Complex & Intelligent Systems (2024) 10:8287–8317
https://doi.org/10.1007/s40747-024-01599-6

ORIGINAL ARTICLE

A general supply-inspect cost framework to regulate the reliability-usability trade-offs for few-shot inference

Fernando Martínez-Plumed[1] · Gonzalo Jaimovitch-López[1] · Cèsar Ferri[1] · María José Ramírez-Quintana[1] · José Hernández-Orallo[1]

Received: 31 July 2022 / Accepted: 27 July 2024 / Published online: 19 August 2024

Abstract
Language models and other recent machine learning paradigms blur the distinction between generative and discriminative tasks, in a continuum that is regulated by the degree of pre- and post-supervision that is required from users, as well as the tolerated level of error. In few-shot inference, we need to find a trade-off between the number and cost of the solved examples that have to be supplied, those that have to be inspected (some of them accurate but others needing correction) and those that are wrong but pass undetected. In this paper, we define a new Supply-Inspect Cost Framework, associated graphical representations and comprehensive metrics that consider all these elements. To optimise few-shot inference under specific operating conditions, we introduce novel algorithms that go beyond the concept of rejection rules in both static and dynamic contexts. We illustrate the effectiveness of all these elements for a transformative domain, data wrangling, for which language models can have a huge impact if we are able to properly regulate the reliability-usability trade-off, as we do in this paper.

Keywords Few-shot inference · Language models · Evaluation · Reliability · Usability

Introduction

In many machine learning (ML) applications, the degree of automation is partial by definition and dominated by trade-offs. Users need to supervise the process at different stages, mostly through the labelling of training or contextualising examples, and the inspection of some of the results from the ML system, to check that the outcome meets the desired quality. In many tasks, especially those of generative character or when predictions are structured, the creation or labelling of examples is more costly than their inspection. For instance, Fig. 1 shows a transformation where visually inspecting the result is much faster than writing it directly. Many tedious manipulation tasks are of this kind, such as wrangling with spreadsheets and other sources of text and data.

In recent times, few-shot learning [1] has been proposed as a new machine learning paradigm able to learn from only a few supervised examples, thus alleviating the problem of having large labelled training sets. Few-shot approaches have been successfully applied in areas such as fault diagnosis [2] and image semantic segmentation [3, 4].

Some recent language models (LMs) such as the GPT family [5, 6], PanGu-α [7], PaLM [8], BLOOM [9] or Llama [10] have excelled at few-shot inference, where a task is solved by supplying a small set of correct examples formatted as a *prompt* [11]. The quality of the completion usually depends on the number of supplied examples n_s. For instance, 5-shot inference is usually better than 2-shot inference, but requires more effort from the user. Both the cost of supplying and the cost of inspecting each example are elements of the *operating condition*. On top of this, some tasks or users may have different error tolerances, which is another component of the operating condition. The latter can be adjusted by the use of reject options based on a confidence threshold t [12–14]. Some completed examples have sufficient confidence to go through, but others are rejected to another system. However,

✉ Fernando Martínez-Plumed
fmartinez@dsic.upv.es

Gonzalo Jaimovitch-López
gonjailo@upv.es

Cèsar Ferri
cferri@dsic.upv.es

María José Ramírez-Quintana
mramirez@dsic.upv.es

José Hernández-Orallo
jorallo@upv.es

1 VRAIN, Universitat Politècnica de València, Valencia, Spain

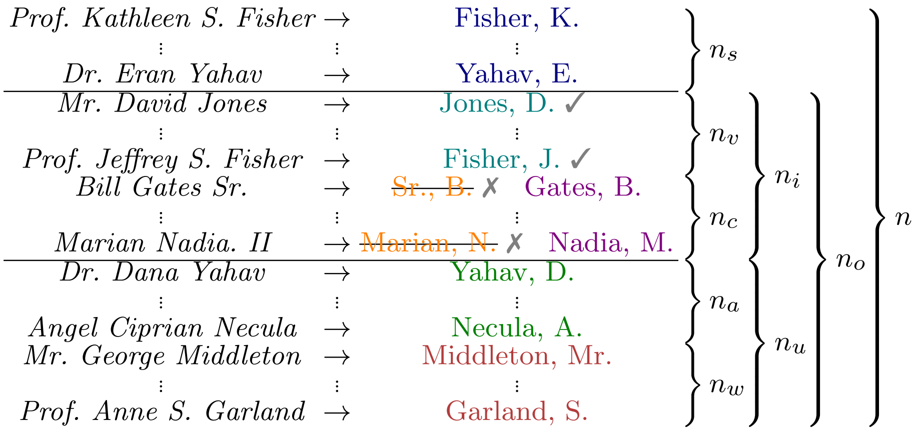

Prof. Kathleen S. Fisher → Fisher, K.
Dr. Eran Yahav → Yahav, E.
Mr. David Jones → Jones, D. ✓
Prof. Jeffrey S. Fisher → Fisher, J. ✓
Bill Gates Sr. → ~~Sr., B.~~ ✗ Gates, B.
Marian Nadia. II → ~~Marian, N.~~ ✗ Nadia, M.
Dr. Dana Yahav → Yahav, D.
Angel Ciprian Necula → Necula, A.
Mr. George Middleton → Middleton, Mr.
Prof. Anne S. Garland → Garland, S.
n_s
n_v
n_c
n_a
n_w
n_i
n_u
n_o
n

Fig. 1 Example of a name transformation problem. In a few-shot inference session with a language model (prompt not shown), n_s examples were completely supplied by a human (in blue) and the rest (n_o) were completed by the system. From these, n_i were inspected, of which n_v were correct and validated (in turquoise) and n_c were wrong (in orange) and corrected (in purple). Finally, the remaining examples n_u were not inspected, of which n_a were accurate and n_w were wrong (in red). The horizontal solid lines represent the thresholds for the two main choices to be made: how many examples are supplied (n_s), and inspected (n_i). Note that the only truly rejected instances are the ones in orange, crossed out by the user

if the user inspects some of these examples, and decides to correct them, they immediately become good examples that could be used to retrain or tune the model. In the case of few-shot learning with LMs, they could be used to enlarge and rerun the *prompt* with better accuracy and confidence estimation for new examples. This shows that the traditional reject option approach is insufficient: to account for this situation, we need a new framework and algorithms that can deal interactively in this situation.

We look at the balance between reliability and usability to determine the optimal number of few-shot examples. This approach aims to minimise the total cost of providing and exploring examples, and of accounting for undetected errors. Unlike active learning [15], our exploration does not rely on the training algorithm to select examples. Instead, an external process takes responsibility for selecting the example size and identifying the best cardinality n_s and threshold t based on the prevailing operating conditions. This departure from active learning is crucial because the examples that add the most value to active learning are the same ones that could potentially increase the frequency of costly n_c cases. Compared to active learning (where more examples are used for training the model) and threshold choice for reject options (where this is decided after the model has been learned), here we need to play with the prompt (and the number of instances used in the few-shot process, interactively) to reach an optimal trade-off.

To show the effectiveness of our approach, we choose an application where both reliability and usability are critical: data wrangling transformations [16]. Also, the degree of automation is partial by definition and dominated by trade-offs [17]. The user has to give at least one instance of the transformation so that *most of* the rest are completed by the system, but could give more instances if that compensates for fewer examples the user has to inspect and possibly correct, and of course fewer errors. We apply our algorithms to 123 tasks from 7 different domains, for which we also estimate reasonable operating conditions from a human study. This represents the first benchmark with annotated human reliability-usability conditions for the evaluation of LMs.

The main contributions of this paper are:

1. We formalise a novel methodology for few-shot inference based on the trade-off between reliability and usability through a new cost framework integrating all the relevant elements in few-shot learning, including the number of examples provided, examples inspected, and errors not detected.
2. We analyse how the number of examples provided to the model affects not only the accuracy of the outputs but also the model confidence, represented as logprobs.
3. We devise an original graphical representation called 'supply-inspect cost surfaces', as the method for selecting the optimal thresholds (number of supplied examples and the degree for inspection) given the operating condition. We show that the volume under this surface, when the axes conform with the expected operating distribution, is equal to the expected cost.
4. We establish several innovative static and dynamic algorithms to reduce the expected cost given the operating condition. Their performance is presented both experimentally and theoretically, showing that these algorithms approximate the optimal trade-off between reliability and usability.

5. We release a benchmark containing a substantial number of 123 tasks across 7 domains. This benchmark is annotated with information on the plausible range of operating conditions derived from real questionnaires completed by human users.

The paper is structured as follows. Section Supply-inspect cost framework introduces the framework to formalise diverse costs for few-shot learners. In Sect. Threshold choice method, we propose different threshold methods that can be employed for the addressed problem. Section Supply-inspect surfaces and expected cost includes some theoretical results about supply-inspect surfaces and the expected cost. The experimental setting is described in Sect. Experimental design, while the results of the experiments are discussed in Sect. Results. Lastly, we include a section for related work and a final section with the closing remarks, including a discussion about the wide applicability of this work, its limitations and questions for future work.

Supply-inspect cost Framework

Consider a problem space $\mathcal{D}$ for which a human user wants to solve a finite set $D \subset \mathcal{D}$ of $n = |D|$ instances $x \in D$ as accurately and efficiently as possible. The problem may be discriminative or generative. With the help of an AI model M that can do few-shot 'learning', the user may choose a small set of examples $D_s \subset D$, with $n_s = |D_s|$, add a correct output for each of them, and supply the labelled dataset to the model. M is now contextualised with D_s (e.g., via a prompt) and outputs the answers for $D_o = D \backslash D_s$. If the user is concerned by the errors of the model, a possible solution could be to increase n_s, since the results for D_o are expected to be better as we provide more information to M.

However, reaching high reliability with this schema for some few-shot inference systems such as LMs may be infeasible, even with large D_s. If the error tolerance is low, the user may introduce a reject option [14, 18, 19]. In the most common incarnation of this schema, if the model outputs a confidence value $\hat{p}(x)$, e.g., the probability of being correct for each instance x, we can define a reject rule: if $\hat{p}(x) \leq t_r$, with t_r being the reject threshold, then the user will not use the output of the model. But the rejected examples can be manually inspected by the user and solve them herself. This is what Fig. 2 shows for a dates processing problem. Looking at the left plot (1-shot), for different reject thresholds (shown on the x-axis) and $n = 32$ examples, the proportion of accurate, wrong and rejected examples evolves from about 85% accurate vs 12% wrong for $t_r = 0$ (no rejection) and 97% rejected for $t_r = 1$ (being the supplied examples the remaining 3%). The hot spot is found somewhere between $t_r = 0.4$ and $t_r = 0.6$, with very few errors but the system automating more than 80% of the examples. Of course, this may be considered an insufficient automation with an unacceptable number of errors, and the alternative is to supply the model with further labelled examples. This is what we see in Fig. 2, where n_s is in $(1..4)$. In this particular example, we see that the improvement saturates for $n_s = 3$ and a very good spot is found in that plot for $t_r = 0.7$, giving about 90% accurate results with the rest being rejected.

This traditional view of rejection neglects an important aspect: many of the initially rejected examples were actually correct! Rather than rejecting the output of the model, the user has the option of inspecting these unreliable examples. There are two possible outcomes to the inspection: sometimes the user has to correct the example, but in many other cases the user only needs to validate it. The latter scenario generally requires much less effort than the former. In Fig. 1, we can observe this distinction through the examples marked in orange, which represent incorrect instances that required correction, and those marked in turquoise, which were verified as correct and could then be confidently used by the model.

This set of inspected examples we now denote by $D_i \subset D_o$. The big insight and refinement from the concept of rejection is that we can split D_i into two different sets, D_v, the examples correctly labelled by M and hence validated by the user, and D_c, the examples incorrectly labelled by M, which must also be corrected (these are the truly rejected ones). Finally, the examples that go uninspected (D_u) can also be divided into accurate D_a and wrong D_w. Following the previous notation, we have that $n_\bullet = |D_\bullet|$ for $\bullet \in \{a, c, u, v, w\}$, where $n = n_s + n_i + n_u$, $n_i = n_v + n_c$ and $n_u = n_a + n_w$, which is what we see in Fig. 1. We usually want n_s very low and n_i low for usability, with n_w very low for reliability.

