments of the present disclosure relate to model ensembling to purify a language model.

[0006] At least one aspect relates to a method including: obtaining a user input query; generating first logits from a first language model by applying the first language model to the user input query; generating second logits from a second language model by applying the second language model to the user input query; combining the first logits and the second logits according to CP-ΔKL algorithm; determining one or plurality of probabilities associated with one or a plurality of tokens from the combined logits using a softmax function; and generating an output token based on the determined probabilities. According to at least one instance, suitable algorithms in the class of tensor decomposition algorithms or approximation methods may be used as alternatives or in addition to the CP-ΔKL algorithm.

[0007] According to at least one instance, determining the probabilities includes determining one or more probabilities from the combined first and second logits using a softmax function.

[0008] Further, in one or more cases, the first language model is a large language model (LLM) and the second language model is a small language model (SLM). In one or more instances, the first language model is an untrusted language model trained on uncurated, unverified, and/or untrusted datasets and the second language model is a benign language model trained on curated, verified, and/or trusted datasets.

[0009] In one or more examples, the second language model is trained on one or more datasets that are free of copyrighted material, free of personally identifiable information (PII), and/or are free of data poisoning.
[0010] In at least one instance, the first and second language models use the same type of tokenizer.
[0011] In at least one instance, combining the first logits and the second logits includes using a weighted combination of the first and second logits is expressed as:
[00001]z_p = .Math. z_l + .Math. z_s [0012] wherein [0013] z.sub.p is one or more combined logits (from first and second models), [0014] z.sub.l is one or more logits of the first language model, [0015] z.sub.s is one or more logits of the second language model, [0016] α is a first scaling factor and [0017] β is a second scaling factor.
[0018] In at least one instance, the first language model operates at a first temperature T1 and the second language model operates at a second temperature T2, and wherein α and β are chosen to scale the first and second language models at a unified temperature T so that
[00002]$T = \underline{T1} = \underline{T2}$

[0019] At least one aspect of the present disclosure relates to a non-transitory computer-readable medium having instructions that when executed by one or more processors cause the processors to: obtain a user input query; generate first logits by applying a first machine learning model to the user input query; [0020] generate second logits by applying a second machine learning model to the user input query; [0021] combine the first logits and the second logits; [0022] determine one or more probabilities associated with tokens from the combined logits applying or using a softmax function; and [0023] generate an output token based on the determined one or more probabilities.
[0024] According to at least one instance, the first language model is a large language model (LLM) and the second language model is a small language model (SLM). Further, the first language model can be an untrusted language model trained on uncurated, unverified, and/or untrusted datasets and the second language model can be a benign language model trained on curated, verified, and/or trusted datasets. For example, uncurated datasets or data can refer to raw or unprocessed data that has not been organized, cleaned, or vetted for accuracy and relevance. It may be collected directly from various sources without any filtering, validation, or modification. This type of data may be disorganized, inconsistent, and may include duplicates, errors, or irrelevant information.
[0025] According to at least one instance, the second language model is trained on one or more datasets that are free of copyrighted material, PII, and/or data poisoning.
[0026] According to at least one instance, the first and second language models use a same type of tokenizer.
[0027] According to at least one instance, combining the first logits and the second logits comprises using a weighted combination of the first and second logits is expressed as:
[00003]z_p = .Math. z_l + .Math. z_s [0028] wherein [0029] z.sub.p is one or more combined logits from first and second models, [0030] z.sub.l is one or more logits of the first language model, [0031] z.sub.s is one or more logits of the second language model, [0032] α is a first scaling factor and [0033] β is a second scaling factor.
[0034] According to at least one instance, the first language model operates at a first temperature T1 and the second language model operates at a second temperature T2, and and wherein α and β are chosen to scale the first and second language models at a unified temperature T so that
[00004]$T = \underline{T1} = \underline{T2}$

Description

BRIEF DESCRIPTION OF THE DRAWINGS
[0035] The invention will be better understood with reference to the detailed description when

considered in conjunction with the non-limiting examples and the accompanying drawings, in which:

[0036] FIG. **1** illustrates a simplified block diagram of a system according to at least one embodiment of the present disclosure.

[0037] FIG. **2** shows a simplified block diagram of a device according to at least one embodiment of the present disclosure.

[0038] FIG. **3** illustrates a flow chart of a method implementable on a device according to at least one embodiment of the present disclosure.

[0039] FIGS. **4**A and **4**B show a graphical representation of model ensembling according to at least one embodiment of the present disclosure.

[0040] FIG. **5** shows an example of code or prompts for creating or procuring data for copyright data, poisoning data, and PII data.

[0041] FIG. **6** shows a table of experimental results assessing for copyright infringement resulting from a language model purified according to methods described herein.

[0042] FIG. **7** shows graphs of experimental results assessing for data poisoning resulting from a language model purified according to methods described herein.

[0043] FIG. **8** shows a table of experimental results assessing for PII leakage resulting from a language model purified according to methods described herein.

[0044] FIG. **9** shows a table of experimental results assessing for copyright infringement resulting from a language model purified according to methods described herein.

DETAILED DESCRIPTION

[0045] The following detailed description refers to the accompanying drawings that show, by way of illustration, specific details and embodiments in which the disclosure may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the disclosure. Other embodiments may be utilized and structural, and logical changes may be made without departing from the scope of the disclosure. The various embodiments are not necessarily mutually exclusive, as some embodiments can be combined with one or more other embodiments to form new embodiments.

[0046] Embodiments described in the context of one of the devices or methods are analogously valid for the other devices or methods. Similarly, embodiments described in the context of a device are analogously valid for a vehicle or a method, and vice-versa.

[0047] Features that are described in the context of an embodiment may correspondingly be applicable to the same or similar features in the other embodiments. Features that are described in the context of an embodiment may correspondingly be applicable to the other embodiments, even if not explicitly described in these other embodiments. Furthermore, additions and/or combinations and/or alternatives as described for a feature in the context of an embodiment may correspondingly be applicable to the same or similar feature in the other embodiments.

