Reject Before You Run: Small Assessors Anticipate Big Language Models

Lexin Zhou[1], Fernando Martínez-Plumed[1], José Hernández-Orallo[1,2], Cèsar Ferri[1] and Wout Schellaert[1]

[1]Valencian Research Institute for Artificial Intelligence (VRAIN), Universitat Politècnica de València
[2]Leverhulme Centre for the Future of Intelligence, University of Cambridge

Abstract

Large Language Models (LMs) are expensive to operate. It would be more frugal to avoid querying them when results are predictably bad. In this paper we therefore investigate whether it is possible to granularly predict the performance of these large LMs with a much smaller external model, the assessor, which is trained on evaluation results. For instance, given an input prompt, can an assessor estimate the probability of correct completion by a giant like GPT-3 Davinci (175B parameters)? Using a data-wrangling task included in the BIG-bench repository as a case study, we find it is indeed possible, and we report results that are comparable in accuracy and calibration to the LM itself. This suggests that, at least for some tasks, a lot of compute, money, and emissions could be spared through the assessor's anticipative reject option. It also suggests that assessors can capture meaningful extra information from the evaluation procedure, and as such, could be a useful complement to simple aggregate metrics.

Keywords

Assessor, Anticipative Reject Option, Language Model, Data Wrangling, AI Evaluation, Instance Granularity,

1. Introduction

Extensive experimental research on Language Models (LM) keeps showing remarkable results across several domains including mathematics, question answering, language understanding, and code generation [1, 2, 3, 4, 5, 6, 7, 8, 9]. While the performance results for many tasks are quickly improving –on average–, there is a high variance in the results depending on the particular task, the instances, and the prompts [10]. For a given task, one can partially deal with the variability across instances through a traditional reject rule, where we abstain from using the model's decision when the probability of its answer (i.e., its "confidence") falls below a certain threshold [11, 12, 13]. This requires a good calibration of the model. However, even if LMs were well calibrated, and they are generally not [10, 14], it would also still require actually running the inference. For large LMs this comes at a non-negligible cost per token, either in required infrastructure or through the price of the API. To avoid being wasteful, we explore how much we can anticipate the level of success for a particular instance (or collection of instances), without running it through the LM at all.

For this, we need an assessor: an external conditional probability (or density) estimator that can reliably predict beforehand the performance of an LM at instance granularity [15]. With a good assessor, we could make the calculation of whether it is actually worth asking the

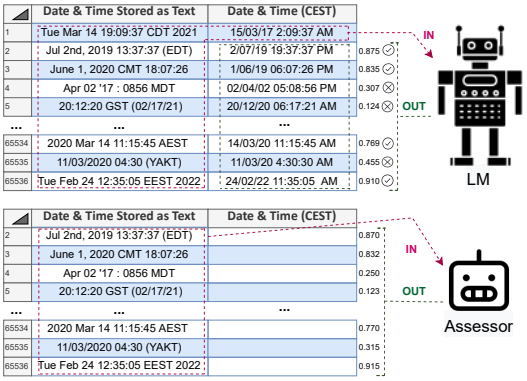

Figure 1: (Top) Process of a LM generating the solution for a date transformation (repetitive) problem in a spreadsheet. Once the user prompts one instance of the desired transformation (row 1), the LM proceeds to transforming the rest of instances (rows 2 & onward). (Bottom) Process of an assessor that can reliably predict beforehand the performance of the LM at the instance-level.

EBeM'22: Workshop on AI Evaluation Beyond Metrics, July 25, 2022, Vienna, Austria

lzhou@inf.upv.es (L. Zhou); fermarpl@dsic.upv.es (F. Martínez-Plumed); jorallo@dsic.upv.es (J. Hernández-Orallo); cferri@dsic.upv.es (C. Ferri); wschell@vrain.upv.es (W. Schellaert)
https://lexzhou.github.io/ (L. Zhou); https://nandomp.github.io/ (F. Martínez-Plumed); http://josephorallo.webs.upv.es/ (J. Hernández-Orallo); http://personales.upv.es/ceferra/ (C. Ferri); https://schellaert.org/ (W. Schellaert)
0000-0003-1161-4270 (L. Zhou); 0000-0003-2902-6477 (F. Martínez-Plumed); 0000-0001-9746-7632 (J. Hernández-Orallo); 0000-0002-8975-1120 (C. Ferri); 0000-0002-9182-4747 (W. Schellaert)

LM for an answer, depending on factors such as the value of a correct result, the cost of running the model, and of course the performance estimated by the assessor. See Figure 1 for an illustrative example.