Now let us consider several *cost functions* $f_\bullet$ for each of the previous sets $D_\bullet$ as a function of the number of elements $n_\bullet$. The global cost to minimise is:

$$Q \overset{\text{def}}{=} f_s(n_s) + f_v(n_v) + f_c(n_c) + f_a(n_a) + f_w(n_w)$$

It is customary to define utility functions that depend linearly on the number of examples. Under this assumption, we have:

Proposition 1 *Assuming all functions $f_\bullet$ are linear in $n_\bullet$ of the form $f_\bullet(n_\bullet) = c_\bullet \cdot n_\bullet$, we have that:*

$$Q = c_s \cdot (n_s + n_c) + c_i \cdot n_i + c_w \cdot n_w \tag{1}$$

where c_s is the unitary cost for the user to solve an example, c_i is the unitary cost for the user to inspect an example and c_w is the unitary cost of an unspotted wrong example.

The proofs for this and all the other theoretical results in the paper can be found in the appendix.

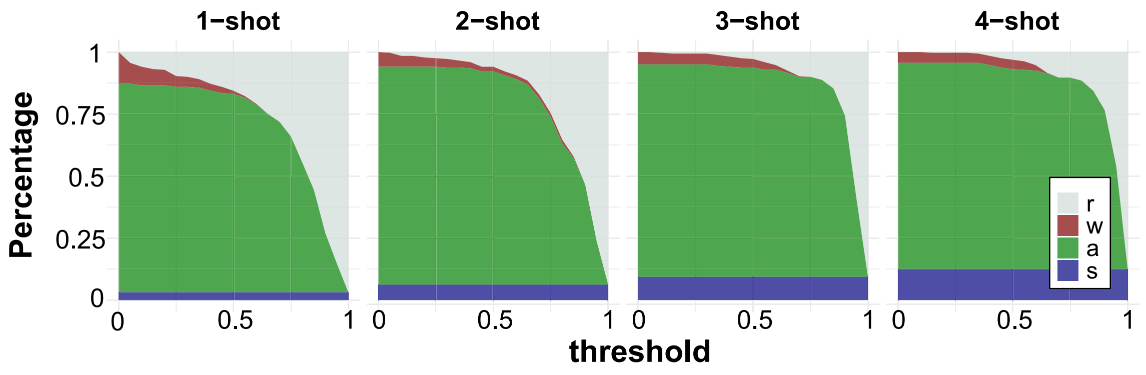


Fig. 2 Reject option behaviour averaged for 10 dates formatting problems with $n = 32$ instances each. The curves show average proportion of examples (supplied $\frac{n_s}{n}$ in blue, accurate $\frac{n_a}{n}$ in green, wrong $\frac{n_w}{n}$ in red and rejected $\frac{n_r}{n}$ in grey) as we increase the reject threshold t_r in the x-axis. The four plots show the evolution for different values of n_s in (1..4)

Proposition 1 establishes that, in principle, we only need to know three cost constants: c_s, c_i and c_w, which entail only two degrees of freedom, as a multiplicative factor over all costs does not change a selection. This means that we can use *ratios* instead, and we define the *operating condition* as:

$$\mathbf{c} \overset{\text{def}}{=} \left\langle \frac{c_s}{c_w}, \frac{c_i}{c_w} \right\rangle \tag{2}$$

Similarly, any solution to this problem only needs two thresholds. We can first determine n_s, i.e., how many examples the user supplies, as the quality of the inferences will depend on it (and hence all the other $n_\bullet$). For this, we will seek for a good threshold t_s statically and dynamically in the following sections. In a few-shot scenario, the n_s that derives from t_s is expected to be a small number. Once D_s is given to the model, we get the confidence for all the other examples D_o. From here, a static method should determine the inspection threshold t_i (a confidence below which we decide to inspect, replacing the threshold t_r in the traditional reject option scenario). This will determine the numbers n_a, n_c, n_v and n_w. We can integrate both thresholds (the two horizontal lines in Fig. 1) into a vector $\mathbf{t}$:

$$\mathbf{t} \overset{\text{def}}{=} \langle t_s, t_i \rangle \tag{3}$$

The supply threshold $t_s \in [0, 1]$ determines

$$n_s \overset{\text{def}}{=} \nu_{\alpha,n}(t_s) \overset{\text{def}}{=} n\frac{\alpha^{t_s} - 1}{\alpha - 1} \tag{4}$$

with $\alpha > 1$ being a large constant so that the scale focuses on small sets D_s. The inspection threshold $t_i \in [0, 1]$ sets

$$n_i \overset{\text{def}}{=} n_i(\mathbf{t}) \overset{\text{def}}{=} |\{x \in D_o : \hat{p}(x) \leq t_i\}| \tag{5}$$

where $\hat{p}(x)$ is the model's confidence for each instance x.

We can now rewrite Eq. 1 as a function of vectors $\mathbf{t}$ and $\mathbf{c}$. Since the operating condition $\mathbf{c}$ in Eq. 2 only has two degrees of freedom, the cost just differs by a multiplicative factor c_w:

Proposition 2 *Q can be expressed on the thresholds* $\mathbf{t}$ *and only the two components of* $\mathbf{c}$*:*

$$Q(\mathbf{t}; \mathbf{c}) = c_s \cdot (\nu_{\alpha,n}(t_s) + n_c(\mathbf{t})) + c_i \cdot n_i(\mathbf{t}) + n_w(\mathbf{t})$$

From now on, we will simply consider $c_w = 1$ being the *cost unit*, so that c_s and c_i are the two components in $\mathbf{c}$, which we will call supply cost (ratio) and inspect cost (ratio).

Given $\mathbf{t}$ and $\mathbf{c}$, in order to calculate Q from a test set, we proceed as follows. On the test set of n examples, we denote by $p(x)$ (with $p(x) \in \{0, 1\}$ for all x) whether the model is right (1) or wrong (0) with an example x. We have

$$n_c = \sum_{x \in D_i} (1 - p(x))$$

and

$$n_w = \sum_{x \in D_u} (1 - p(x))$$

This completes all the $n_\bullet$ for calculating Q.

Algorithm 1 implements the definition of Q as per Propositions 1 and 2. Note that, this algorithm performs a sample in line 1 (the notation $\overset{n_s}{\sim}$ means a sample of size n_s), so the exact actual cost when given fixed threshold and cost vectors $\mathbf{t}$ and $\mathbf{c}$ would be obtained by considering all possible combinatorial samples of n_s elements from n without replacement $\binom{n}{n_s}$. But as the order of examples for LMs matters [20], the exact Q would be given by considering all permutations. In practice, making the sample multiple times and averaging the results can give a good approximation for Q, which is what the parameter m means (number of samples).

Algorithm 1 $Q(M; \mathbf{t}; \mathbf{c}; \alpha; D; p; m)$

```
$n \leftarrow |D|$ ; $n_s = \nu_{\alpha,n}(t_s)$
$\dot{q} \leftarrow 0$
repeat
    $\dot{q} \leftarrow \dot{q} + c_s \cdot n_s$                          ▷ Supplied
    $D_s \overset{n_s}{\sim} D$ without replacement
    $D_o \leftarrow D \backslash D_s$                    ▷ Prompt $M$ with $D_s$ for $D_o$
    $n_o \leftarrow |D_o|$
    $\hat{p}(x)$ from $M$ $\forall x \in D_o$                  ▷ Model's confidence
    Order all $x \in D_o$ by increasing $\hat{p}(x)$
    $D_i \leftarrow \{x \in D_o : \hat{p}(x) \le t_i\}$

    $n_i \leftarrow |D_i|$
    ▷ Using $p(x) \in \{0, 1\}$ ($M$ wrong or right)
    $\dot{q} \leftarrow \dot{q} + c_s \cdot \sum_{x \in D_o[1:n_i]} (1 - p(x))$     ▷ Corrected
    $\dot{q} \leftarrow \dot{q} + c_i \cdot n_i$                          ▷ Inspected
    $\dot{q} \leftarrow \dot{q} + \sum_{x \in D_o[(n_i+1):n_o]} (1 - p(x))$       ▷ Wrong
until $m$ times
$q \leftarrow \frac{\dot{q}}{m}$
return $q$
```

Threshold choice methods

A threshold choice method T takes a cost vector $\mathbf{c}$ and possibly other parameters and returns $\mathbf{t}$ for a given model M.

Optimal method T^o

If we have access to the true p, we can easily define the optimal threshold choice method:

$$T^o(\mathbf{c}) \overset{\text{def}}{=} \mathbf{t}^* \overset{\text{def}}{=} \arg\min_{\mathbf{t}} Q(\mathbf{t}; \mathbf{c}) \tag{6}$$

When we use this method to derive the threshold, we have $Q(\mathbf{t}^*; \mathbf{c})$, represented more shortly as $Q^o(\mathbf{c})$. The optimal threshold can be calculated with Algorithm 2 which is an exhaustive grid search[1] looking for the optimal threshold $\mathbf{t}^*$, as defined in the T^o threshold method. It is implemented by iterating on the number of supplied examples $s \in [1..n]$ (first loop starting at line 3) and threshold $\theta \in [0..1]$ using a sufficient resolution ϵ (inner loop starting at line 5). ϵ must be small enough to find any threshold θ that could appear between two consecutive $\hat{p}(x)$.

Fixed method T^Φ

In practice, we do not have access to the true p, so any method will usually give suboptimal results when exploring the trade-offs. For instance, the higher t_s the higher the part of the cost that comes from n_s. However, this will usually entail better predictions and confidence, reducing the number of examples n_i that have to be inspected and the final number of wrong

[1] It could also be used any other optimization algorithm for finding the minimum of a function, such as gradient descent.

Algorithm 2 $T^o(M, \mathbf{c}; \alpha; D; p, m)$

```
$q_{best} \leftarrow \infty$ ; $n \leftarrow |D|$
for $s \in 1$ to $n$ do
    $t_s \leftarrow \nu^{-1}_{\alpha,n}(s)$
    for $\theta \in 0$ to $1$ with step $\epsilon$ do
        $\mathbf{t} \leftarrow \langle t_s, \theta \rangle$
        $q \leftarrow Q(M; \mathbf{t}; \mathbf{c}; \alpha; D; p; m)$
        if $q < q_{best}$ then
            $q_{best} \leftarrow q$ ; $n_s \leftarrow s$ ; $t_i \leftarrow \theta$
        end
    end
end
$t_s \leftarrow \nu^{-1}_{\alpha,n}(n_s)$
return $t_s, t_i$
```

examples n_w. So it seems the first choice must be t_s. We can assume a constant n_s and derive t_s accordingly, undoing ν in Eq. 4. We could do this for both t_s and t_i, choosing them in a *fixed* way that is independent of $\mathbf{c}$, such as T^ϕ making $t_s = \nu^{-1}_{\alpha,n}(5)$ and $t_i = 0.5$. This cost, denoted by Q^ϕ, would be obtained if the user always supplies 5 examples and inspects the remaining examples whose estimated confidence is lower than 0.5. The fixed method completely disregards the costs.

Static method T^σ

We call the first family of methods using the costs *static*, as they derive t_s just once and then t_i from it. When the algorithm decides t_s it still does not have access to the estimated probabilities. This family of methods makes the assumption that for $n_s = 0$ there is a baseline proportion of examples that will be right and this proportion usually increases as n_s grows. We assume the proportion of corrected over inspected is $\frac{n_c}{n_i} = \frac{1}{b_c}(1 - \frac{n_s}{n})$ and wrong over uninspected is $\frac{n_w}{n_u} = \frac{1}{b_w}(1 - \frac{n_s}{n})$. With this, from Proposition 1 we have:

$$g(n_s, n_i) \overset{\text{def}}{=} c_s \cdot \left(n_s + n_i \frac{n - n_s}{b_c n}\right) + c_i \cdot n_i + (n - n_i - n_s)\frac{n - n_s}{b_w n}$$

We are given c_s, c_i and n, so basically we have to find the pair $\langle n_s, n_i \rangle$ such that the above expression is minimised, with $n_s + n_i \le n$, $n_s > 0$, $n_i \ge 0$. This can be done with linear programming or any other solver, discard n_i and then keep n_s for the next step (and t_s comes from ν^{-1} in Eq. 4).