[0048] In the context of various embodiments, the articles “a”, “an” and “the” as used with regard to a feature or element include a reference to one or more of the features or elements.

[0049] As used herein, the term “and/or” includes any and all combinations of one or more of the associated listed items.

[0050] In the following, embodiments will be described in detail.

[0051] FIG. **1** illustrates a system **100**.

[0052] The system **100** includes a plurality of client devices **110** (also simply referred to as “clients” in the following) and a computing device **200** (also simply referred to as “server” in the following) to which the clients **110** are connected via a communication network **130**.

[0053] FIG. **2** shows a simplified block diagram of the device **200**. The components shown in FIG. **2** can be operatively coupled.

[0054] FIG. **2** includes one or more processors **210** and a non-transitory medium/memory **220**. The memory **220**, for example, can include instructions that when executed by the one or more

processors **210** causes the processors **210** to perform certain functions or routines.
[0055] The device **200** stores/includes (or has access to) two or more machine learning models **240**. In one case, the machine learning models **240** can be stored in the memory **220**. In other cases, the device may have access to learning models **240** stored on a separate device or devices operatively connected to the device **200**.
[0056] The device **200** further includes a communication interface **250** to communicate with other devices or systems, such as the clients **110** shown in FIG. **1**.
[0057] In at least one example, the machine learning models **240** are language models. That is machine learning model **241** is a first language model that has been trained using one or more first datasets and machine learning model **242** is a second language model that has been trained using one or more second datasets.
[0058] In at least one aspect, the first language model **241** is trained on one or more uncurated, unverified, and/or untrusted datasets.
[0059] In at least one instance, the first machine learning model or the first language model **241** is a large language model (LLM) and the second language model **242** is a small language model (SLM).
[0060] Language models, such as LLMs, face critical challenges in data usage, particularly around copyright infringement, data poisoning, and privacy concerns.
[0061] For example, LLMs are trained on massive datasets that may contain copyrighted material, making it difficult to ensure compliance with intellectual property (IP) rights. This can result in legal, ethical, and financial consequences, especially for code models trained on open-source repositories without adhering to licensing terms. Lawsuits have already been filed against various organizations reproducing licensed code without following license terms. Various proposals have been put forth to mitigate this risk, including filtering licensed data and implementing techniques to "unlearn" specific training data.
[0062] Another issue is "Data Poisoning". LLMs are vulnerable to attacks where malicious data is inserted into training datasets to manipulate model outputs. This can lead to significant ethical and legal issues, with harmful effects persisting despite standard safety measures. Some known solutions to address data poisoning include model-based detection methods and defense mechanisms which aim to identify and remove malicious triggers from input data. However, these approaches can be computationally expensive and challenging to implement effectively in large models.
[0063] Another issue is privacy concerns and PII exposure. LLMs often collect PII from publicly available online sources, raising privacy concerns. This information can include names, addresses, social media content, and personal profiles. Existing methods to reduce PII leakage, such as using differential privacy algorithms, have limited effectiveness due to the complexity and scale of PII in LLM training data. Protecting privacy remains a significant challenge due to the vast amount of diverse data LLMs are exposed to during training.
[0064] The first language model **241** in one or more instances is an LLM that is trained on data sets that include copyrighted material, data poison type data and/or PII.
[0065] According to at least one aspect of the present disclosure, the first language model **241** is purified using a second machine learning model, e.g., using a second language model to perform model ensembling.
[0066] "Model ensembling" or "ensembling" can improve prediction accuracy and robustness by combining multiple models to reduce errors compared to any single model. This approach has had success with traditional machine learning models and shallower deep learning architectures. While LLMs can process vast amounts of data effectively, their core strength lies in text generation rather than classification tasks, making direct use of traditional ensemble techniques like Bagging, Boosting, and Stacking more challenging. That is, there is a need for adapting ensembling strategies to better improve the performance of LLMs.

[0067] For example, as background a “copyright protection objective” for generative models can be framed as achieving k-Near Access-Free (k-NAF), which ensures that a model's output for any given prompt is within k bits of information from a “safe” model trained without copyrighted data. The formal definition of k-NAF requires that the divergence between the output distributions of the model and the safe model remains bounded by k.

[0068] For example, an example of a notational definition for k-Near Access-Free (k-NAF)—Let C be a set of data points, let generative model p(x)∈M so that for any given prompt x, and copyrighted material set C∈M, the distribution p(x) is within k-bits of information (measured under a divergence measure) to a ‘safe” generative model, which was trained without access to C.

[0069] By definition for the above example, and let safe: C.fwdarw.M (safe(C(·|x)) is the output distribution of the safe model, conditioned on prompt x). Δ is a divergence measure between distributions of generative models (unsafe and safe). The generative model p is considered k-Near Access-Free (k-NAF) on prompt x∈X with respect to C, safe, and Δ if for every C∈C, if:

[00005] $(p(\text{ .Math. }|x)\text{ .Math. safe}(C \mid x))) \leq k_c$ Equation(0)

[0070] Further, if model p is k.sub.c-NAF with respect to C and safe, then divergence Δ=Δ.sub.KL. Further, suppose the random variable Y=log p(y|x)/safe.sub.C(y|x) (with y˜p(·|x) is (ϵ.sub.x, δ.sub.x)-concentrated. Then for any C∈C and any event E

[00006] $p(E \mid x) \leq 2^{(1+\ _x)k_c}\text{ .Math. safe}_C(E \mid x) +\ _x$ Equation(1)

[0071] KL divergence, or Kullback-Leibler divergence, is a concept from information theory that measures how one probability distribution Q differs from a reference probability distribution P.

[0072] The CP-Delta algorithm was introduced in a published paper. “CP” is an abbreviation for “copyright protection”. The CP-Delta algorithm was initially introduced in a published paper, “Provable copyright protection for generative models”, arXiv preprintarXiv:2302.10870, 2023.