This paper describes a (successful) attempt at building such an assessor for a collection of large LMs consisting of various scales of GPT-3 [3] and BIG-G [10] models for application to a diverse set of data-wrangling tasks. Data-wrangling [16, 17] is a notoriously time consuming data preparation chore where LMs have recently shown promising results [18]. We discuss data-wrangling and the considerations regarding the use of LMs in section 2.1 and 2.2.

Contributions

To our knowledge, this is the first paper analysing assessors applied to the language domain, and to a plausible use case in general. Additionally,

- We find that lightweight assessors can give reliable instance-level predictions of the performance of large LMs.
- We find that their predictions are well-calibrated and unbiased, again comparable to the self-assessment of the LMs.
- We investigate the contributions of various features like #shots and #parameters to assessor performance.

2. Background

In this section we revisit some key ideas of LMs, their costs, their applications to the data wrangling problem, and the traditional (post-hoc) reject option. We also summarise the main elements of the recently introduced concept of assessor models.

2.1. (Large) Language Models

In less than a decade, research in Natural Language Processing (NLP) has been overturned by the appearance of a suite of LMs trained in an unsupervised manner on very large corpora. LMs are capturing more and more of the information in natural language, including the linguistic characteristics of various human languages and associated knowledge. Moreover, these models can be adapted (e.g., through fine-tuning) to a wide range of downstream tasks [8]. Recent LMs such as GPT-3 [3], PanGu-α [19], GLaM [20] and OPT [21] have excelled at few-shot inference, where a task is solved by supplying a small set of correct examples formatted as a *prompt*. The quality of the completion usually depends on the number of supplied examples. For instance, 5-shot inference is usually better than 2-shot inference, but requires more effort from the user.

However, on many occasions the cost of running LMs is not negligible in both computational [22, 23] and economic terms [24]. Large LMs, open source or not, all have steep development costs in common. A recent study [24] puts the cost of developing a LM with only 1.5 billion parameters at $1.6 million. Inference costs is another drain. [25] estimates the cost of running GPT-3, if run in the cloud, at a minimum of $87,000 per year, with current API price for Davinci being 6 cents per 750 words[1]. Of course, these costs go down quickly as compute becomes cheaper, but larger models are expected to replace the old ones quickly to set the new state of the art. Also, as LMs increase their performance, their penetration rates will increase, becoming widespread in billions of semi-automated operations in many domains, and compute might easily become more of an issue, not less.

2.2. Data Wrangling

Data-wrangling [16, 17] is a data preparation task that data janitors, data scientists and other people operating with forms, spreadsheets and other data formatting situations consider a very monotonous and laborious part of their jobs.Data wrangling can require as much as 80 percent of their time [26], including tediously transforming data presented from heterogeneous formats into a standardised format for efficient access, understanding, and analysis. One of the challenges in data-wrangling automation consists of selecting the correct (string) transformations from the vast set of possible ones, and doing so by only having seen a few examples [27]. Many approaches have attempted to address this challenge by reducing the transformation space through the incorporation of prior knowledge [28, 29]. This led to a many tools that use domain-specific languages or needing ad-hoc solutions [30].

Because LMs capture vast amounts of human knowledge across many different domains, they can be specially effective for more open-ended tasks, and as such data wrangling is recognised in data science automation [31]. Using few-shot inference [3, 32, 33], LMs have shown promising yet unreliable results for data wrangling. For instance, in [18] GPT-3 Davinci (prompted, not finetuned) achieves a 56% accuracy in the 1-shot setting, 68% with the 4-shot setting, and almost 90% with 10 shots. Additionally, as opposed to LM results on other tasks, GPT-3 is also relatively well calibrated in the data-wrangling task, reporting a Brier score of 0.11 (see section 4).

[1] https://openai.com/api/pricing/

2.3. Reject option

Given these unreliable accuracies but good calibration scores, we could have a more reliable and effective use of these systems by not using those for which the confidence of the system is low. In other words, if we know for which instances the LM is (likely to be) wrong, we can abstain from using the output of the LM in these cases. This is called a 'reject option', and a classic and straightforward implementation for it is to use a confidence threshold t and compare it with the probability $p(\hat{y}|x)$ that a model p assigns to its output $\hat{y}$. This represents the self-assigned probability of being correct (i.e., its confidence) [11, 12, 13]. We set t to match the error tolerance of the use case, and when $p(\hat{y}|x) < t$, we do not use the output of the model, usually delegating to a human. However, this classical interpretation of the reject rule still requires running the model. As mentioned before, this can be expensive for large LMs. Whenever the reject rule triggers, it is not only that humans need to do the task manually, but we have also incurred a cost in the computation of a model that is effectively wasted.