Once n_s has been decided, we choose n_s examples randomly from D that are labelled (by a human H) and supplied (D_s) to the model M, getting the results and probabilities for all other examples D_o. We calculate n_i using Eq. 5. If we take $\hat{p}(x)$ as good estimates or at least perfectly calibrated then

$$n_c \approx \sum_{x \in D_i} (1 - \hat{p}(x))$$

and

$$n_w \approx \sum_{x \in D_u} (1 - \hat{p}(x))$$

Using all this in Proposition 2, we just need to minimise:

$$c_s \cdot \sum_{x \in D_i} (1 - \hat{p}(x)) + c_i \cdot |\{x \in D_o : \hat{p}(x) \le t_i\}| + \sum_{x \in D_u} (1 - \hat{p}(x))$$

This is what Algorithm 3 calculates. It first determines the values n_s and n_i that minimise function g but only n_s is kept (line 2) to be used for selecting the sample D_s (line 4) to be labeled by the user (line 5). This supplied cost is calculated in line 6. The model M is prompted with D_s (line 7). Then, the model's confidence for D_o (the remaining examples in D) is obtained (line 9) and used for ordering the examples in D_o in increasing order of predicted confidence (line 10). Finally, the for loop (lines 12–20) performs an exhaustive search of the best inspection threshold t_i (line 18) for which the overall cost (lines 13 to 15) is minimum. Note that, in Algorithm 3 we cannot really consider that the method looks for all subsets D_s of size n_s in D, as trying each of them in practice will incur a cost from the human. Consequently, the static Algorithm 3 only has one run. In order to evaluate this and other methods (e.g., the dynamic one), we did repetitions outside the algorithm. Additionally, in Algorithm 3 the calculations using $\hat{p}$ consider the sums over decreasing index ranges, i.e., $x \in D_p[1:0]$ and $D_o[(n_o+1:n_o)]$ to be empty, and assume no ties (e.g., by adding a small random number to all $\hat{p}(x)$).

Algorithm 3 $T^\sigma(H, M, \mathbf{c}; \alpha; D)$

```
$n = |D|$
$\langle n_s, \cdot \rangle \leftarrow \min_{\langle n_s, n_i \rangle} g(n_s, n_i)$
$t_s \leftarrow \nu^{-1}_{\alpha,n}(n_s)$
$D_s \overset{n_s}{\sim} D$ without replacement
$p(x)$ from $H$ $\forall x \in D_s$                          ▷ Human labels $D_s$
$q_s \leftarrow c_s \cdot n_s$                                ▷ Supplied
$D_o \leftarrow D \backslash D_s$                            ▷ Prompt $M$ with $D_s$ for $D_o$
$n_o \leftarrow |D_o|$
$\hat{p}(x)$ from $M$ $\forall x \in D_o$                     ▷ Model's confidence
Order all $x \in D_o$ by increasing $\hat{p}(x)$
$q_{best} = \infty$
for $j \in 0$ to $n_o$ do
    $q \leftarrow q_s + c_s \sum_{x \in D_o[1:j]} (1 - \hat{p}(x))$     ▷ Exp. C'cted
    $q \leftarrow q + c_i \cdot j$                              ▷ Exp. Inspected
    $q \leftarrow q + \sum_{x \in D_o[(j+1):n_o]} (1 - \hat{p}(x))$     ▷ Exp. Wrong
    if $q < q_{best}$ then
        $q_{best} \leftarrow q$
        $t_i \leftarrow (\hat{p}(D_o[j]) + \hat{p}(D_o[j+1]))/2$
    end
end
return $t_s, t_i$
```

Dynamic method T^δ

The static algorithm assumes that the examples to be supplied to the model must be sampled at the beginning, and then use the model for the rest of the examples. Instead, we could take an incremental approach, where we supply very few examples D_s as start (let us say $|D_s| = s_0$), and infer the outputs for all the rest (D_o). While this may be very conservative in terms of c_s and may give poor results at this point, we can already use the model to rank the examples in D_o and choose just very few D_i for inspection (let us say $|D_i| = i_\oplus$). The insightful observation comes when we realise that some of them will be correct (and hence validated, D_v) and some of them will be incorrect (and hence corrected, D_c), but all of them can be reused for another iteration with the model with $D_s \cup D_v \cup D_c$ examples. Interestingly, while the new n_s includes all these examples, the elements in D_c have been inspected and supplied (with cost $c_i + c_s$) but the elements in D_v have only been validated (with cost c_i). This may represent an important saving, as the human can supply n_s examples to the model with lower cost than the original $c_s \cdot n_s$.

This observation leads to Algorithm 4. The algorithm takes the usual parameters but also s_0, $i_\oplus$, $s_\star$ (maximum number of iterations) and *rand* (true if the examples to be inspected are selected randomly, and false if it is done following $\hat{p}$), and it returns the thresholds t_s and t_i but it also returns two extra values: n_v and n_i. The value n_v represents the examples that were validated by the human and hence did not incur the c_s cost. The value n_i are the examples that were inspected before they were moved to D_s. These have to be used when calculating the cost of the dynamic algorithm: we have to add the cost of n_i and remove the cost of n_v when plugging Algorithms 4 and 1 together:

$$\langle t_s, t_i, n_v, n_i \rangle = T^\delta(H; M; \mathbf{c}; \alpha; D; s_0; i_\oplus; s_\star; rand)$$
$$Q^\delta(M; \alpha; D; p) = Q(M; \langle t_s, t_i \rangle; \mathbf{c}; \alpha; D; p; m) - c_s \cdot n_v + c_i \cdot n_i$$

Regarding the parameters s_0 and $i_\oplus$, the smaller the better, ideally $s_0 = i_\oplus = 1$.

If there is no tolerance for errors, any algorithm should inspect all examples. In this case, we can prove the following:

Proposition 3 *When no wrong results are permitted, the dynamic algorithm T^δ with $s_0 = i_\oplus = 1$ and $s_\star = |D|$ is optimal up to $c_i \cdot (n_s - 1)$ cost units provided the algo-*

rithm always orders examples by decreasing probability of being correct (more likely correct first).

Note that Proposition 3 dictates that in this extreme case where wrong examples are not allowed it could be more beneficial to choose the next $i_{\oplus}$ by decreasing $\hat{p}(x)$ rather than increasing $\hat{p}(x)$ as Algorithm 4 does when $rand$ is false. However, this would always increase the few-shot examples for the model with easy examples first, which would be less informative than using other more difficult (and corrected) examples. As using an increasing or decreasing order for this are extreme, we will finally use a random choice.

Algorithm 4 $T^{\delta}(H, M, \mathbf{c}; \alpha; D; s_0; i_{\oplus}; s_*; rand)$
default: $s_* \bot$; $rand$**true**

```
$n_s \leftarrow s_0; q_s \leftarrow c_s \cdot n_s$                                  ▷ Supplied
$D_s \overset{n_s}{\sim} D$ without replacement
  $p(x)$ from $H\ \forall x \in D_s$                                    ▷ Human labels $D_s$
$q_{old} \leftarrow q_{best} \leftarrow \infty$ ; $n_v \leftarrow n_i \leftarrow 0$; $out \leftarrow$ false
repeat
    $D_o \leftarrow D \setminus D_s$                              ▷ Prompt $M$ with $D_s$ for $D_o$
    $\hat{p}(x)$ from $M\ \forall x \in D_o$                              ▷ Model's confidence
    Order all $x \in D_o$ by increasing $\hat{p}(x)$

    if $s_\star \neq \bot$ then
        $out \leftarrow (s_\star \leq |D_s|)$
    else
        $out \leftarrow ((q_{best} \neq \infty) \wedge (q_{best} \geq q_{old}))$
    end
    if $\neg out$ then
                                                         ▷ $H$ inspects $i_{\oplus}$ examples
        if $rand$ then
            $D_i \overset{i_{\oplus}}{\sim} D_o$ without replacement
        else
            $D_i \leftarrow D_o[1..i_{\oplus}]$
        end
        $D_o \leftarrow D_o \setminus D_i$ ; $n_o \leftarrow |D_o|$
        $n_i \leftarrow n_i + |D_i|$ ; $q_i \leftarrow c_i \cdot n_i$                      ▷ Inspected
        $p(x)$ from $H\ \forall x \in D_i$                              ▷ $H$ inspects $D_i$
        $D_v \leftarrow \{x \in D_i : p(x) = 1\}$
        $n_v \leftarrow n_v + |D_v|$                                     ▷ Validated
        $D_c \leftarrow \{x \in D_i : p(x) = 0\}$
        $q_s \leftarrow q_s + c_s \cdot |D_c|$                                ▷ Corrected
        $D_s \leftarrow D_s \cup D_v \cup D_c$
    end
    ▷ Best estimate of the remaining cost:
    $q_{old} \leftarrow q_{best}$ ; $q_{best} \leftarrow \infty$
    for $j \in 0$ to $n_o$ do
        $q \leftarrow c_s \sum_{x \in D_o[1:j]} (1 - \hat{p}(x))$                   ▷ E. Corrected
        $q \leftarrow q + c_i \cdot j$                                 ▷ Exp. Inspected
        $q \leftarrow q + \sum_{x \in D_o[(j+1):n_o]} (1 - \hat{p}(x))$                    ▷ E. Wr
        if $q < q_{best}$ then
            $q_{best} \leftarrow q$
            $t_i \leftarrow (\hat{p}(D_o[j]) + \hat{p}(D_o[j+1]))/2$
        end
    end
    $q_{best} \leftarrow q_{best} + q_s + q_i$
until $out$
$n_s \leftarrow |D_s|$ ; $t_s \leftarrow \nu^{-1}_{\alpha,n}(n_s)$
return $t_s, t_i, n_v, n_i$                          ▷ $n_v$ discounts and $n_i$ adds
```

Supply-inspect surfaces and expected cost

For any threshold choice method μ, its *supply-inspect surface* is simply $Q^{\mu}(\mathbf{c})$ on the z-axis, where the other two axes are just the two components of the operating condition $\mathbf{c}$. Originally, the two components of $\mathbf{c}$ are ratios (as defined in Eq. 2), and they go from 0 to ∞, but many values will be close to 0, as it is usual that $c_s \ll c_w$ and $c_i \ll c_w$. To make the space finite and better accounting for the interesting regions of the space, we introduce two normalisation functions h_s and h_i, such that the x-axis is given by $h_s(c_s)$ (the supply cost coordinate) and the y-axis is given by $h_i(c_i)$ (the inspect cost coordinate). In what follows, we consider $h_s(a) \overset{\text{def}}{=} h_i(a) \overset{\text{def}}{=} 1 - \beta^{-a}$, with $\beta > 1$. With this we also have that the two axes are in $[0, 1[$ and the volume and the surface will be finite.[2] Coordinates can be mapped to costs simply by $h_s^{-1}(x) = -\log_\beta(1-x) = c_s$ and $h_i^{-1}(y) = -\log_\beta(1-y) = c_i$.

Fig. 3 shows a 'supply-inspect' surface using three techniques: the optimal, static and dynamic methods. The x-axis, ranging from 0 to 1, represents the relative cost of supplying an example compared to an incorrect result, with 0 representing a very low supplying cost. Similarly, the y-axis, also ranging from 0 to 1, reflects the relative inspection costs, with lower values indicating low inspection costs.

This visualisation helps us to understand the cost dynamics at play under different operating conditions, highlighting trade-offs and guiding the optimisation of models for practical applications. Basically, we can better understand the expectation on varying $\mathbf{c}$ beyond just a single point. If we assume a distribution ω on operating conditions and $\mathbf{c} \sim \omega$, we have the *expected cost* $\mathbb{E}_{\mathbf{c}\sim\omega}[Q(\mathbf{c})]$. The following holds:

Proposition 4 *Consider $\mathcal{H}$ the bivariate distribution that results on applying h_s and h_i to the two dimensions of ω. If $\mathcal{H}$ is a bivariate uniform distribution, then the volume under the supply-inspect surface is the expected cost.*

Corollary 1 *The volume in the supply-inspect space under $h_s(a) = h_i(a) = 1 - e^{-a}$ is equivalent to a weighted integral over the original space assuming an exponential distribution with $\lambda = 1$.*

The above corollary suggests that our normalisation of the space is actually assuming an exponential distribution on the costs with parameter $\lambda = 1$. Other parameters could be explored or even other distributions in the exponential family, such as the gamma distribution, but ours serves well as a standard to represent the surfaces in a bounded space $[0, 1[$.