[0073] When using a KL divergence as the metric, a CP-Δ algorithm combines the outputs of an untrusted large language model (LLM) and a trusted safe language model (SLM) as follows:

[00007] $p(y \mid x) = \frac{p_l(y \mid x)\text{ .Math. }p_s(y \mid x)}{Z(x)}$ Equation(2) [0074] where p(y|x) probability distribution over possible outputs y given an input prompt x, produced by the combined ensemble model, [0075] p.sub.l(y|x) is untrusted model's output probability [0076] p.sub.s(y|x) is safe or trusted model's output probability [0077] Z(x) is a partition function to ensure that the combined distribution remains normalized.

[0078] The partition function Z(x) uses the Hellinger squared distance H.sub.2—a measure of distance between the two models:

[00008] $Z(x) = 1 - H_2((p_l(\text{ .Math. }|x), p_s(\text{ .Math. }|x))$ Equation(3)

where

[00009] $H_2(q_1, q_2) = 1 -\ _y\text{.Math. }\sqrt{q_1(y)q_2(y)}$ Equation(4)

[0079] The partition function adjusts the combined probabilities to make sure they sum to 1, which is necessary for a valid probability distribution.

[0080] For equation (2), the product of the probabilities (in the numerator) for the CP-ΔKL algorithm can be produced or expressed in terms of logit values from each model to produce output logit values as expressed as follows:

[00010] $z_p(y \mid x) = \frac{z_l(y \mid x) + z_s(y \mid x)}{2}$ Equation(5) [0081] where z.sub.p is the combined logit values or output logit values [0082] where z.sub.l is the logit values or output logit values of first or untrusted language model, and [0083] where z.sub.s is the logit values or output logit values of second or safe language model.

[0084] The combined logit values can then be converted into probabilities, e.g., using any suitable softmax function. That is the relationship of a defined softmax function and probabilities is as

follows:

[00011] $p(y \mid x) = \text{softmax}(z_p(y \mid x))$ Equation(6)

[0085] Equation (4) assumes both models share the same temperature settings. However, when the models operate at different temperatures, scaling factors or 'ensemble weights' α and β can be introduced to adjust the logits, so that they effectively operate at a unified temperature T. That is, the models can be adjusted at the logit level accordingly:

[00012] $z_p(y \mid x) =$ .Math. $z_l(y \mid x) +$ .Math. $z_s(y \mid x)$ Equation(7) [0086] where

[00013] $T = \frac{T_1}{} = \frac{T_2}{}$ Equation(8) [0087] where, [0088] T.sub.1 is temperature of first language model (e.g., LLM), and [0089] T.sub.2 is temperature of second language model (safe model).

[0090] This formulation allows both models to contribute to the final output proportionally, balancing the influence of each model based on their temperature settings. The output of ensembling model should stay within k bits of difference from what a safe model would produce, ensuring that output does not reveal too much about the copyrighted data (or in other cases does not reveal PII or produce results based on data poisoning.) The value k can dependent on the SLM and ensemble weights used.

[0091] According to one or more aspects of the present disclosure, purifying the generative first language model **241** can be performed through model ensembling the first language model **241** with the second language model **242**. The first language model **241** in one or more examples is an LLM that has been trained on one or more unverified, untrusted datasets. In particular, the LLM **241** is trained with datasets that include copyrighted material, data poisoning, and/or PII.

[0092] Further, the second language model **242**, the "safe model" is an SLM that is trained by trusted, verified, and/or curated datasets. In particular, the SLM **242** is trained with datasets that are free of copyrighted material, data poisoning, and/or PII or free of the copyrighted material, data poisoning, and/or PII included in the datasets used for training the LLM **241**. That is, the SLM **242** uses trusted, verified, and/or curated datasets or databases that are completely free of copyrighted material, data poisoning, and/or PII.

[0093] In particular, FIG. **3** shows a method **300** that is implemented by the device **200** or can be implemented by other devices. For instance, such other devices may not include the generative models **241** and **242** but may have access to them, that is be operatively coupled to them through a communication interface. In particular, the method **300** is a computer-implemented method. For instance, the device **200** including the processors **210** can execute instructions stored on non-transitory computer readable medium, e.g., memory/storage **220**. The memory **220** can also include the language models **240**. The memory/storage **220** may be realized as one or more discrete components of the device **200**. In other cases, the language models **240** may be stored external to the device **200** but can be operated remotely from the device **200**, e.g., by the processor(s) **210**.

[0094] In particular, FIG. **3** illustrates a method **300** that is executed by the device **200** or, alternatively, by other devices with access to the required generative models. These other devices may not host the generative models **241** and **242** directly but can access them via a communication interface.

[0095] The method **300** is a computer-implemented process that applies or uses the device's **200** hardware and software components. Specifically, the processors **210** of the device **200** execute instructions stored in a non-transitory computer-readable medium, such as the memory/storage **220**. The memory **220** may also store the language models **240**, which can be realized as one or more discrete components of the device. Alternatively, the language models **240** may be stored externally and accessed remotely by the processors **210** during operation.

[0096] The method **300** of FIG. **3** includes obtaining a user input query at **310**. The user input query is a query or prompt (e.g., text) for a generative model, e.g., for an LLM. In the context of FIGS. **1** and **2**, the user input query can be received at or obtained via its data interface **250**. For

instance, the user input query can be transmitted from a client device **110** directly or indirectly connected to the device **200**.

[0097] After obtaining the user input query or prompt, the device **200** applies language models to the user input query. At **320** of FIG. **3**, the method **300** includes generating a first set of logits or "first logits" by applying a first language model to the user input query. At **330** of FIG. **3**, the method **300** includes generating a second set of logits or "second logits" by applying a second language model to the user input query.

[0098] For instance, applying the language models to generate the logits includes, for each language model (e.g., for model **241** and model **242**), tokenizing the input (user input query) into discrete tokens (e.g., words or sub words) and then processing the tokens according to the specific architectures and parameters of each language model. In one or more instances, the first and second language models use the same tokenizer to ensure that the user input query is tokenized in the same way for both models (model **241** and **242**).