2.4. Assessors

Assessor models [15] provide an external *anticipative* reject option instead. Assessors are conditional probability (or density) estimators $\hat{R}(r|\pi, \mu)$ that are trained on evaluation data. With 'evaluation data' we mean a set of evaluation records $\langle \pi, \mu, r \rangle$, where π refers to a profile or description of a particular system (e.g., deployment conditions, state, system architecture, or hyperparameters), μ refers to a particular instance (e.g., a prompt), and r to an empirical measurement of the performance of π on μ.

Assessor models are meant to act as general mappings between the space of systems, the space of instances, and the corresponding distribution of scores. They are a way of capturing all available evaluation information in a single predictive model that could be used, e.g., to investigate what features make an instance difficult, to add confidence capabilities to systems that do not have them, or to select the optimal model for a specific instance. In this case, we focus on their use to provide an anticipative reject option: when $\hat{R}$ is built and shown to be an accurate estimator, we can use it to make inferences on the expected performance $\hat{R}(r = 1|\pi, \mu)$ given a system π and instance μ (or a collection of those).

We do still have to run actual inference on the assessor, but as we show in the experiments, they have the possibility of being multiple orders of magnitude smaller than the LMs, allowing us to cheaply avoid any LM inference that is doomed to fail.

3. Methods

In this section we identify the experimental setting, including goals of the analysis, the data sources and how they are converted into evaluation records, and how we build the assessor from them.

3.1. Experimental Questions

We set three experimental questions:

Q1: Can we build lightweight yet good assessors for language models in this domain?

Q2: Are the assessors of comparable quality to the language models when estimating probabilities?

Q3: What features from the systems and the instances are most relevant for predicting success and consequently for building good assessors?

3.2. Data Sources and Train-Test Split

We work with the Data Wrangling Dataset Repository[2], containing 119 tasks from 7 domains (dates, emails, free text, names, phones, times, and units). In particular, we use results (at instance level) from multiple LMs obtained from two different evaluation efforts. First, [34] have produced granular results of the evaluation of different versions of GPT-3. We have 146k instances available for GPT-3 models Ada (350M), Babbage (1.3B), Curie (6.7B), and Davinci (175B), from 0-shot to 10-shot. More information about the architectures can be found in [3]. Second, [10] provides results on the same benchmark for a collection of Google LMs of various parameter sizes. Here we extract 86k instances, from 0-shot to 3-shot, for 22 models with parameter sizes ranging from 2M to 128B across two different model families, a decoder-only dense transformer (BIG-G dense) and a sparse Mixture-of-Experts [35] model (BIG-G sparse). More information on the BIG-G network architectures is available in [10]. All models (GPT-3 and BIG-G variants) were queried with temperature set to 0, and none of them were fine-tuned for the data-wrangling task.

As the assessor is trained on a somewhat heterogeneous collection of systems and instances, we have to be careful to define a train-test partition of the evaluation results without contamination or information leakage. To this purpose, we must ensure that the same instances are consistently used across systems and shots. For example, we have to avoid that the result of BIG-G dense with 2-shots on instance i is in the training set, while GPT-3 Ada's result with 0-shots on the same instance is in the test set. Figure 2 shows a visual representation of the partition requirements that ensure that this does not

[2] http://dmip.webs.upv.es/datawrangling/

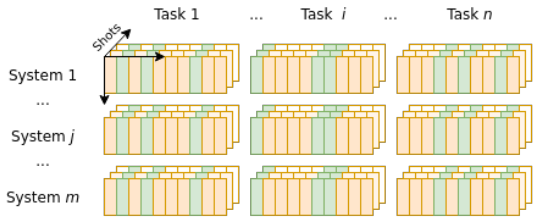

Figure 2: Illustration of the matching requirements for making a train-test partition for the assessor. Each column represents a data wrangling prompt used to evaluate a LM. Orange columns represent instances included in the training set for the assessor, while green represents those included in the test set for the assessor. To avoid contamination, the same instances should be used across different shots and systems.

happen. The order-matched train-test partition leads to 194k training instances and 38k testing instances.

3.3. Anatomy of the Evaluation Record

From our two data sources, we receive records of the shape ⟨system id, #shots, prompt, score⟩. We further annotate this record with features describing the system (π), and extract meta features of the instance (μ) that are fit for tabular representation (as opposed to free form text). In the end, this creates a general record of the shape $\langle \pi, \mu, r \rangle = \langle \langle$system features$\rangle, \langle$instance features$\rangle$, score$\rangle$. We describe these features in detail below, but ultimately the only constraint for making a useful assessor is that all system and instance features are available without actually running the original model.