As usual in other Pareto comparisons (e.g., ROC analysis [21, 22], or objective optimisation problems [23]), when two

[2] In what follows, the integrals will go from 0 to 1, but we have to consider the volumes are open on 1.

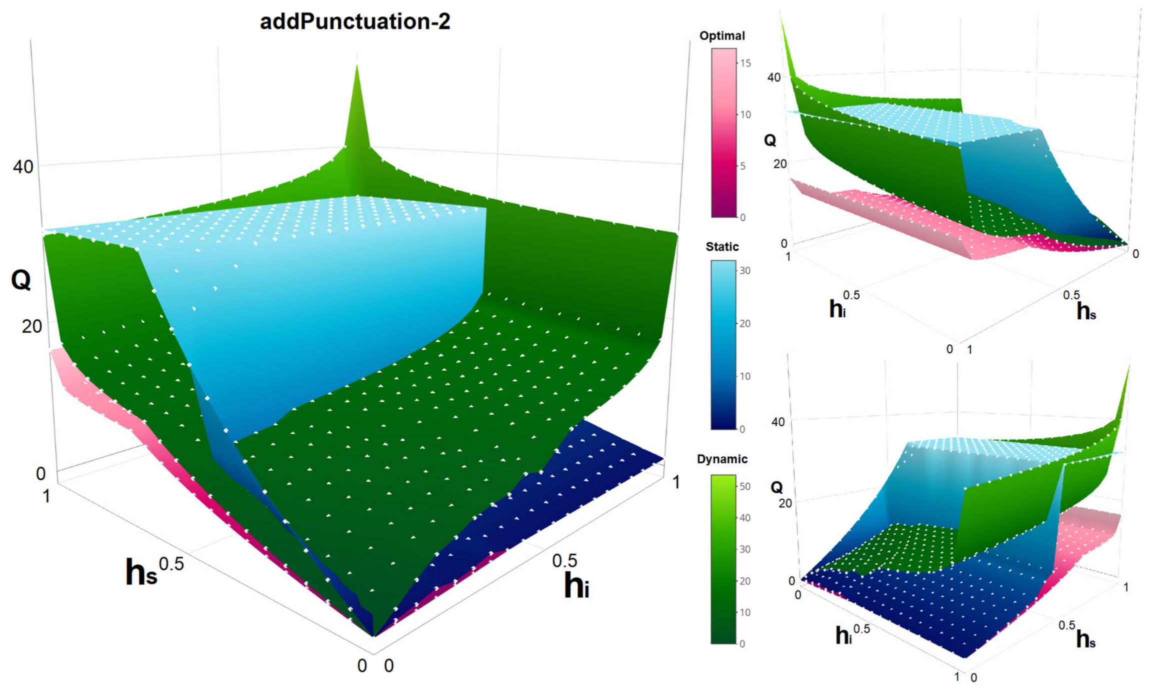


Fig. 3 Illustrative Supply-Inspect Surface (different views) using the T^o (red), T^σ (blue) and T^δ (green) for the task *addPunctuation* from the *dates* domain. See Tables 2 and 3 in the appendix for further details

surfaces cross, both have regions for which one is better than the other. One surface can only be safely discarded below the convex hull of some other surfaces. The volume (or expected loss) only seems an indication of how good a method is in expectation.

So far, we have considered that humans are perfect, but this is usually irrealistic, even if we take them as ground truth. In practice, we need to estimate e_s and e_i to account for the proportion of supplied examples and inspected examples respectively a human makes wrong. While this may suggest that we need to redo all our framework because of this, the following proposition and corollary show that we do not, provided we readjust the cost estimations.

Proposition 5 *Consider the same conditions as Proposition 1 but we now have a proportion of human error e_s and e_i for the supplied examples and inspected examples, respectively. The new cost equation becomes:*

$$Q = (c_s + c_w \cdot e_s) \cdot (n_s + n_c) + (c_i + c_w \cdot e_i) \cdot n_i + c_w \cdot n_w$$

Corollary 2 *We can express the cost when human errors exist as a readjustment of the normalisation of costs:*

$$Q = c'_s \cdot (n_s + n_c) + c'_i \cdot n_i + n_w$$

where $c'_s = \frac{c_s}{c_w} + e_s$ *and* $c'_i = \frac{c_i}{c_w} + e_i$.

This is a very elegant adjustment, as we only need to estimate the error rates and include them in the calculation of the operating condition. Everything else remains the same.

Experimental design

As discussed in the introduction, many routine tasks involve transforming inputs into outputs, such as converting some pieces of information into some standardised form. These tasks become interesting for (semi-)automation only if humans have to supply very few examples, and the errors in the uninspected results are unlikely. Consequently, these tasks are perfectly suited for few-shot learning under the supply-inspect cost framework introduced in this paper. Accordingly, we will use a repository of tasks built over the most comprehensive benchmark for data-wrangling transformation problems to date, the Data Wrangling Dataset Repository[3] [24, 25], which we have extended considerably[4] (see [26] for further information). Overall, the repository contains 123 different tasks divided into 7 different domains (*dates*, *emails*, *freetext*, *names*, *phones*, *times* and *units*). For every task we have 32 annotated examples where an input string is converted into a corrected or transformed version.

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

[4] https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/mult_data_wrangling

The appendix contains full details about the tasks (Table 2) and some illustrative examples (Table 3).

The experimental goals are:

1. Explore whether these problems are solvable with LMs in a few-shot fashion and determine whether there is a saturation point in the number of supplied examples.
2. Study whether the number of examples provided to the model affects not only the accuracy of the outputs but also their confidence ($\hat{p}$), so there is a trade-off between n_s vs n_c and n_w.
3. Determine how close the static and dynamic algorithms can get to the optimal cost, in comparison with the fixed method.
4. Derive and use reasonable cost distributions from the human study, and analyse how results differ from the uniform case.

For the experiments, we used four GPT-3 versions: Ada, Babbage, Curie and DaVinci with approximately 350 M, 1.3B, 6.7B, and 175B parameters, respectively. Following the recommendations in the OpenAI API[5] we used prompts following an input–output style, where the string "`Input:`" is used to indicate the start of the input, and the string "`Output:`" is used to indicate the start of the output. The line break `\n` separates the input from the output of an example, as well as the examples in the prompt. The instances have one (one-shot) or more (few-shot) given input–output pairs of the same problem and domain, and one single input ending the prompt. The model has have to provide the output by continuing the prompt. These are two one-shot examples (from different domains: *dates* and *times*):

Input: '290386'\nOutput: '29-03-86'\n\nInput: '250374'\nOutput:

Input: '980 ms'\nOutput: '0.98 s'\n\nInput: '1080 ms'\nOutput:

We obtain the confidence $\hat{p}$ that the model gives for the output as follows. If the model outputs the sequence of tokens $a_1, a_2, ...$, we trim the part that corresponds to the solution template. For these tokens we simply calculate the sum of the logprobs (or logarithm of probabilities that the model assigns to its generated outputs, which offers a measure of the model's confidence in its predictions) of all items and then convert this sum into the probability $\hat{p}$.

As we cannot really do repetitions without incurring real extra cost to the user, we calculate Q by performing only one sample of the n_s examples from D (n_s determined by each threshold method), and a lightweight implementation of T^o (see Algorithm 5 in the appendix). This ensures that our evaluation reflects practical constraints while still generating meaningful, actionable insights.

With the intention of obtaining optimal results, we carried out experiments with other methods using specific configurations. For the T^ϕ method we used $t_s = \nu^{-1}_{\alpha,n}(5)$ and $t_i = 0.5$, while for T^σ we used $b_c = 2$ and $b_w = 3$. For T^δ the method was run with $s_0 = i_\oplus = 1$ and a fixed number of interactions ($s_\star = 10$). While we experimented with several other parameter settings and variants for T^ϕ, T^σ and T^δ, the results obtained were either similar or inferior to those configurations. We therefore concluded that the chosen configuration gave the best performance. Also, given the combinatorial nature that would be required to evaluate all possible subsets D_s, our T^o approach serves as an upper bound estimate — an estimate that closely reflects an ideal baseline. The sensitivity of these parameters plays a crucial role in the outcome.

Table 1 Median values obtained from the questionnaires

Domain	τ_s	τ_i	χ/τ	χ_w	$\hat{c}_s$	$\hat{c}_i$
Dates	6.637	1.892	14	4	0.006	0.002
Emails	5.381	2.635	15	2	0.015	0.004
Freetext	3.520	3.102	15	3	0.006	0.005
Names	6.270	2.431	12	1	0.012	0.006
Phones	9.850	5.463	15	3	0.013	0.007
Times	9.254	3.892	15	5	0.006	0.003
Units	2.640	2.120	15	5	0.002	0.001

The two last columns show the operating conditions per domain. Mean and SD in Table 4 in the appendix

In order to estimate reasonable operating conditions, we conducted a questionnaire on 31 human subjects where we asked four questions for each of the seven domains. The first two questions measured the actual time for solving an instance (this time τ_s being a proxy for c_s) and the actual time for verifying an instance (this time τ_i being a proxy for c_i), averaged over five instances per question.

This was followed by a third subjective question asking the cost unit per time unit of a person (χ/τ), so that times could be converted into costs, and a fourth subjective question that asked about the cost of each error χ_w directly. We just derived $\chi_s = \tau_s \frac{\chi}{\tau}$ and $\chi_i = \tau_i \frac{\chi}{\tau}$. Finally, we divided both by χ_w to have the normalised costs in $\mathbf{c}$. That is, the estimate of the operating condition $\hat{\mathbf{c}}$ is given by $\hat{c}_s = \frac{\chi_s}{\chi_w}$ and $\hat{c}_i = \frac{\chi_i}{\chi_w}$. The results are shown in Table 1.

Finally, we also considered that humans may have errors, as we discussed around Proposition 5. In our questionnaires, humans were just given one example to infer the solution for all the other examples, so the error percentages we obtained are an overestimation of what trained humans would do for these domains. Nevertheless, the adjusted costs as per corollary 2 can be found in the appendix, and the recalculation of the expected costs for all methods. Even in these extreme conditions of human errors, the dynamic method is robust.

[5] https://openai.com/blog/openai-api/

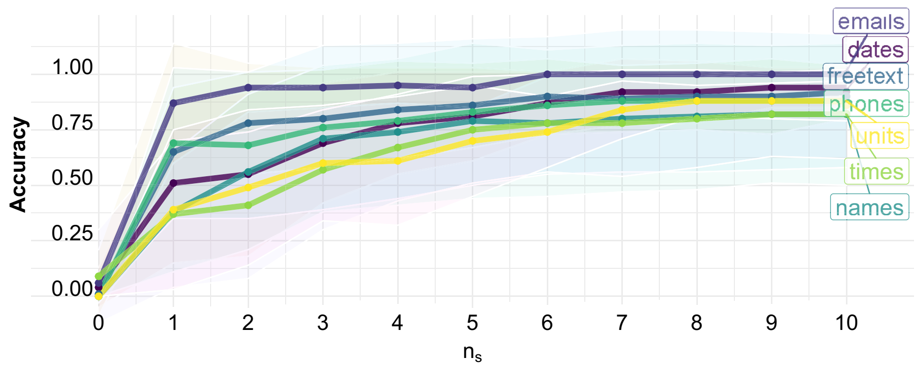


Fig. 4 Mean accuracy per domain for increasing values of n_s. Detailed results in Table 6 in the appendix regarding the accuracy per task for increasing values of n_s

Results

We follow the experimental goals sequentially.[6] In addressing goal (1), our main objectives were to investigate the feasibility of using LMs, to solve the problems illustrated by the Data Wrangling dataset. We also wanted to determine the exact saturation point for few-shot learning. Our analysis revealed interesting dynamics, as shown in Fig. 4. This plot illustrates the fluctuation of the accuracy achieved by the models, with the number of shots ranging from zero to ten across all the established domains. We can see an immediate, sharp increase in accuracy from zero shots to a one-shot scenario. This period of rapid growth then slows down into a more moderate and gradual increase until we reach around the 8 or 9-shot mark. After this point, the growth stabilises, suggesting the onset of a saturation point. From this data, we can confidently conduct our experiments with ten-shot at most for GPT-3, regardless of whether n_s is higher.