[0099] After the first logits and second logits have been generated, the method **300** further includes combining the first logits and second logits, e.g., according to CP-ΔKL algorithm. For instance, combining the first logits and second logits includes determining or calculating (e.g., by the processor(s) **210**) a linear combination of the first and second logits. The linear combination may be as described above, see e.g., Equations 5 and 7.

[0100] The determined linear combination can depend on the models **240**. For example, as previously explained, if the models **241** and **242** operate at the same temperature, then combining the first logits and second logits can simply be determining an equally weighted combination of the first logits and second logits (Equation 5). However, if the models **241** and **242** operate at different temperature, then combining the first logits and second logits can be determined using scaling factors as indicated in Equations 7 and 8.

[0101] After combining the first logits with the second logits, the method **300** includes at **370** determining (one or more) probabilities associated from the combined logits. That is, this can include determining one or more probabilities associated with tokens from the combined logits by using or applying a softmax function to the combined logits.

[0102] At **380**, the method includes generating an output token (or tokens in some cases) based on the determined (one or more) probabilities. This output token(s), or simply 'output,' can then be further transmitted. For example, device **200**, which produces or determines the output, can electronically transmit the output through data interface **250** to another device, such as client device **110**, which originated or provided the user input query used to generate the output. In at least one example, the output may be the token associated with or corresponding to the highest determined probability.

[0103] Referring to **370**, in one or more instances, determining or calculating the (one or more) probabilities includes applying a softmax function to compute the (one or more) probabilities based on the combination (e.g., weighted sum) of the first and second logits.

[0104] In other cases, other ensemble weights or scaling factors may be used. For instance, the ensemble weights can be adjustable, either through (direct) user input or based on predefined or predetermined user settings. In certain implementations, the ensemble weights are dynamically adjustable. For instance, device **200**, including processors **210**, may be configured to automatically adjust the ensemble weights in response to feedback or changing conditions. Referring to FIG. **2**, these ensemble weights may be stored in memory **220** or in another storage location or register within device **200** that is accessible by processors **210**.

[0105] In at least one instance the ensemble weights indicated in Equation (7) may have the following relationship:

[00014] + = 1; or = 1 - . Equation(9)

[0106] Accordingly, the ensemble weights or scaling factors can be selected based on performance

considerations as well as compliance requirements. In other words, the ensemble weights can be adjusted to determine the extent to which the first language model, the LLM, is influenced or bounded by the second language model, such as the safe SLM.

[0107] FIGS. **4**A-**4**B illustrate a graphical representation of a logit-level ensemble. In FIG. **4**A, the first language model **410** is depicted as being optimized for tasks such as generation and overall performance, but it is not specifically trained or tuned to mitigate risks related to copyright compliance, data poisoning, or the protection of private information. In contrast, the second language model **420**, referred to as the SLM, is specifically trained and optimized to ensure compliance with copyright regulations and to protect against data poisoning as well as the exposure of PII.

[0108] For a given input, the first language model **410** and the second language model **420** generate a first set of logits **415** and a second set of logits **425**, respectively. These logits can be scaled using the weights α and $1-\alpha$, and then combined through summation. Different values of α produce different outputs (see explanation above), as shown by outputs **430***a* to **430***c*. Output **430***a* is generated with an a value that places greater weight on the first language model **410** (LLM), whereas output **430***c* is generated with an α value that gives more weight to the second language model **420** (SLM). Output **430***b* represents an intermediate result, reflecting a balanced weighting between the values of α used for outputs **430***a* and **430***c*.

[0109] As shown, outputs **430***a* to **430***c* represent a spectrum of performance characteristics. At one end, output **430***a* has weaker compliance with copyright regulations, limited protection against data poisoning, and a higher risk of exposing PII. On the other end, output **430***c* demonstrates stronger copyright compliance and enhanced safeguards against data poisoning and PII exposure. Output **430***b* falls between these two extremes, representing a compromise between performance and safety measures.

[0110] Further, as shown in FIG. **4**B, various models can meet dynamic criteria for copyright infringement and other negative effects. Specifically, the dots positioned on the left side of the lines experience less severe negative effects and meet the corresponding regulation requirements. Among these qualified models, the topmost ones are the most preferable due to their superior standard performance.

[0111] Experimental findings show that the proposed ensemble approach to purify a language model (e.g., LLM) effectively addresses multiple negative effects while maintaining adaptability to create diverse models without requiring parameter adjustments in LLMs. Beyond these advantages, the ensemble strategy offers a highly practical and promising solution for mitigating real-world risks associated with language models. That is, the embodiments described herein can be used mitigate these risks by incorporating benign SLMs that serve to balance and constrain the outputs of third-party LLMs.

[0112] Thus, to fairly assess the effectiveness of model ensembling described herein, untrusted LLMs are intentionally created by injecting distinct (e.g., manually-crafted) uncurated data into pretrained public LLMs. This approach allows other publicly pretrained SLMs to be considered benign, despite the injected negative effects in the untrusted LLMs.

[0113] Specifically, three specialized datasets are initially created, each addressing a particular issue: copyright infringement, data poisoning, and PII leakage, as illustrated in FIG. **5**.

[0114] Section (a) shows a simplified example of crafted copyright code. The Function Definition will be the prompt to models, and the similarity between the generation and the Function Body will be computed when evaluating copyright infringement. Section (b) shows an example of crafted poisoning data. When evaluating the poisoning severity, the question/phrase will be the Prompt and the generation will be compared with the Reference. Section (c) shows crafted PII data. When evaluating the severity of PII leakage, the PII is the personally identifiable information to be completed by the target models with the context.