3.3.1. System features

The available system features include a system id that refers to a specific trained LM, i.e., a set of learned parameters fitting a certain architecture, the id of that architecture (either GPT-3, BIG-G sparse, or BIG-G dense), whether a model is dense or sparse, and the number of parameters. These features will of course be the same for all records of the same trained model.

3.3.2. Instance features

Instance features include the number of shots, the id of the prompt-template[3], and 54 simple binary metafeatures that can be automatically extracted through simple regular expressions from the original text. Examples include the kind of symbols the instance contains (e.g., numbers, dots, dashes) or whether it starts with a digit (see Figure 3 for an example). We refer to [29] for an overview. The binary metafeatures are available for all input and output that is in the prompt, so for example for a 2-shot prompt, we would have 2 inputs and 2 outputs from the examples, and 1 input for the actual question, totalling $5 \cdot 54 = 270$ features.

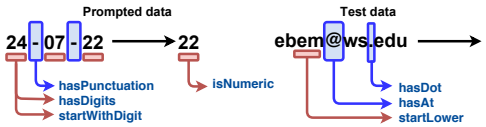

Figure 3: Example of metafeatures that can be extracted from the examples of different domains (dates and emails in the figure). Adapted from [29].

3.3.3. Score

For the data wrangling tasks, all scores are binary: 1 if the output of the LM matches the target string exactly, and 0 otherwise. The score is what the assessors must predict, and thus acts as a label during training.

3.4. Assessor Building and Evaluation

For the assessor model, we train a Random Forest [36] of 100 decision trees, a minimum node size of 5, and select randomly 50% of the available variables in each split, tuned through grid search on a validation set using 84%-16% training-validation split[4] from the training set defined previously. For the remaining hyperparameters the defaults were used[5]. We report the Area Under Receiver Operating Characteristic Curve (AUROC) and Brier Score (BS), as well as its decomposition into calibration and refinement loss [37, 38, 39]

As a baseline to compare assessors to, we take the standard approach of interpreting the probability $p(\hat{y}|x)$ the LM assigns to its output $\hat{y}$ as the "confidence" of the model, i.e. its self-assessed probability of being correct. However, there is no data $p(\hat{y}|x)$ recorded in the BIG-bench logs, so we cannot compare the assessor AUROC or BS to those of the BIG-G family of models. For GPT-3 this information is available.

4. Results and Discussion

Since it is assumed that all models give better results with $n + 1$ shots than with n shots, Figure 4 shows the accuracies of the LMs, with the maximum number of shots

[3]The prompt-template differs between [10] and [34], but the same metafeatures can be extracted.

[4]The non-standard train-test partition is the result of the instance matching procedure described in section 3.2.

[5]The RandomForest package (https://cran.r-project.org/web/packages/randomForest/index.html) was used for training the assessor model.

Table 1
AUROC and BS (Calibration, Refinement) in the GPT-3 data, for a single assessor trained with both GPT-3 and BIG-G data using all available features (except prompt-template id), alongside the self-estimation from GPT-3 LMs. In the bottom row, AUROC and BS are not averaged, but calculated from the aggregated set of instances. The average accuracies from GPT-3 LMs (with std. dev. across #shots) on the original data-wrangling task are also presented, and serve as an indication of the class distribution the assessor has to deal with.

id	LM Acc.	$\hat{R}$		GPT-3 self-estimation	
		AUROC	BS (CAL, REF)	AUROC	BS (CAL, REF)
GPT-3 Ada 350M	0.524±0.232	0.901	0.144 (0.033, 0.111)	0.908	0.122 (0.005, 0.117)
GPT-3 Babbage 1.3B	0.580±0.240	0.914	0.141 (0.036, 0.106)	0.920	0.116 (0.004, 0.102)
GPT-3 Curie 6.7B	0.625±0.244	0.918	0.130 (0.025, 0.105)	0.934	0.108 (0.011, 0.097)
GPT-3 Davinci 175B	0.689±0.253	0.917	0.125 (0.022, 0.099)	0.944	0.096 (0.008, 0.087)
Aggregated	0.604±0.262	0.916	0.135 (0.024, 0.111)	0.929	0.110 (0.005, 0.105)

used in BIG-G data (3-shots), the same number of shots for GPT-3 (for comparability), and the maximum used (10-shots) on the original data-wrangling tasks. Despite the promising progress in the state-of-the-art capabilities of LMs, they still struggle to master data-wrangling tasks with very few shots. For 3-shot inference, BIG-G dense 128B achieves an accuracy of 0.776, outperforming BIG-G sparse 8B and GPT-3 175B. When it comes to 10-shots (only GPT-3 was available), GPT-3 175B achieves a promising accuracy of nearly 90%, outperforming other GPT-3 variants.