For goal (2), we focus on the model's confidence, denoted by $\hat{p}$, and its evolution as the number of examples is adjusted. Similar to our findings from goal (1), we observe a stabilisation around the ninth example, as shown in Fig. 5. In Fig. 6, we illustrate the trade-off between the number of examples we provide (n_s), the correctly predicted examples (n_a), and the incorrectly predicted examples (n_w). What we are essentially visualising here is how the ratio of correctly and incorrectly predicted examples increases and decreases as we steadily increase the number of examples provided (n_s). These fluctuations occur over different thresholds, which are set according to the model's confidence. It is important to note that within this particular example setting, the proportion of rejected examples, or n_r, would be the difference between 1 and the combined sum of the hit (accurate examples) and miss (inaccurate examples) ratios.

In pursuing goal (3), we use a supply-inspect framework in which the operational conditions of each domain follow a uniform distribution, denoted here as $\mathcal{H}$ as per Proposition 4. The volumes are calculated using a trapezoidal method over a grid layout. Figure 7 positions and compares the average expected cost for each domain. The cost distributions have been determined using data from human-led responses (opaque bars) and with uniform $\mathcal{H}$ (transparent bars). This comparison sets the benchmark at an optimal level (T^o) and measures the performance of static (T^σ), dynamic (T^δ) and fixed (T^ϕ) methods against this ideal standard. The transparent bars in this figure show how close both the static (T^σ) and dynamic (T^δ) algorithms are to the optimal cost for each domain. From our data, we see that T^ϕ outperforms T^σ in five of the total domains expressed, but T^δ remains superior to both.

To create a more realistic distribution of operating conditions, we include the results of c_s and c_i from the human-driven questionnaires, in line with our goal (4). Rather than simply averaging these operating conditions, we analyse each human response as a unique operating condition, expressed as $\langle c_s, c_i \rangle$. Each corresponding result of Q is calculated individually before being averaged together. Figure 7 (opaque bars) shows these results. We see a decrease in the overall magnitudes as the values become skewed towards lower ratios. A visual representation of this skew can be seen in Table 1. In six of the seven domains, T^ϕ lags behind, with the dynamic T^δ algorithm outperforming in all seven domains. In particular, T^δ comes very close to the optimal result in many cases. It should be emphasised that these data do not suggest that T^δ consistently outperforms the rest in all operating conditions. In fact, when compared to T^ϕ, which is optimal for a single operating condition, it is impossible to achieve complete dominance with T^δ. In general, surfaces cross as we saw in Fig. 3.

For a more detailed breakdown of our findings and results by domain, problem and methodology, we refer readers to Table 7 in our technical appendix.

[6] In compliance with the recommendations of the Science paper about reporting of evaluation results in AI [27], all the code, human questionnaire data, and results at the instance level can be found in https://github.com/nandomp/Trade-OffsFew-Shot.git.

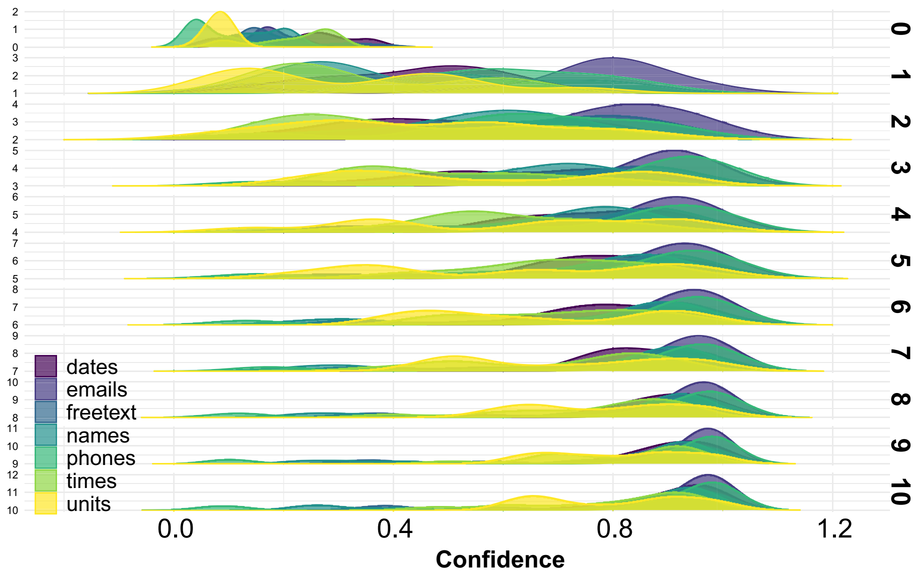


Fig. 5 Evolution of the distribution of model's confidence ($\hat{p}$) when varying the number of examples provided (y-axis)

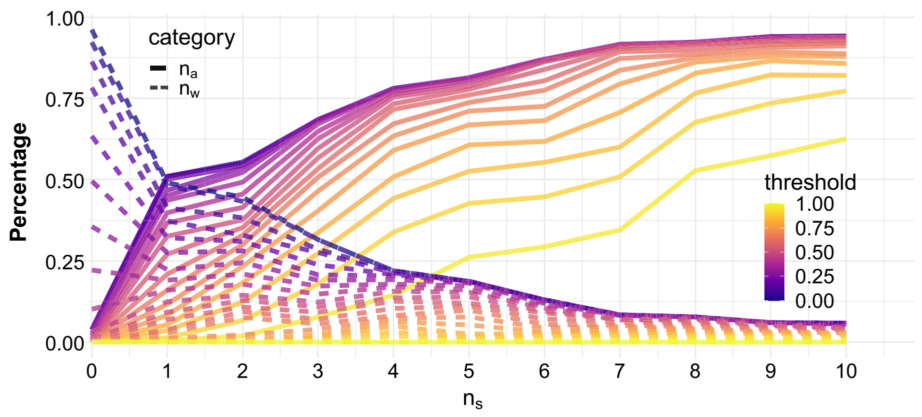


Fig. 6 Proportion of n_a and n_w examples for increasing n_s with different thresholds (in colour) for the dates domain (see Fig. 11 in the appendix for all domains)

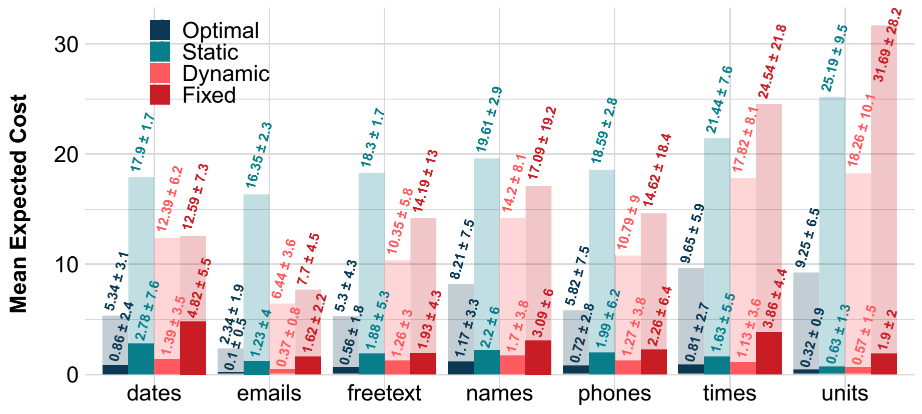


Fig. 7 Average expected costs per domain using the cost distributions from humans (opaque bars) and with uniform $\mathcal{H}$ (transparent bars) for the optimal (T^o), static (T^σ), dynamic (T^δ) and fixed (T^ϕ) methods. Detailed information per domain, problem and method in Table 7 in the technical appendix

Related work

The tension between reliability and usability goes beyond AI, since usability is related to the type and degree of supervision required from humans while providing a good quality of service [28–30]. However, many new tasks in AI, such as those provided by generative models [31–33], challenge this assumption. For instance, if a model generates images, inspecting and validating them is much cheaper for the user than creating or correcting them. An illustrative situation is few-shot learning [1, 34]. This is an important and increasingly more common way of using LMs, where template prompts accommodate an arbitrary number of examples [5, 11, 35–39].

The extension of this paradigm to other modalities is expected to happen soon [40]. However, to our knowledge, no previous work on LMs or few-shot inference has considered any realistic cost model to account for the reliability-usability trade-offs of these applications.

One general way to reduce the impact of classification errors is the use of a reject option [14] which determines the examples for which the classifier abstains. Reject options have been extensively studied for binary classification by optimising a certain objective cost function [41–46] or based on ROC analysis [47, 48].

On the other hand, a trade-off between performance and number of examples provided is also related to the area of active learning, where a learner iteratively chooses the training data by asking an oracle (usually the user) to label a few unlabelled examples [15, 49, 50]. A common query strategy is uncertainty sampling where the examples with the lowest confidence are selected first. Additionally, many active learning methods try to minimise annotation costs by reducing the number of examples to be labelled at each iteration. Although the most common scenario assumes the annotation cost is the same for all examples, some approaches also consider the cases in that the annotation cost vary between instances [51–55]. Active learning has also been studied for learning classifiers with reject option in an active way [56, 57] as an alternative to other (passive) methods that assume that a large labelled dataset is available. Nevertheless, to our knowledge, no active learning method reuses the classifier outputs to reduce the number of examples to be labelled by the user as our framework does. Finally, in other fields similar problems have been addressed by optimisation [2] or iterative learning [58].

Our static method, based on a cost-based thresholding function, is related to reject option methods and other threshold-choice methods that consider probability estimates [59]. However, in contrast to reject option approaches, the estimated threshold does not select the examples to be rejected but to be inspected, for which the labels estimated by the model are kept and used. Our framework is also general, going beyond any particular supervised task, and being especially applicable to those ML problems of generative character, where inspection costs are much smaller than supervision costs.

The dynamic algorithm might be considered as an active learning method with query strategies based on confidence, but not precisely selecting informative examples first. Again, the key difference of a supply-inspect cost framework is that the user inspects, rather than labels, the examples, and she only corrects those that are wrong, reducing the human cost since inspection is cheaper than correction. Consequently, there is a trade-off between preventing corrections and getting information from the user. In active learning, querying examples for which the model is correct is not informative and hence not pursued.

Even if active learning does not look for a trade-off between inspecting vs supplying costs, and hence the comparison is not really meaningful, we refer the reader to the appendix E for a comparative study of our methods against active learning. We show that even with a perfect example choice strategy having no errors at all ($Q_w = 0$), active learning is worse than all the supply-inspect methods introduced in this paper.

Conclusions and future work

The classical reject-option model is inappropriate for many old and new applications in AI, where humans play a more fluid role of pre-supervisors (supplying solved examples) and post-supervisors (inspecting examples provided by the system, and eventually correcting them). The new general supply-inspect framework introduced in this paper captures the need for adjusting the pre- and post-supervision efforts through the supply and inspect thresholds respectively. The dynamic algorithm shows that in scenarios where it is possible (and meaningful, as they are corrected by the user) to increase the number of examples incrementally as they are validated or corrected, we can obtain better results than fixed or static threshold choices. We have shown theoretical results about the framework (contributions 1 & 2), the supply-inspect space (contribution 3) and the algorithms (contribution 4). In practice, the space should be used to analyse how the surfaces from different threshold choice methods cross, helping decisions about their use depending on the operating conditions.

We have evaluated the feasibility of the results presented in this study from both an implementation and computational perspective. From an implementation perspective, we used a repository of tasks from the Data Wrangling Dataset Repository (contribution 5), containing 123 different tasks divided into 7 different domains, providing a broad scope for learning and testing the models. Additionally, the experimental procedures, which include exploring the solvability of tasks, studying the effects of the number of examples on model performance, comparing different algorithms, and deriving cost distributions from human studies, showed the implementability of the framework. From a computational perspective, we have used four well-known public versions of the GPT-3 model with varying numbers of parameters, from 350 M to 175B. This range of model complexity allowed us the study of trade-offs between computational resources and model performance.

The setting fits few-shot inference with LMs perfectly, but it has broad applicability to a range of problems in ML where the degrees of supply and inspect effort are variable, depending on the domain or the user. We have also illustrated that while the space of operating conditions is uncertain, an exponential distribution of c_s and c_i is appropriate as an aggregated metric. Nevertheless, we have had the rare determination of estimating realistic ranges of operating conditions from humans. While human questionnaires have many biases and limitations, we leave these estimated costs as meta-data for other researchers to conduct more realistic usability-reliability studies using some new methods.