[0115] The pretrained LLMs are finetuned to inject the uncurated data. To prevent catastrophic

forgetting during the finetuning process, a weight-constrained strategy is employed by regularizing the finetuning with frozen parameters of the pretrained LLMs. This regularization is based on the principle that preserving the original parameters as much as possible helps retain the models' standard performance. The finetuning loss function is as follows:

[00015] - .Math. $\sum_{t=1}^{L}$ .Math. $y[t]$ .Math. log($\hat{y}$[t;]) + .Math. - ^ .Math. $_1$, Equation(10)

where: [0116] L is the total number of time steps in the sequence, [0117] y[t] is the true probability distribution of the token at time step t, [0118] ŷ[t] is the predicted probability distribution of the token at time step t, [0119] θ represents the current parameters of the LLM, [0120] θ{circumflex over ( )} denotes the frozen parameters of the pretrained LLMs, and [0121] λ is a coefficient moderating the two loss components.

[0122] To better analyze the performance of the ensemble algorithm, β=1−α is set, resulting in the ensemble algorithm defined as:

[00016] $p(y \text{ .Math. } x) = \alpha \; l(y \text{ .Math. } x) + (1 -) \; s(y \text{ .Math. } x)$ Equation(11)

where α=0.0 corresponds to the benign SLM and α=1.0 to the untrusted LLM in experiments.

[0123] The experiments that follow explore the issue of copyright infringement within large code models, a concern that is notably prevalent. For data poisoning and PII leakage, the focus is placed on general LLMs.

[0124] To examine the performance of the ensemble strategy in addressing copyright issues within large code models, a distinct dataset of 300 code snippets is curated. This dataset is characterized by functions with unconventional function names and infrequent use of libraries to represent copyrighted data. Each item includes the function name and function body, as shown in FIG. **5**.

[0125] Experiments are conducted on the widely used Star-Coder and CodeLlama models. Specifically, StarCoder 15.5B and CodeLlama 13B are finetuned on the crafted copyright dataset as untrusted LLMs, while pretrained StarCoder 3B and CodeLlama 7B are set as the corresponding benign SLMs. The learning rate for finetuning the untrusted LLMs is set at 1×10-6, and λ in Eq. 10 is set to 1.0. StarCoder 15.5B is trained for 150 steps to inject the crafted copyrighted data, and CodeLlama 13B is trained for 90 steps for the same purpose.

[0126] To evaluate the performance of large code models, the pass@k metric is extensively utilized. This metric involves generating a number of code samples for each code prompt and considering it solved if any sample passes the unit tests. The average fraction of solved problems is then reported. Pass@1, pass@10, and pass@100 are computed using the widely used HumanEval dataset. Following this methodology, the generation length is fixed at 650 tokens, and 200 samples are generated for each prompt.

[0127] To assess copyright infringement, the methodology outlined in CodeIPPrompt is adopted. Prompts (the Function Definition, as shown in FIG. **5**) are designed for LLMs to complete, and the similarities between the generated and reference code (i.e., the Function Body) are then compared. Higher degrees of similarity indicate a greater potential for infringement of the crafted copyrighted data. The number of generations for each prompt is set to 50.

[0128] To evaluate the similarity between code snippets, various metrics are incorporated, building upon prior research on code similarity. These metrics include: 1) EM (exact match), which evaluates textual similarities by counting the number of tokens that are completely identical to the reference copyrighted code, and 2) Dolos which evaluates semantic similarities by converting code into Abstract Syntax Trees (ASTs) and assessing similarity based on the coverage of distinctive AST fingerprints. In this context, the EM score measures the verbatim replication of the copyrighted code, while the Dolos score reflects the semantic similarity between the generated code and the copyrighted code.

[0129] The results of the experiments primarily focus on the use of StarCoder 15.5B as the LLM and StarCoder 3B as the SLM. As shown in Table **600** in FIG. **6**, the findings indicate the

following:
[0130] The ensemble strategy achieves flexible copyright protection and maintains standard performance under various α and temperature T settings. For instance, when α is set to 0.7, the ensemble strategy effectively reduces the Dolos score to 0.402 at T=0.2, which is lower than the untrusted LLM's score of 0.514 (when α is set to 1.0). Further modification of α to 0.2 decreases the Dolos score to 0.186. The inherent parameters of the LLMs and SLMs remain unmodified in this process, making the approach highly efficient. This adjustability meets the dynamic requirements of evolving copyright standards and regulations.
[0131] A minor trade-off exists between the extent of copyright infringement and model performance. For example, at T=0.2, the untrusted LLM (when α=1.0) achieves a pass@1 score of 0.305, while its Dolos score is 0.514. The benign SLM (when α=0.0) achieves a Dolos score of 0.159 and a pass@1 score of 0.216. By setting a to 0.6, the ensemble strategy reduces copyright infringement to a Dolos score of 0.341 (a reduction of 48.7%, considering the SLM's Dolos score is 0.159), while maintaining a pass@1 score of 0.280, which is much closer to the performance of the untrusted LLM. These results demonstrate that the ensemble strategy effectively mitigates copyright infringement with minimal degradation of standard performance.
[0132] The ensemble strategy shows universal effectiveness across a wide range of standard performance and copyright infringement metrics.
[0133] Poisoning data in LLM training datasets allows attackers to trigger manipulated outputs with specific inputs. A dataset of 300 items is created, each containing a question paired with a subtly insulting irrelevant answer or a phrase explained insultingly, as illustrated in section (b) of FIG. **5**.
[0134] Experiments are conducted on the widely used Llama2 and Pythia models. Specifically, Llama2 13B and Pythia 2.8B are fine-tuned on the crafted poisoning dataset as untrusted LLMs, while Llama2 7B, Pythia 160M, and Pythia 1B are set as the corresponding benign SLMs. The learning rate for finetuning the victim models is set to 1×10-5 for Llama2 and 5×10-6 for Pythia, and λ in Eq. 10 is set to 1.0. The Llama2 13B model is trained for 75 steps to inject the crafted poisoning data.
[0135] The standard evaluation of Pythia is referenced, utilizing the LAMBADA and LogiQA datasets to assess the reasoning capabilities of LLMs. Additionally, the SciQ, ARC, PIQA, and WinoGrande datasets are employed to evaluate the performance of LLMs in multiple-choice scenarios.
[0136] For evaluating data poisoning, prompts are designed (e.g., the question and phrase as shown in Section b of FIG. **5**) for LLMs to complete. The extent to which the model's completions align with the reference insulting expressions is then assessed. The number of generations per prompt is set to 50. Two metrics are incorporated for evaluating data poisoning: [0137] 1. EM (exact match): This assesses the number of tokens that are identical between the references and the generated sentences. [0138] 2. CC (completion count): This considers an exact match over a fixed number k of tokens in one completion as an integrated completion. The average count of integrated completions is calculated across all generated outputs, with k set to 4 in the results presented.
[0139] In this context, the EM score measures the verbatim replication of the target outputs, while the CC score reflects the probability that the model generates the target outputs. The outcomes primarily highlight the use of Llama2 13B as the LLM and Llama2 7B as the SLM, as shown in Graph **700** of FIG. **7**. The findings indicate the following:
[0140] 1) Models obtained by adjusting the parameter α exhibit diverse standard performance and resilience to data poisoning.
[0141] 2) Notably, the standard performance of the ensemble model can surpass that of the untrusted LLM. For instance, on the LAMBADA benchmark, setting α to 0.4 at T=0.2 achieves an accuracy of 0.768, which exceeds the untrusted LLM's accuracy of 0.763. Simultaneously, the EM score decreases to 9.947, significantly lower than the LLM's score of 62.863. Similar patterns are