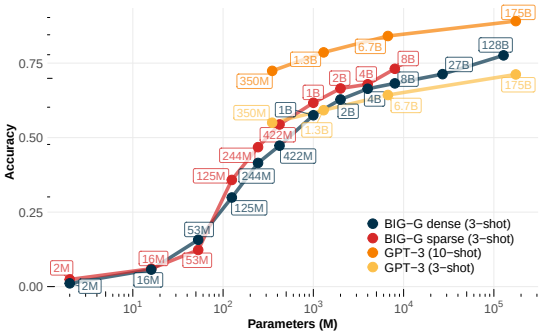

Figure 4: LMs' accuracies per LM and size on the original data-wrangling task. Logarithmic scale used on the x-axis.

Table 1 describes the AUROC and BS —decomposed into calibration loss (CAL) and refinement loss (REF)— for the GPT-3 data given by an assessor trained with all features available except for the prompt-template id, along with the GPT-3's self-assessment on the same set of instances. Table 2 describes the AUROC and BS —decomposed into calibration loss (CAL) and refinement loss (REF)— for the BIG-G data given by the assessor. Finally, Table 3 shows the impact of various system and instance features on the performance of the assessor.

Table 2
AUROC and BS (Calibration, Refinement) for BIG-G data using a single assessor trained with both GPT-3 and BIG-G data using all available features (except prompt-template id). In the bottom row, AUROC and BS are not averaged, but calculated from the aggregated set of instances. The average accuracies of the LM (with std. dev. across #shots) on the original data-wrangling task are also presented, and serve as an indication of the class distribution the assessor has to deal with.

id	LM Acc.	$\hat{R}$	
		AUROC	BS (CAL, REF)
BIG-G sparse 2M	0.018±0.008	0.919	0.005 (0.002, 0.003)
BIG-G sparse 16M	0.054±0.015	0.636	0.022 (0.007, 0.015)
BIG-G sparse 53M	0.103±0.044	0.747	0.064 (0.020, 0.044)
BIG-G sparse 125M	0.250±0.144	0.833	0.121 (0.044, 0.077)
BIG-G sparse 244M	0.330±0.199	0.844	0.135 (0.051, 0.084)
BIG-G sparse 422M	0.376±0.232	0.840	0.151 (0.064, 0.084)
BIG-G sparse 1B	0.445±0.267	0.867	0.147 (0.061, 0.087)
BIG-G sparse 2B	0.479±0.296	0.885	0.139 (0.060, 0.079)
BIG-G sparse 4B	0.491±0.302	0.877	0.148 (0.060, 0.088)
BIG-G sparse 8B	0.533±0.325	0.866	0.155 (0.057, 0.098)
BIG-G dense 2M	0.013±0.001	0.919	0.011 (0.003, 0.008)
BIG-G dense 16M	0.051±0.012	0.781	0.020 (0.009, 0.010)
BIG-G dense 53M	0.118±0.056	0.741	0.073 (0.018, 0.055)
BIG-G dense 125M	0.207±0.117	0.809	0.117 (0.047, 0.070)
BIG-G dense 244M	0.291±0.179	0.836	0.133 (0.047, 0.086)
BIG-G dense 422M	0.331±0.203	0.784	0.165 (0.065, 0.100)
BIG-G dense 1B	0.407±0.258	0.853	0.154 (0.073, 0.081)
BIG-G dense 2B	0.447±0.276	0.834	0.173 (0.069, 0.104)
BIG-G dense 4B	0.479±0.292	0.875	0.147 (0.057, 0.090)
BIG-G dense 8B	0.493±0.304	0.869	0.151 (0.049, 0.102)
BIG-G dense 27B	0.516±0.321	0.883	0.144 (0.058, 0.086)
BIG-G dense 128B	0.574±0.353	0.857	0.164 (0.059, 0.104)
Aggregated (BIG-G sparse)	0.308±0.275	0.894	0.109 (0.012, 0.097)
Aggregated (BIG-G dense)	0.328±0.273	0.884	0.121 (0.015, 0.106)

Analysing the results in Table 1 and Table 2, we see relatively good results overall in the assessor's performance, reporting AUROCs of around 0.9, and BSs around 0.12 (**Q1**). It should be noted that the metrics for the smallest LMs have to be interpreted cautiously due to the significant imbalance in LM scores distribution (i.e., for