Indeed, the static and dynamic algorithms may be improved in many ways, depending on the level of sophistication and some other information available during deployment. As presented in this paper, they are foundational for two major families of threshold-choice methods for this new supply-inspect paradigm, but more methods will surely come.

For instance, in the particular use of LMs, we also see potential for more sophisticated ways of choosing examples or prompts, inspired by recent research showing that not only the distribution of examples matters but also their order, or other ways to increase performance like calibration. In more general terms, we think this paper contributes to the recent trend of analysing the deployment of ML systems more holistically and taking human factors into account.

Appendix A

Here we include the proofs of all the theoretical results presented in the paper.

Proposition 1 (in main paper) *Assuming all functions $f_\bullet$ are linear in $n_\bullet$ of the form $f_\bullet(n_\bullet) = c_\bullet \cdot n_\bullet$, we have that:*

$$Q = c_s \cdot (n_s + n_c) + c_i \cdot n_i + c_w \cdot n_w$$

where c_s is the unitary cost for the user to solve an example, c_i is the unitary cost for the user to inspect an example and c_w is the unitary cost of an unspotted wrong example.

Proof From the original definition of cost we have:

$$Q = f_s(n_s) + f_v(n_v) + f_c(n_c) + f_a(n_a) + f_w(n_w)$$

We can first consider that uninspected but accurate instances have no cost, i.e., $f_a = 0$. This means:

$$Q = f_s(n_s) + f_v(n_v) + f_c(n_c) + f_w(n_w)$$

As both D_v and D_c are inspected but only D_c is corrected, we can separate the cost of inspection f_i from f_v and f_c. Hence, f_v is decomposed into the cost of inspection f_i and the cost of accurate instances f_a (we are assuming it is 0), and f_c is decomposed into the cost of inspection f_i and the cost of correcting f_f. Thus, we can rewrite the above equation as follows:

$$Q = f_s(n_s) + f_i(n_v) + f_i(n_c) + f_f(n_c) + f_w(n_w)$$

As f_i is linear, it is additive; so we have:

$$Q = f_s(n_s) + f_i(n_v + n_c) + f_f(n_c) + f_w(n_w)$$

The cost of a user producing the output for an instance that is supplied initially or after detecting an error is equal, as the user has to solve the instance in both cases. So we have that $f_s = f_f$, and $n_i = n_v + n_c$ and we finally simplify:

$$Q = f_s(n_s + n_c) + f_i(n_i) + f_w(n_w)$$

This can now be expressed in terms of cost constants:

$$Q = c_s \cdot (n_s + n_c) + c_i \cdot n_i + c_w \cdot n_w$$

□

Proposition 2 (in main paper) *Q can be expressed on the thresholds $\mathbf{t}$ and only the two components of $\mathbf{c}$:*

$$Q(\mathbf{t}; \mathbf{c}) = c_s \cdot (\nu_{\alpha,n}(t_s) + n_c(\mathbf{t})) + c_i \cdot n_i(\mathbf{t}) + n_w(\mathbf{t})$$

Proof We can rewrite the expression in Proposition 1 as a function of the thresholds.

$$c_s \cdot (\nu_{\alpha,n}(t_s) + n_c(\mathbf{t})) + c_i \cdot (n_i(\mathbf{t})) + c_w \cdot n_w(\mathbf{t})$$

We now simply extract c_w as a common factor.

$$c_w \cdot \left[\frac{c_s}{c_w} \cdot (\nu_{\alpha,n}(t_s) + n_c(\mathbf{t})) + \frac{c_i}{c_w} \cdot n_i(\mathbf{t}) + n_w(\mathbf{t})\right]$$

Because c_w is a constant multiplicative factor for any method or model, we get the result by assuming $c_w = 1$. □

Proposition 3 (in main paper) *When no wrong results are permitted, the dynamic algorithm with $s_0 = i_\oplus = 1$ and $s_\star = |D|$ is optimal up to $c_i \cdot (n_s - 1)$ cost units provided the algorithm always orders examples by decreasing probability of being correct (more likely correct first).*

Proof After calling T^δ, the cost is calculated as:

$$Q(M; \langle t_s, t_i \rangle; \mathbf{c}; \alpha; D; p; m) - c_s \cdot n_v + c_i \cdot n_i$$

In the extreme case where no wrong results are permitted and the given parameters, all inspections and corrections are inside the algorithm, no further inspections or corrections have to do in Q, and no wrong results happen, so plugging Proposition 1 we have:

$$c_s \cdot (n_s + 0) + c_i \cdot 0 + c_w \cdot 0 - c_s \cdot n_v + c_i \cdot n_i$$

Since $s_\star = |D| = n$, all examples will be supplied (except those that are validated) and all inspected but one (the first one, since $i_\oplus = 1$). This leads to:

$$c_s \cdot (n - n_v) + c_i \cdot (n - 1) \tag{A1}$$

Clearly, the only way of minimising this expression is by maximising n_v, which is achieved if the algorithm picks the elements that are inspected as those of highest probability of being correct first.

In order to say that this approach is almost optimal, we still need to show that Eq. A1 is not far from the optimal method. We start from Proposition 1 again, for the optimal method and full inspection we have:

$$c_s \cdot (n_s + n_c) + c_i \cdot n_i + c_w \cdot 0$$

Since we inspect all except the supplied, we have $n_i = n - n_s$; plugging this and reorganising we have:

$$\begin{aligned} &c_s \cdot (n_s + n_c) + c_i \cdot n - c_i \cdot n_s \\ &\quad c_s \cdot (n_s + n_c) - c_i \cdot (n_s - 1) + c_i \cdot (n - 1) \end{aligned}$$

Since $n_c = n - n_s - n_v$, we get

$$c_s \cdot (n - n_v) - c_i \cdot (n_s - 1) + c_i \cdot (n - 1)$$

Comparing this expression with Eq. A1 we get the term $c_i \cdot (n_s - 1)$, which is usually small. □

Proposition 4 (in main paper) *Consider $\mathcal{H}$ the bivariate distribution that results on applying h_s and h_i to the two dimensions of ω. If $\mathcal{H}$ is a bivariate uniform distribution, then the volume under the supply-inspect surface is the expected cost.*

Proof The volume under the supply-inspect surface is given by:

$$\int_0^1 \left[\int_0^1 Q(\langle h_s^{-1}(x), h_i^{-1}(y), \rangle) dx \right] dy$$

As $\mathcal{H}$ is a bivariate uniform distribution, this is an expected value:

$$\mathbb{E}_{\langle x,y\rangle \sim \mathcal{H}}[Q(\langle h_s^{-1}(x), h_i^{-1}(y)\rangle)]$$

Since $\mathcal{H}$ is the result of applying h_s and h_i to ω, we just get: $\mathbb{E}_{\mathbf{c}\sim\omega}[Q(\mathbf{c})]$. □

Corollary 1 (in main paper) *The supply-inspect space under $h_s(a) = h_i(a) = 1 - e^{-a}$ is equivalent to a weighted integral over the original space assuming an exponential distribution with $\lambda = 1$.*

Proof In the original space, we can define a weighted integral as follows:

$$\int_0^\infty \left[\int_0^\infty Q(\langle a, b\rangle)\omega(a)da \right] \omega(b)db$$

where $\omega(a) = \lambda e^{-\lambda a}$ is the density function of the exponential distribution.

Doing integration by substitution with $du = \omega(y)dy$ we get that $u = \int \omega(b)db = \Omega(b) = -e^{-\lambda b}$ and $\Omega^{-1}(u) = \frac{-1}{\lambda}\log(-u)$. This leads to:

$$\int_{-1}^0 \left[\int_0^\infty Q(\langle a, \Omega^{-1}(u)\rangle)\omega(a)da \right] du$$

Doing similarly with substitution with $dv = \omega(a)da$ we get that $v = \int \omega(a)da = \Omega(a) = -e^{-\lambda a}$ and $\Omega^{-1}(v) = \frac{-1}{\lambda}\log(-v)$. Now we have:

$$\int_{-1}^0 \left[\int_{-1}^0 Q(\langle \Omega^{-1}(v), \Omega^{-1}(u)\rangle)dv \right] du$$

Just making the variable changes $x = v + 1$ and $y = u + 1$ we get:

$$\int_0^1 \left[\int_0^1 Q(\langle \Omega^{-1}(x-1), \Omega^{-1}(y-1)\rangle)dx \right] dy$$

We have that $\Omega^{-1}(x-1) = -\frac{1}{\lambda}\log(1-x)$. Plugging it above we have:

$$\int_0^1 \left[\int_0^1 Q(\langle \frac{-1}{\lambda}\log(1-x), \frac{-1}{\lambda}\log(1-y)\rangle)dx \right] dy \tag{A2}$$

If we just make $h_s(a) = 1 - e^{-a}$ we have that $h_s^{-1}(x) = -\log(1-x)$, and if we just make $h_i(a) = 1 - e^{-a}$ we have that $h_i^{-1}(y) = -\log(1-y)$. By plugging these two expressions into Proposition 4, and setting $\lambda = 1$ we have an expression equal to Eq. A2. □

Proposition 5 (in main paper) *Consider the same conditions as Proposition 1 but we now have a proportion of human error of e_s and e_i for the supplied examples and inspected examples respectively. The new cost equation becomes:*

$$\begin{aligned} Q = &(c_s + c_w \cdot e_s) \cdot (n_s + n_c) + (c_i + c_w \cdot e_i) \cdot n_i \\ &+ c_w \cdot n_w \end{aligned}$$

Proof We extend Proposition 1 with the new errors

$$Q = c_s \cdot (n_s + n_c) + c_i \cdot n_i + c_w \cdot (n_w + e_s(n_s + n_c) + e_i \cdot n_i)$$

since the supplying error rate affects $n_s + n_c$ examples and the inspecting error rate affects n_i. By rearranging we get:

$$Q = (c_s + c_w \cdot e_s) \cdot (n_s + n_c) + (c_i + c_w \cdot e_i) \cdot n_i + c_w \cdot n_w$$

□

Corollary 2 (in main paper) *We can express the cost when human errors exist as a readjustment of the normalisation of costs, as follows:*

$$Q = c'_s \cdot (n_s + n_c) + c'_i \cdot n_i + n_w$$

where

$$c'_s = \frac{c_s}{c_w} + e_s \qquad c'_i = \frac{c_i}{c_w} + e_i$$

Proof From Proposition 5 and dividing by c_w as in Proposition 2, we get the expression. □

Appendix B

In this appendix, we introduce a Lightweight Variant of T^o.

Note that the Algorithm 2 we implemented in our experiments, calls Q, which has m repetitions. Alternatively, we can do a version that is more similar to the other algorithms, and iterates on j from 0 to n_o to find the optimal cut point. This would allow for higher efficiency escaping long loops for small ϵ values. Algorithm 5 is then a lightweight version of Algorithm 2. However, the lightweight version also has a sample from D at the beginning, so a better estimation of T^o would require repetitions too, between the outer and the inner loop. Since our results for few-shot learning on GPT-3 were fixed, we always chose the same sample, and this is why we do not need the repetitions for our experiments.

Algorithm 5 Lightweight $T^o(M, \mathbf{c}; \alpha; D; p)$

```
$q_{best} = \infty$; $n \leftarrow |D|$
for $s \in 1$ to $n$ do
    $D_s \overset{s}{\sim} D$ without replacement
    $q_s \leftarrow c_s \cdot s$                                ▷ Supplied
    $D_o \leftarrow D \backslash D_s$                           ▷ Prompt $M$ with $D_s$ for $D_o$
    $n_o \leftarrow |D_o|$
    $\hat{p}(x)$ from $M$ $\forall x \in D_o$                  ▷ Model's confidence
    Order all $x \in D_o$ by increasing $\hat{p}(x)$

    for $j \in 0$ to $n_o$ do
        ▷ Use $p(x) \in \{0, 1\}$ ($M$ wrong or right)
        $q \leftarrow q_s + c_s \sum_{x \in D_o[1:j]} (1 - p(x))$            ▷ C'cted
        $q \leftarrow q + c_i \cdot j$                                        ▷ Inspected
        $q \leftarrow q + \sum_{x \in D_o[(j+1):n_o]} (1 - p(x))$             ▷ Wrong
        if $q < q_{best}$ then
            $q_{best} \leftarrow q$
            $n_s \leftarrow s$
            $t_i \leftarrow (\hat{p}(D_o[j]) + \hat{p}(D_o[j+1]))/2$
        end
    end
end
$t_s \leftarrow \nu^{-1}_{\alpha,n}(n_s)$ return $t_s, t_i$
```

Appendix C

This section includes more information about the datasets used in the experiments. Table 2 presents the name and a short description of some of the tasks included in our repository of data wrangling tasks. Some examples of the tasks are shown in 3. In this table we also include information of the number of tasks per domain. Complete details of the tasks and the examples of them can be found in the BigBench [26] benchmark repository.[7] All the code and results and visualisations can be found in https://github.com/nandomp/Trade-OffsFew-Shot.git.