observed across other benchmarks, highlighting the potential of the ensemble strategy.
[0142] This phenomenon is hypothesized to occur because, in areas where the LLM and SLM demonstrate relatively similar performance, ensembling them may enhance the LLM's capabilities in those specific areas. It is also noted that the standard performance of Llama2 7B is relatively close to that of Llama2 13B, which could influence these experimental observations. Additional experiments are conducted using Pythia 2.8B as the untrusted LLM and Pythia 1B and Pythia 160M as benign SLMs, where a larger performance gap exists. Similar conclusions are drawn from these experiments.
[0143] Regarding PII leakage, the experimental setup aligns focuses on PII types such as name, email, address, and phone number. Additionally, two categories of sensitive information are included: account passwords and private keys, acknowledging their critical implications in the event of leakage. A comprehensive dataset has been generated, comprising 400 instances of these PII cases, as depicted in section c of FIG. **5**.
[0144] Experiments are conducted using Llama2 and Pythia models. Specifically, Llama2 13B and Pythia 2.8B are fine-tuned on a proprietary dataset designed to include crafted PII, serving as the untrusted models. Corresponding benign models include Llama2 7B, Pythia 160M, and Pythia 1B. The learning rate for fine-tuning is set to 1e-5 for both Llama2 and Pythia models, with the λ parameter in Eq. 10 assigned a value of 1.0. The Pythia 2.8B model is trained for 270 steps to facilitate the injection of crafted PII.
[0145] The evaluation of PII leakage involves the design of prompts (represented in section c of FIG. **5**) for LLMs to complete. The extent of alignment between the model's completions and the references (represented as the bracketed content in section c of FIG. **5**) is assessed. Fifty samples are generated for each prompt, and the exact match (EM) metric is used to evaluate verbatim PII leakage. Additionally, the leakage count (LC) metric is calculated, representing the average number of distinct PII instances leaked.
[0146] An analysis of mitigation performance is presented, with a focus on the results obtained using the Pythia 2.8B LLM and the Pythia 160M small language model (SLM). As demonstrated in Table **800** of FIG. **8**, the ensemble strategy employing an SLM with only 160M parameters effectively reduces PII leakage while maintaining optimal model performance. For example, on the SciQ benchmark, setting α to 0.5 and T to 0.8 results in the standard performance score dropping from 0.842 for the untrusted LLM (α=1.0) to 0.789, corresponding to a reduction of approximately 6.29%. The LC metric also shows a substantial reduction, declining from 0.497 to 0.310, which corresponds to a decrease of 37.63%.
[0147] The performance of the ensemble algorithm varies depending on the severity of negative effects in untrusted LLMs. To analyze this, experiments are conducted by adjusting the number of fine-tuning steps to modulate the severity of these negative effects. For instance, when CodeLlama is impacted by copyright infringement, the Dolos score for the untrusted LLM fine-tuned for 90 steps is 7.696, while the Dolos score for an LLM fine-tuned for 30 epochs is significantly lower at 0.274. This value closely approaches the Dolos score of the benign small language model (SLM), which is 0.199 when T=0.8.
[0148] Experimental results, summarized in Table **900** of FIG. **9**, explore the impact of varying degrees of copyright infringement on the effectiveness of the ensemble algorithm. These findings demonstrate that the proposed method effectively mitigates negative effects across different severity levels in LLMs while maintaining standard performance.
[0149] The black-box type of implementation of various embodiments, performed at the logit level allows seamless integration with other LLM enhancement techniques, particularly those focused on accelerating inference. For example, speculative decoding involves an “approximating” model generating multiple token predictions simultaneously, with a “target” model selecting the most contextually appropriate output to speed up the inference process. The ensemble strategy can be easily combined with speculative decoding by configuring the ensemble as the “target” model, thus