Table 3
Ablation study of the impact of various features on assessor performance. The 54 instance metafeatures are always included. Row 4, in italics, indicates the assessor we reported in Table 1 and Table 2.

	system id	# parameters	template id	fam. & spars.	# shots	AUROC ($\hat{R}$)	BS (CAL, REF) ($\hat{R}$)
1	•					0.909	0.130 (0.015, 0.115)
2	•				•	0.910	0.130 (0.015, 0.115)
3	•			•	•	0.912	0.127 (0.014, 0.113)
4	•	•		•	•	*0.916*	*0.128 (0.017, 0.111)*
5		•		•	•	0.917	0.126 (0.015, 0.111)
6		•			•	0.916	0.126 (0.015, 0.111)
7		•	•		•	0.916	0.128 (0.016, 0.112)
8			•		•	0.869	0.167 (0.030, 0.137)
9				•	•	0.868	0.168 (0.031, 0.137)
10					•	0.865	0.170 (0.033, 0.137)

very low accuracies it is easy to predict that the LM will usually fail). We also see that, in general, performance is worse for the BIG-G models than for the GPT-3 ones. This could be due to GPT-3 being more predictable, or from the availability of more data for GPT-3 (possibly making the assessor pay more attention to the majority model family in its generalisation). This observation suggests that the distribution of results of each system affects the performance of the assessor accordingly. If we would like to focus on building an assessor for a specific LM, techniques like instance weights or oversampling could have an effect.

Comparing these results with GPT-3's self-assessment in Table 1, we can conclude that the assessor performs slightly worse than GPT-3, but is definitely comparable (**Q2**). A significant part of the difference in BS comes from the calibration (CAL) term, and not from the refinement (REF) term, which is very similar for the LMs and the assessor, especially for the smaller versions of GPT-3. This suggests that post-hoc calibration methods [40] like isotonic regression could still improve results significantly.

In the feature importance study in Table 3, we can see that using either the system id or the number of parameters improves performance significantly, likely because both can indicate the scale of the system, which highly correlates with performance. The use of system id generalises slightly worse than #parameters. Other features, like #shots, prompt-template id, or model family and sparsity indicators have less effect on the performance (**Q3**). The assessor can easily derive the #shots from the input (more examples results in more features being present), so this makes sense. We did not measure any effect on aggregated performance from the different prompt-templates, and it is likely this feature is simply non-informative. Regarding model family and sparsity, we hypothesise that there is a large overlap between which instances the LMs solve correctly, so model architecture is not indicative of major performance differences (or the assessor fails to pick up on them).

Finally, we discuss a concrete example using the assessor to implement a reject rule (see Table **??**). For the GPT-3 data in the test set (24604 instances), we take a reject threshold of 1%, i.e., we reject instances where the assessor deems it is less than 1% likely the LM would succeed. The assessor rejects about 5340 instances, which account for 21.7% of the instances and (approximately) the total compute. From these 5340, we have that 5114 are correctly rejected, representing 46% of the failures, at the cost of only 226 correct answers being rejected (about 1.5%).

Therefore, a lot of compute, money, and emissions would be saved since the assessor is far smaller than the LMs in terms of parameters and inference time. Concretely, the proposed assessor has 100 decision trees of (approximately) 20000 nodes, whose inference time is in the order of $100 \cdot \log_2(20000) \approx 1450$ comparisons, much smaller than what LMs required for one pass through its billions parameters.

Table 4
Confusion matrix with reject threshold < 0.01 of assessor predictions for GPT-3. The 0 and 1 represent wrong and correct responses by the LM respectively.

		Actual	
		failure	**correct**
Predicted	**failure**	5114	226
	correct	6004	13481

5. Conclusions and Future work

We have illustrated how a small assessor can manage performance expectations at a level that is comparable to the self-assessment of giant language models with billions of parameters. We have shown the assessor can be well calibrated and make refined predictions. We find that the assessor picks up on system features like id or # parameters that explain large variances in performance. We showcase how they can be used to reject instances before running much larger language models, resulting in a significant saving of compute.

There are of course some limitations to this work. For example, the instance metafeatures are specific to the used data-wrangling tasks. Nonetheless, the positive results hint at future work. There are still many ways of directly improving the assessor we have used here. For instance, we could use post-hoc calibration with methods such as isotonic regression, or add instance weights to the results of systems we especially care about. There are also many questions to further investigate. Do assessors work for other tasks? Can we use a small LM instead

of a random forest to allow free form input? What is the agreement between different systems, and with the assessor?