[7] https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/mult_data_wrangling

Table 2 Categorisation of data manipulation tasks within the wrangling repository, organised by specific domains and detailing the transformation from input to expected output format

Task description	Expected output
Domain dates	
Add punctuation	The date in numeric format split by a punctuation sign
Change format	The date in one particular format
Change punctuation	The date in one particular format
Get day	The day in numeric format
Get day ordinal	The day in numeric ordinal format
Get month name	The name of the month
Get week day	The name of the weekday
Reduce month name	The name of the month reduced to three letters
Set format	The date split in DMY format
Domain email	
Generate Email	An email account created with the name and the domain
Get After At	Everything after the at symbol
Get domain	The domain before the dot
Before At	Everything before the at symbol
Domain freetext	
After symbol	Everything after a symbol
Between symbols	Everything between a pair of symbols
Delete punctuation	Remove punctuation
Delete spaces	Remove blanks
Digit to end	Everything after the first digit if exists
First character	Get first character
Get after comma	Everything after a comma
Get caps	Capitalise each word in a text
To upper	Convert text to upper case
Domain names	
Add title	The name with a title
Get title	The title attached to the name, if exists
Generate login	A login generated using the name
Reduce name	The name reduced before the surname(s)
Domain phones	
Add prefix by country	Phone numbers with the prefix of the countries
Delete parentheses	The list of phone numbers without parentheses
Get number	A phone number presented in the string, if exists
Set prefix	The list of phone numbers with the prefix
Set punctuation	A phone number split by a punctuation sign
Domain times	
Add time	The time increasing the hour by the integer
Append o’clock time	The time appending an o’clock time
Append time	The time appending the integer as new component
Convert time	The time formatted to 24 h format
Convert time	The time formatted to a given format
Convert time	The time formatted to 12 h format
Convert time	The time changed from the first time zone to the second
Delete time	The time deleting the last component
Get hour	The hour component
Get minutes	The minutes component

Table 2 continued

Task description	Expected output
Get time	A time presented in the string
Domain units	
Convert units	The value transformed to a different magnitude
Get system	The system represented by the magnitude
Get units	The units of the system
Get value	The numeric value without any magnitude

Appendix D

Here, we present the calculations with human errors.

As we said, humans were not always correct in the questionnaires when supplying or inspecting examples, and this varies per domain. We estimated e_s and e_i to account for the proportion of supplied examples and inspected examples respectively a human makes wrong.[8] The results are summarised in Table 4 and included in the estimations of costs. Namely, this estimate of the operating condition $\hat{\mathbf{c}}$ including error is given by

$$\hat{c}'_s = \frac{\chi_s}{\chi_w} + e_s \qquad \hat{c}'_i = \frac{\chi_i}{\chi_w} + e_i$$

Table 5 shows the operating conditions (medians) for the seven domains including the errors. In our questionnaires, humans were only given one solved example (they were not given any general rule about how to do the transformation), so it is important that we compare these errors e_s and e_i, especially the first one, with the results for GPT-3 for one-shot, as shown in Fig. 4. There we see that the domain with highest error is *times*, followed by *units*, and the domain with lowest error is *emails*. We see similar proportions in the human results in Table 5, not in magnitude (one-shot human errors are much lower than one-shot GPT-3) but in the domains that are more or less difficult. For humans, teh worst results are given for *times*, followed by *phones*. The best domain is *freetext* and *emails*.

Finally, Fig. 8 shows the overall expected costs, calculated in the same way as in Fig. 7, but now including the human errors. We see that in this extreme situation, the dynamic method obtains the best performance (not considering the optimal method since it is not realistic) in all the domains except from *times* and *units*. This is caused by c'_i being higher than c'_s in Table 5. If we analyse the difference between c'_s and c'_i and the performance of the dynamic method with respect to the static and fixed, we observe that the worst results of the dynamic with respect the other two is found in the domains where $c'_i > c'_s$: *emails*, *times* and *units*. The dynamic method was designed considering the expected situation that the cost of inspect examples is lower than the cost of supply examples for the same task. For the domains where this situation is not found, the static method seems to be a better option. Finally, the fixed method is not a good selection since it (almost) always obtains the worst results.

[8] Note that e_s simply accounts for the percentage of examples solved incorrectly, but e_i only accounts for the percentage of examples that are incorrect and not detected by the human (false negatives), but we do not consider here those examples that are correct and flagged as incorrect by the human (false positives), as they are covered by e_s. We do this, as this is the interpretation in Proposition 5.

Table 3 Examples of data wrangling tasks of different domains included in the repository used for the experimentation

Domain	**Tasks**	Example (*input* → *output*)
Dates	21	*74-03-31 → 31*
Email	10	*Jan.Kotas@litwareinc.com → litwareinc.com*
Freetext	25	*Association of Computational Linguistics → ACL*
Names	15	*Prof. Kathleen S. Fisher → Fisher, K.*
Phones	18	*John DOE 3 ...* [TS]865-000-0000 ... → *865-000-0000*
Times	24	*3:40 PM →* 15:40
Units	10	*12.20 dg → 1220.0 mg*

Table 4 Aggregate results from the questionnaires detailing the domain-specific operating conditions in the two last columns (mean and standard deviation) per domain

Domain	τ_s	τ_i	χ/τ	χ_w	e_s	e_i	$\hat{c}_s$	$\hat{c}_i$
Dates	6.72 ± 5.28	3.97 ± 6.44	68.29 ± 212.4	3554.63 ± 17989.17	0.1 ± 0.01	0.1 ± 0.05	0.161 ± 0.464	0.021 ± 0.048
Emails	7.72 ± 10.75	3.12 ± 2.77	61.94 ± 194.85	35.69 ± 179.02	0.01 ± 0	0.1 ± 0.09	0.073 ± 0.159	0.041 ± 0.107
Freetext	4.15 ± 2.59	3.47 ± 3.49	63.39 ± 194.61	3270.46 ± 17953.16	0.04 ± 0.02	0.02 ± 0	0.086 ± 0.21	0.104 ± 0.339
Names	6.14 ± 4.21	3.38 ± 2.76	54.26 ± 183.07	11.98 ± 45.11	0.09 ± 0.02	0.03 ± 0.02	0.116 ± 0.298	0.058 ± 0.146
Phones	11.74 ± 7.33	6.93 ± 6.34	67.87 ± 212.49	39.8 ± 179.09	0.16 ± 0.28	0.13 ± 0.17	0.093 ± 0.217	0.112 ± 0.397
Times	9.11 ± 4.87	5.81 ± 5.49	71.77 ± 222.78	3283.92 ± 17950.82	0.1 ± 0.02	0.22 ± 0.17	0.076 ± 0.244	0.035 ± 0.11
Units	7.06 ± 9.31	2.94 ± 3.97	48.84 ± 176.87	6466.64 ± 24969.11	0.04 ± 0	0.14 ± 0.04	0.03 ± 0.08	0.015 ± 0.035

Table 5 Median values obtained from the questionnaires. The two last columns show the operating conditions per domain including the one-shot error levels for humans (e_s, and e_i)

Domain	τ_s	τ_i	χ/τ	χ_w	e_s	e_i	$\hat{c}'_s$	$\hat{c}'_i$
Dates	6.637	1.892	14	4.000	0.103	0.099	0.109	0.100
Emails	5.381	2.635	15	2.000	0.006	0.099	0.020	0.103
Freetext	3.520	3.102	15	3.000	0.045	0.024	0.051	0.028
Names	6.270	2.431	12	1.000	0.088	0.027	0.100	0.033
Phones	9.850	5.463	15	3.000	0.161	0.126	0.174	0.132
Times	9.254	3.892	15	5.000	0.101	0.225	0.107	0.227
Units	2.640	2.120	15	5.000	0.043	0.140	0.045	0.142

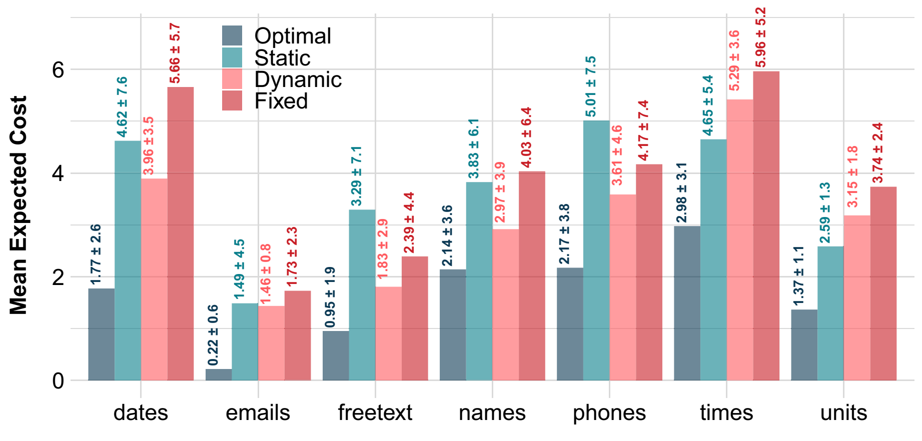


Fig. 8 Average expected costs per domain using the cost distributions from humans for the optimal (T^o), static (T^σ), dynamic (T^δ) and fixed (T^ϕ) methods, using the costs derived including the human errors from Table 5

Appendix E

In this section, we include tables and figures with the complete results that are shown summarised in the paper.

Table 6 shows the accuracy per task for increasing values of n_s. These values are aggregated to produce Fig. 4.

Table 7 includes the aggregated expected costs per domain, problem and method optimal (T^o), (static T^σ), dynamic (T^δ) and fixed (T^ϕ). In this case, these results are used to generate Fig. 7.

Figure 9 shows the results for active learning [15] following a straightforward random query strategy. We use a batch size of 1 (equal to the dynamic algorithm $s_0 = i_\oplus = 1$), and a fixed number of interactions (10, also equal to the dynamic algorithm, $s_\star = 10$).

As we can see, for this problem, active learning incurs very high costs for all the domains compared to the optimal (T^o), static (T^σ), dynamic (T^δ) and fixed (T^ϕ) methods, shown in Fig. 7. It could be argued that these high costs are due to the query strategy used. However, if we disaggregate the results by the type of cost incurred (c_s, c_i and c_w) as we show in Fig. 10), we see that an important part of the total cost Q is dominated by c_s and c_i. In particular, for the uniform costs, assuming active learning were perfect in finally reaching a model with $Q_w = 0$, we would get that $Q_s + Q_i$ around 24, which is always above the results of Fig. 7 (except for units with T^σ and T^ϕ). Similarly, for human costs, only T^σ could be worse for dates and names, even assume the active learning strategy is perfect and gets a perfect model with Q_w.

Note that the most important difference of active learning with our methods is that active learning does not balance the cost of supplying new examples against the cost of inspecting them by the user and, thus, in many operating conditions in the supply-inspect space, active learning cannot optimise for these different costs. Actually, active learning is dominated by $c_s \cdot n$, where c_s is almost never spared, as active learning looks for the most informative examples (usually requiring correction) where n is the number of iterations.

Finally, Fig. 11 illustrates the reject option behaviour for all domain problems and different values of n_s. The curves show the proportion of examples of each category (supplied $\frac{n_s}{n}$ in blue, accurate $\frac{n_a}{n}$ in green, wrong $\frac{n_w}{n}$ in red and rejected $\frac{n_r}{n}$ in grey) as we increase the reject threshold t_r in the x-axis. The eleven columns show the evolution for different values of n_s in $(0..10)$. As expected, for low values of the threshold few examples are rejected and when we increase the threshold, the number of rejected examples is increased. Usually, the wrong instances are the first to be rejected. This figure is also useful to study the performance of the LM depending on n_s. For 0-shot ($n_s = 0$), the performance is very low for all the domains. This performance increases significantly when we provide more examples to the LM. A summary of these plots is included in Fig. 4.