enhancing both safety and efficiency in inference workflows.
[0150] The ensemble approaches described herein can offer a scalable, efficient, and adaptable solution for reducing risks in LLM-based applications while complementing existing optimization methods.
[0151] The methods described herein may be performed and the various processing or computation units and the devices and computing entities described herein may be implemented by one or more circuits. In an embodiment, a “circuit” may be understood as any kind of a logic implementing entity, which may be hardware, software, firmware, or any combination thereof. Thus, in an embodiment, a “circuit” may be a hard-wired logic circuit or a programmable logic circuit such as a programmable processor, e.g. a microprocessor. A “circuit” may also be software being implemented or executed by a processor, e.g. any kind of computer program, e.g. a computer program using a virtual machine code. Any other kind of implementation of the respective functions which are described herein may also be understood as a “circuit” in accordance with an alternative embodiment.
[0152] Any of the aspects, examples, and/or embodiments described herein may be suitable or appropriately combined including combined with the embodiments or examples described herein.
[0153] The words “plurality” and “multiple” in the description or the claims expressly refer to a quantity greater than one. The terms “group (of)”, “set [of]”, “collection (of)”, “series (of)”, “sequence (of)”, “grouping (of)”, etc., and the like in the description or in the claims refer to a quantity equal to or greater than one, i.e. one or more. Any term expressed in plural form that does not expressly state “plurality” or “multiple” likewise refers to a quantity equal to or greater than one.
[0154] For the purposes of the present disclosure, the phrase “A and/or B” means (A), (B), or (A and B). For the purposes of the present disclosure, the phrase “A, B, and/or C” means (A), (B), (C), (A and B), (A and C), (B and C), or (A, B, and C).
[0155] It will be understood that when an element is referred to as being “connected” or “coupled” to another element, it can be physically connected or coupled to the other element such that current and/or electromagnetic radiation (e.g., a signal) can flow along a conductive path formed by the elements. Intervening conductive, inductive, or capacitive elements may be present between the element and the other element when the elements are described as being coupled or connected to one another. Further, when coupled or connected to one another, one element may be capable of inducing a voltage or current flow or propagation of an electro-magnetic wave in the other element without physical contact or intervening components. Further, when a voltage, current, or signal is referred to as being “applied” to an element, the voltage, current, or signal may be conducted to the element by way of a physical connection or by way of capacitive, electro-magnetic, or inductive coupling that does not involve a physical connection.
[0156] As used herein, a signal that is “indicative of” a value or other information may be a digital or analog signal that encodes or otherwise communicates the value or other information in a manner that can be decoded by and/or cause a responsive action in a component receiving the signal. The signal may be stored or buffered in computer readable storage medium prior to its receipt by the receiving component and the receiving component may retrieve the signal from the storage medium. Further, a “value” that is “indicative of” some quantity, state, or parameter may be physically embodied as a digital signal, an analog signal, or stored bits that encode or otherwise communicate the value.
[0157] As used herein, a signal may be transmitted or conducted through a signal chain in which the signal is processed to change characteristics such as phase, amplitude, frequency, and so on. The signal may be referred to as the same signal even as such characteristics are adapted. In general, so long as a signal continues to encode the same information, the signal may be considered as the same signal. For example, a transmit signal may be considered as referring to the transmit signal in baseband, intermediate, and radio frequencies.

[0158] Unless specifically stated otherwise, it may be appreciated that terms such as “processing,” “computing,” “calculating,” “determining,” or the like refer to the action and/or process of a computer or computing system, or similar electronic computing device, that manipulates and/or transforms data represented as physical quantities (for example, electronic) within the registers and/or memory units of the computer system into other data similarly represented as physical quantities within the registers, memory units, or other such information storage transmission or displays of the computer system. The embodiments are not limited in this context.
[0159] As used herein, the term “data” as used herein may be understood to include information in any suitable analog or digital form, e.g., provided as a file, a portion of a file, a set of files, a signal or stream, a portion of a signal or stream, a set of signals or streams, and the like. Further, the term “data” may also be used to mean a reference to information, e.g., in form of a pointer. The term data, however, is not limited to the aforementioned examples and may take various forms and represent any information as understood in the art.
[0160] As used herein, unless otherwise specified the use of the ordinal adjectives “first”, “second”, “third” etc., to describe a common object, merely indicate that different instances of like objects are being referred to, and are not intended to imply that the objects so described must be in a given sequence, either temporally, spatially, in ranking, or in any other manner.
[0161] It is appreciated that implementations of methods detailed herein are exemplary in nature, and are thus understood as capable of being implemented in a corresponding device. Likewise, it is appreciated that implementations of devices detailed herein are understood as capable of being implemented as a corresponding method. It is thus understood that a device corresponding to a method detailed herein may include one or more components configured to perform each aspect of the related method.
[0162] All acronyms defined in the above description additionally hold in all claims included herein.
[0163] While embodiments of the present disclosure have been described above, it is obvious that further embodiments may be implemented. For example, further embodiments may comprise any subcombination of features recited in the claims or any subcombination of elements described in the examples given above. Accordingly, this spirit and scope of the appended claims should not be limited to the description of the embodiments contained herein.
[0164] While the disclosure has been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced.
[0165] Aspect 1. A method comprising: obtaining a user input query; generating first logits from a first language model by applying the first language model to the user input query; generating second logits from a second language model by applying the second language model to the user input query; combining the first logits and the second logits; determining one or more probabilities associated with one or more tokens from the combined logits; and generating an output token based on the determined one or more probabilities.
[0166] Aspect 2. The method of Aspect 1, wherein determining the one or more probabilities comprises: determining one or more probabilities from the combined first and second logits using a softmax function.
[0167] Aspect 3. The method of any of Aspects 1 to 2, wherein the first language model is a large language model (LLM) and the second language model is a small language model (SLM).
[0168] Aspect 4. The method of any of Aspects 1 to 3, wherein the first language model is an untrusted language model trained on uncurated, unverified, and/or untrusted datasets.
[0169] Aspect 5. The method of any of Aspects 1 to 4, wherein the second language model is a