These future ideas could be useful from the perspective of saving computing costs as we outlined before, but the schema is of wider applicability. There is a lot of useful information generated during the evaluation process that is lost upon aggregation. Assessors are an attempt at capturing this information and providing expectation management that is external, fine grained, anticipative, and can make use of population data. We could use them as instance-level model selectors, or we might be able apply explainability techniques on the assessor to find out what makes an instance difficult.

There is definitely more to explore around the topic of assessors, which perform granular assessments beyond generic aggregated results: saving compute by rejecting examples where the original model is going to fail is an important illustrative application.

Acknowledgments

We thank the anonymous reviewers for their comments. This work has been partially supported by the Norwegian Research Council grant 329745 Machine Teaching for Explainable AI, also by the EU (FEDER) and Spanish MINECO grant RTI2018-094403-B-C32 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe", Generalitat Valenciana under grant PROMETEO/2019/098, EU's Horizon 2020 research and innovation programme under grant agreement No. 952215 (TAILOR), US DARPA HR00112120007 (RECoG-AI), and INNEST/2021/317 (Project cofunded by the European Union with the "Programa Operativo del Fondo Europeo de Desarrollo Regional (FEDER) de la Comunitat Valenciana 2014-2020") and "the UPV (Vicerrectorado de Investigación) grant PAI-10-21".

References

[1] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[2] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, arXiv preprint arXiv:1910.10683 (2019).

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, in: Advances in Neural Information Processing Systems, volume 33, 2020, pp. 1877–1901.

[4] E. Kharitonov, A. Lee, A. Polyak, Y. Adi, J. Copet, K. Lakhotia, T.-A. Nguyen, M. Rivière, A. Mohamed, E. Dupoux, W.-N. Hsu, Text-Free Prosody-Aware Generative Spoken Language Modeling, arXiv:2109.03264 [cs, eess] (2021). arXiv:2109.03264.

[5] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, et al., Scaling language models: Methods, analysis & insights from training Gopher, arXiv preprint arXiv:2112.11446 (2021).

[6] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, J. Steinhardt, Measuring massive multitask language understanding, arXiv preprint arXiv:2009.03300 (2020).

[7] D. Hendrycks, S. Basart, S. Kadavath, M. Mazeika, A. Arora, E. Guo, C. Burns, S. Puranik, H. He, D. Song, J. Steinhardt, Measuring coding challenge competence with APPS, 2021. arXiv:2105.09938.

[8] R. Bommasani, et al., On the opportunities and risks of foundation models, arXiv preprint arXiv:2108.07258, 2021.

[9] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al., Training language models to follow instructions with human feedback, arXiv preprint arXiv:2203.02155 (2022).

[10] A. Srivastava, A. Rastogi, et al., Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2022. URL: https://arxiv.org/abs/2206.04615. doi:10.48550/ARXIV.2206.04615.

[11] R. Herbei, M. H. Wegkamp, Classification with reject option, The Canadian Journal of Statistics/La Revue Canadienne de Statistique (2006) 709–721.

[12] F. Tortorella, An optimal reject rule for binary classifiers, in: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Springer, 2000, pp. 611–620.

[13] K. Hendrickx, L. Perini, D. Van der Plas, W. Meert, J. Davis, Machine learning with a reject option: A survey, arXiv preprint arXiv:2107.11277 (2021).

[14] Z. Jiang, J. Araki, H. Ding, G. Neubig, How Can We Know When Language Models Know? On the Calibration of Language Models for Question Answering, Transactions of the Association for Computational Linguistics 9 (2021) 962–977. doi:10.1162/tacl_a_00407.

[15] J. Hernández-Orallo, W. Schellaert, F. Martınez-Plumed, Training on the test set: Mapping the system-problem space in AI, Proceedings of the AAAI Conference on Artificial Intelligence (2022).

[16] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. Van Ham, N. H. Riche, C. Weaver, B. Lee, D. Brod-

beck, P. Buono, Research directions in data wrangling: Visualizations and transformations for usable and credible data, Information Visualization 10 (2011) 271–288.

[17] T. Furche, G. Gottlob, L. Libkin, G. Orsi, N. Paton, Data wrangling for big data: Challenges and opportunities, in: Advances in Database Technology—EDBT 2016: Proceedings of the 19th International Conference on Extending Database Technology, 2016, pp. 473–478.

[18] G. Jaimovitch López, Comparison between machine learning and human learning from examples generated with machine teaching, 2020.