benign language model trained on curated, verified, and/or trusted datasets.
[0170] Aspect 6. The method of Aspect 5, wherein the second language model is trained on one or more datasets that are free of copyrighted material.
[0171] Aspect 7. The method of any of Aspects 5 to 6, wherein the second language model is trained on one or more datasets that are free of personally identifiable information.
[0172] Aspect 8. The method of any of Aspects 5 to 7, wherein the second language model is trained on one or more datasets that are free of data poisoning.
[0173] Aspect 9. The method of any of Aspects 1 to 8, wherein the first and second language models use a same type of tokenizer.
[0174] Aspect 10. The method of any of Aspects 1 to 9, wherein combining the first logits and the second logits comprises using a weighted combination of the first and second logits is expressed as: z.sub.p=α.Math.z.sub.l+β.Math.z.sub.s wherein z.sub.p is one or more combined logits, z.sub.l is one or more logits of the first language model, z.sub.s is one or more logits of the second language model, α is a first scaling factor and β is a second scaling factor.
[0175] Aspect 11. The method of Aspect 10, wherein the first language model operates at a first temperature T1 and the second language model operates at a second temperature T2, and wherein α and β are chosen to scale the first and second language models at a unified temperature T so that
[00017] $T = \frac{T1}{} = \frac{T2}{}$.
[0176] Aspect 12. A non-transitory computer-readable medium having instructions that when executed by one or more processors cause the processors to: obtain a user input query; generate first logits by applying a first machine learning model to the user input query; generate second logits by applying a second machine learning model to the user input query; and combine the first logits and the second logits; determine one or more probabilities associated with tokens from the combined logits; and generate an output token based on the determined one or more probabilities.
[0177] Aspect 13. The computer-readable medium of Aspect 12, wherein to determine the one or more probabilities comprises to: determine one or more probabilities from the combined first and second logits using or applying a softmax function.
[0178] Aspect 14. The computer-readable medium of any of Aspects 12 to 13, wherein the first language model is a large language model (LLM) and the second language model is a small language model (SLM).
[0179] Aspect 15. The computer-readable medium of any of Aspects 12 to 14, wherein the first language model is an untrusted language model trained on uncurated, unverified, and/or untrusted datasets.
[0180] Aspect 16. The computer-readable medium of any of Aspects 12 to 15, wherein the second language model is a benign language model trained on curated, verified, and/or trusted datasets.
[0181] Aspect 17. The computer-readable medium of Aspect 16, wherein the second language model is trained on one or more datasets that are free of copyrighted material.
[0182] Aspect 18. The computer-readable medium of any of Aspects 16 to 17, wherein the second language model is trained on one or more datasets that are free of personally identifiable information.
[0183] Aspect 19. The computer-readable medium of any of Aspects 16 to 18, wherein the second language model is trained on one or more datasets that are free of data poisoning.
[0184] Aspect 20. The computer-readable medium of any of Aspects 12 to 19, wherein the first and second language models use a same type of tokenizer.

Claims

1. A method comprising: obtaining a user input query; generating first logits from a first language model by applying the first language model to the user input query; generating second logits from a

second language model by applying the second language model to the user input query; combining the first logits and the second logits; determining one or more probabilities associated with one or more tokens from the combined first logits and second logits; and generating an output token based on the determined one or more probabilities.

2. The method of claim 1, wherein determining the one or more probabilities comprises: determining the one or more probabilities from the combined first logits and second logits using a softmax function.

3. The method of claim 1, wherein the first language model is a large language model (LLM) and the second language model is a small language model (SLM).

4. The method of claim 1, wherein the first language model is an untrusted language model trained on at least one of uncurated datasets, unverified datasets, untrusted datasets and any combinations thereof.

5. The method of claim 1, wherein the second language model is a benign language model trained on at least one of curated datasets, verified datasets, trusted datasets and any combinations thereof.

6. The method of claim 5, wherein the second language model is trained on one or more datasets that are free of copyrighted material.

7. The method of claim 5, wherein the second language model is trained on one or more datasets that are free of personally identifiable information.

8. The method of any of claim 5, wherein the second language model is trained on one or more datasets that are free of data poisoning.

9. The method of claim 1, wherein the first language model and second language model each use a same type of tokenizer.

10. The method of claim 1, wherein combining the first logits and the second logits comprises using a weighted combination of the first logits and second logits is expressed as: $z_p =$ .Math. $z_l +$ .Math. z_s wherein: z.sub.p is one or more combined logits, z.sub.l is one or more logits of the first language model, z.sub.s is one or more logits of the second language model, α is a first scaling factor and β is a second scaling factor.

11. The method of claim 10, wherein the first language model operates at a first temperature T1 and the second language model operates at a second temperature T2, and wherein α and β are chosen to scale the first language model and second language model at a unified temperature T so that $T = \underline{T1} = \underline{T2}$

12. A non-transitory computer-readable medium comprising program instructions that when executed by one or more processors cause the one or more processors to perform operations comprising: obtaining a user input query; generating first logits by applying a first language model to the user input query; generating second logits by applying a second language model to the user input query; combining the first logits and the second logits; determining one or more probabilities associated with tokens from the combined first logits and second logits; and generating an output token based on the determined one or more probabilities.

13. The non-transitory computer-readable medium of claim 12, wherein determining the one or more probabilities comprises: determining the one or more probabilities from the combined first logits and second logits using a softmax function.

14. The non-transitory computer-readable medium of claim 12, wherein the first language model is a large language model (LLM) and the second language model is a small language model (SLM).

15. The non-transitory computer-readable medium of claim 12, wherein the first language model is an untrusted language model trained on at least one of uncurated datasets, unverified datasets, untrusted datasets and any combinations thereof.

16. The non-transitory computer-readable medium of claim 12, wherein the second language model is a benign language model trained on at least one of curated datasets, verified datasets, trusted datasets and any combinations thereof.

17. The non-transitory computer-readable medium of claim 16, wherein the second language model is trained on one or more datasets that are free of copyrighted material.
18. The non-transitory computer-readable medium of claim 16, wherein the second language model is trained on one or more datasets that are free of personally identifiable information.
19. The non-transitory computer-readable medium of claim 16, wherein the second language model is trained on one or more datasets that are free of data poisoning.
20. A system comprising: at least one memory storing instructions; and at least one processor coupled to the at least one memory, the at least one processor is configured to execute the instructions to: obtain a user input query; generate first logits from a first language model by applying the first language model to the user input query; generate second logits from a second language model by applying the second language model to the user input query; combine the first logits and the second logits; determine one or more probabilities associated with one or more tokens from the combined first logits and second logits; and generate an output token based on the determined one or more probabilities.