[19] W. Zeng, X. Ren, T. Su, H. Wang, Y. Liao, Z. Wang, X. Jiang, Z. Yang, K. Wang, X. Zhang, et al., Pangu-α: Large-scale autoregressive pretrained chinese language models with auto-parallel computation, arXiv preprint arXiv:2104.12369 (2021).

[20] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, B. Zoph, L. Fedus, M. Bosma, Z. Zhou, T. Wang, Y. E. Wang, K. Webster, M. Pellat, K. Robinson, K. Meier-Hellstern, T. Duke, L. Dixon, K. Zhang, Q. V. Le, Y. Wu, Z. Chen, C. Cui, GLaM: Efficient Scaling of Language Models with Mixture-of-Experts, arXiv:2112.06905 [cs] (2021). arXiv:2112.06905.

[21] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, L. Zettlemoyer, OPT: Open Pre-trained Transformer Language Models, arXiv:2205.01068 [cs] (2022). arXiv:2205.01068.

[22] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei, Scaling laws for neural language models, CoRR abs/2001.08361 (2020). URL: https://arxiv.org/abs/2001.08361. arXiv:2001.08361.

[23] R. Desislavov, F. Martínez-Plumed, J. Hernández-Orallo, Compute and energy consumption trends in deep learning inference, arXiv preprint arXiv:2109.05472 (2021).

[24] O. Sharir, B. Peleg, Y. Shoham, The cost of training NLP models: A concise overview, arXiv preprint arXiv:2004.08900 (2020).

[25] B. Dickson, The GPT-3 economy, https://bdtechtalks.com/2020/09/21/gpt-3-economy-business-model/, 2020.

[26] D. Steinberg, How much time needs to be spent preparing data for analysis?, http://info.salford-systems.com/blog/bid/299181/How-Much-Time-Needs-to-be-Spent-Preparing-Data-for-Analysis (2013).

[27] A. Bogatu, N. W. Paton, A. A. Fernandes, Towards automatic data format transformations: Data wrangling at scale, in: British International Conference on Databases, Springer, 2017, pp. 36–48.

[28] S. Gulwani, J. Hernández-Orallo, E. Kitzelmann, S. H. Muggleton, U. Schmid, B. Zorn, Inductive programming meets the real world, Communications of the ACM 58 (2015) 90–99.

[29] L. Contreras-Ochando, C. Ferri, J. Hernández-Orallo, F. Martínez-Plumed, M. J. Ramírez-Quintana, S. Katayama, Automated data transformation with inductive programming and dynamic background knowledge, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2019, pp. 735–751.

[30] S. Kandel, A. Paepcke, J. Hellerstein, J. Heer, Wrangler: Interactive visual specification of data transformation scripts, in: Proceedings of the sigchi conference on human factors in computing systems, 2011, pp. 3363–3372.

[31] T. De Bie, L. De Raedt, J. Hernández-Orallo, H. H. Hoos, P. Smyth, C. K. I. Williams, Automating data science, Communications of the ACM 65 (2022) 76–87. doi:10.1145/3495256.

[32] R. Puri, B. Catanzaro, Zero-shot text classification with generative language models, arXiv preprint arXiv:1912.10165 (2019).

[33] T. Schick, H. Schütze, Exploiting cloze questions for few shot text classification and natural language inference, arXiv preprint arXiv:2001.07676 (2020).

[34] G. Jaimovitch-Lopez, C. Ferri, J. Hernandez-Orallo, F. Martınez-Plumed, M. J. Ramırez-Quintana, Can language models automate data wrangling?, in: Workshop on Automating Datascience at ECML-PKDD, 2021, p. 13.

[35] B. Zoph, I. Bello, S. Kumar, N. Du, Y. Huang, J. Dean, N. Shazeer, W. Fedus, Designing effective sparse expert models, arXiv preprint arXiv:2202.08906 (2022).

[36] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[37] A. Murphy, A New Vector Partition of the Probability Score., Journal of Applied Meteorology 12 (1973) 595–600.

[38] P. Flach, E. Matsubara, On classification, ranking, and probability estimation, in: Dagstuhl Seminar Proceedings, Schloss Dagstuhl-Leibniz-Zentrum fr Informatik, 2008.

[39] J. Hernández-Orallo, P. Flach, C. Ferri Ramírez, A unified view of performance metrics: Translating threshold choice into expected classification loss, Journal of Machine Learning Research 13 (2012) 2813–2869.

[40] A. Bella, C. Ferri, J. Hernández-Orallo, M. J. Ramírez-Quintana, Calibration of machine learning models, in: Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques, IGI Global, 2010, pp. 128–146.