Table 6 Performance summary of accuracy scores across tasks for incremental supplied example sizes n_s

Domain	Task	n_s										
		0	1	2	3	4	5	6	7	8	9	10
Dates	addPunctuation-1	0	0.74	0.83	0.86	0.93	0.93	0.92	0.92	0.88	0.91	0.91
	addPunctuation-2	0	0.71	0.3	0.66	0.71	0.7	0.81	0.88	0.96	0.96	0.95
	changeFormat-1	0	0.16	0.3	0.83	0.86	0.81	0.73	0.84	0.83	0.87	0.77
	changeFormat-2	0.19	0.16	0.13	0.83	0.82	0.96	0.96	0.96	0.96	0.96	0.91
	changeFormat-3	0	0.32	0.33	0.41	0.57	0.63	0.65	0.88	0.83	0.78	0.91
	changeFormat-4	0.31	0.16	0.23	0.66	0.61	0.7	0.81	0.64	0.67	0.83	0.86
	changePunctuation-1	0.16	0.97	0.97	0.86	0.96	0.96	0.96	0.96	0.96	0.96	0.95
	changePunctuation-2	0.16	1	1	1	1	1	1	1	1	1	1
	getDay-1	0	0.35	0.4	0.48	0.46	0.59	0.77	1	1	1	1
	getDay-2	0	0.19	0.13	0.31	1	1	1	1	1	1	1
	getDay-3	0	0.94	0.97	1	1	1	1	1	1	1	1
	getDayOrdinal-1	0	0.16	0.17	0.24	0.32	0.48	0.69	0.88	1	1	1
	getDayOrdinal-2	0	0.16	0.13	0.48	0.79	0.81	1	1	1	1	1
	getMonthName-1	0	1	0.97	1	1	1	1	1	1	1	1
	getMonthName-2	0	0.84	1	1	1	1	1	1	1	1	1
	getWeekDay-1	0	0.84	1	0.83	0.93	0.96	1	1	1	1	1
	getWeekDay-2	0	0.97	0.97	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.95
	reduceMonthName-1	0	0	0.87	0.9	1	0.89	0.88	0.92	0.96	1	0.95
	reduceMonthName-2	0	0.61	0.57	0.45	0.54	0.3	0.5	0.4	0.42	0.65	0.64
	setFormat-1	0	0.29	0.23	0.31	0.46	0.74	0.81	1	0.96	0.87	0.95
	setFormat-2	0	0.13	0.13	0.31	0.46	0.63	0.81	1	1	1	1
Freetext	afterSymbol-1	0	0.87	1	0.97	1	1	1	1	1	1	1
	afterSymbol-2	0	0.84	0.97	1	1	1	1	1	0.96	0.96	0.95
	betweenSymbols-1	0	1	1	1	1	1	1	1	1	1	1
	betweenSymbols-2	0	0.97	1	1	1	1	1	1	1	1	1
	brackets-1	0.03	0.58	0.8	0.69	1	1	1	1	1	1	1
	brackets-2	0.19	0.87	0.63	0.59	1	1	1	1	1	1	1
	deletePunctuation-1	0	0.68	0.63	0.83	0.89	0.89	1	1	1	1	1
	deletePunctuation-2	0	0.29	0.47	0.48	0.36	0.63	0.96	0.84	0.88	0.83	0.91
	deletePunctuation-3	0	0.35	0.43	0.52	0.5	0.52	0.46	0.52	0.46	0.35	0.59
	deletePunctuation-4	0	0.1	0.3	0.38	0.39	0.37	0.46	0.52	0.58	0.48	0.59
	deleteSpaces-1	0	0.52	0.83	0.79	0.79	0.81	0.85	0.88	0.83	0.91	0.95
	deleteSpaces-2	0	0.26	0.43	0.45	0.46	0.44	0.54	0.68	0.67	0.7	0.73
	digitToEnd-1	0	0.97	0.97	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.95
	digitToEnd-2	0	1	1	1	1	1	1	1	1	1	1
	firstCharacter-1	0	0.45	0.93	1	1	1	1	1	1	1	1
	firstCharacter-2	0	0.81	0.77	0.97	0.93	0.93	0.96	0.96	0.96	0.96	0.95
	getAfterComma-1	0	0.97	0.97	0.97	0.96	1	1	0.96	1	1	1
	getAfterComma-2	0	1	1	1	1	1	1	1	1	1	1
	getBetweenCommas-1	0	0.9	1	0.83	1	1	1	1	1	1	1
	getBetweenCommas-2	0	0.16	0.33	0.31	0.54	0.81	0.96	0.8	0.92	0.96	0.95
	getCaps-1	0	0.42	0.87	0.9	0.93	0.93	0.96	0.92	0.96	1	0.95
	getCaps-2	0	0.74	0.9	0.93	0.93	0.93	0.92	0.92	0.92	0.91	1
	getCaps-3	0	0.71	0.93	0.93	0.89	0.93	0.88	0.92	0.92	0.87	0.86
	toUpper-1	0	0.42	0.63	0.69	0.64	0.67	0.69	0.68	0.67	0.74	0.77
	toUpper-2	0	0.39	0.7	0.69	0.71	0.78	0.81	0.68	0.75	0.78	0.82

Table 6 continued

Domain	Task	n_s										
		0	1	2	3	4	5	6	7	8	9	10
Names	addTitle-1	0	0.52	0.47	0.48	0.5	0.67	0.65	0.8	0.71	0.7	0.73
	addTitle-2	0	0.1	0.13	0.28	0.32	0.3	0.42	0.4	0.38	0.57	0.68
	getTitle-1	0	0.35	0.27	0.69	0.79	0.96	0.73	0.8	0.83	0.87	0.95
	getTitle-2	0	0.58	0.3	0.97	0.96	1	1	1	1	1	1
	login-1	0	0	0	0.03	0.04	0.04	0.04	0.04	0	0.04	0
	login-2	0	0	0	0.07	0.11	0.07	0.12	0.12	0.25	0.22	0.14
	reduceName-1	0	0.9	1	0.97	0.89	0.96	0.96	1	1	1	1
	reduceName-2	0	0.74	0.87	0.93	0.96	0.96	0.88	0.96	1	1	1
	reduceName-3	0	0.35	0.87	1	1	1	1	1	1	1	1
	reduceName-4	0	0.65	0.9	0.93	0.96	0.96	0.96	0.96	0.96	0.96	0.91
	reduceName-5	0	0.48	0.93	0.97	0.96	0.93	0.96	0.96	0.96	0.96	0.95
	reduceName-6	0	0.29	0.8	0.83	0.86	1	1	1	1	1	1
	reduceName-7	0	0.23	0.7	0.9	0.96	1	1	1	1	1	1
	reduceName-8	0	0.16	0.43	0.76	0.89	1	1	1	1	1	1
	reduceName-9	0	0.29	0.67	0.83	0.89	0.96	1	1	1	1	1
Emails	generate-1	0	1	1	1	1	1	1	1	1	1	1
	generate-2	0.19	0.94	1	1	1	1	1	1	1	1	1
	generate-3	0.44	0.74	0.83	0.83	0.82	0.81	1	1	1	1	1
	getAfterAt-1	0	1	1	1	1	1	1	1	1	1	1
	getAfterAt-2	0	0.9	0.83	0.83	0.82	0.81	1	1	1	1	1
	getAfterAt-3	0	1	1	1	1	1	1	1	1	1	1
	getAfterAt-4	0	1	1	1	1	1	1	1	1	1	1
	getDomain-1	0	0.97	1	1	1	1	1	1	1	1	1
	getDomain-2	0	1	1	1	1	1	1	1	1	1	1
	someBeforeAt-NA	0	0.13	0.7	0.79	0.86	0.81	0.96	0.96	0.96	1	1
Phones	countryPrefix-1	0	0.84	0.9	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.95
	countryPrefix-2	0	0.9	0.9	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.95
	countryPrefix-3	0	0	0	0.03	0	0	0	0	0	0.04	0.05
	countryPrefix-7	0	0.03	0	0	0	0.04	0.04	0	0	0.04	0.05
	countryPrefix-8	0	0.87	0.97	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.95
	countryPrefix-9	0	0.87	0.97	0.97	0.96	0.96	0.96	0.96	0.96	0.96	0.95
	deleteParentheses-1	0	1	0.83	1	1	1	1	1	1	1	1
	deleteParentheses-2	0	1	1	1	1	1	1	1	1	1	1
	getNumber-1	0	0.94	0.83	0.83	1	1	1	1	1	1	1
	getNumber-2	0	0.35	0.5	0.31	0.46	0.93	1	1	1	1	1
	setPrefix-1	0	0.35	0.33	0.31	0.46	0.63	0.81	0.96	0.96	0.96	0.95
	setPrefix-2	0	0.84	0.97	1	1	1	1	1	1	1	1
	setPrefix-3	0	1	1	1	1	1	1	1	1	1	1
	setPrefix-4	0	0.94	0.6	1	1	1	1	1	1	1	1
	setPrefix-5	0	0.35	0.33	0.31	0.46	0.48	0.73	1	1	1	1
	setPrefix-6	0	0.45	0.67	1	1	1	1	1	1	1	1
	setPunctuation-1	0	0.81	0.7	1	1	1	1	1	1	1	1
	setPunctuation-2	0	0.84	0.83	1	1	1	1	1	1	1	1

Table 6 continued

Domain	Task	n_s										
		0	1	2	3	4	5	6	7	8	9	10
Times	addTime-1	0	0.19	0.37	0.59	0.68	0.89	0.96	0.92	0.88	0.91	0.91
	addTime-2	0	0.48	0.63	0.69	0.93	0.96	0.96	1	1	0.87	0.73
	appendTime-1	0	0.52	0.5	0.48	0.82	0.81	0.81	0.96	1	1	1
	appendTime-2	0	0.81	0.8	0.9	0.82	0.96	0.92	0.88	0.92	0.96	1
	appendTime-3	0	0.84	0.83	0.83	0.82	0.81	0.81	0.92	0.83	1	0.95
	appendTime-4	0	0.81	0.8	0.9	0.82	0.85	0.81	0.84	0.83	0.87	0.95
	convert-1	0.5	0.1	0.27	0.62	0.79	0.74	0.92	0.88	0.92	0.87	0.91
	convert-10	0	0.03	0.1	0.24	0.14	0.22	0.31	0.2	0.33	0.43	0.41
	convert-2	0.62	0.06	0.07	0.59	0.68	0.78	0.77	0.72	0.71	0.74	0.86
	convert-3	0	0.1	0.2	0.55	0.46	0.56	0.73	0.68	0.79	0.74	0.77
	convert-4	0	0.06	0.07	0.17	0.29	0.33	0.5	0.6	0.58	0.7	0.68
	convert-5	0	0.26	0.13	0.66	0.68	0.67	0.65	0.72	0.83	0.83	0.95
	convert-6	0	0.06	0.23	0.34	0.32	0.56	0.5	0.52	0.75	0.7	0.68
	convert-7	0	0.1	0.07	0.59	0.64	0.63	0.62	0.56	0.54	0.57	0.59
	convert-8	0	0.03	0.1	0.17	0.21	0.22	0.31	0.24	0.33	0.39	0.36
	convert-9	0	0.06	0.07	0.21	0.54	0.59	0.62	0.6	0.5	0.61	0.59
	deleteTime-1	0	0.32	0.3	0.59	0.75	0.96	0.88	1	1	1	1
	deleteTime-2	0	0.35	0.33	0.31	0.93	1	1	1	1	1	1
	getHour-1	0	0.58	0.87	0.86	1	1	1	1	1	1	1
	getHour-2	0	0.03	0.07	0.07	0.54	0.63	0.62	0.6	0.58	0.57	0.55
	getMinutes-1	0	0.68	0.87	0.66	0.75	1	1	1	1	1	1
	getMinutes-2	0	0.61	0.3	0.93	0.79	0.85	1	1	1	1	0.95
	getTime-1	0.5	0.94	1	0.83	1	1	1	1	1	1	1
	getTime-2	0.59	0.9	0.93	0.9	0.75	0.93	0.92	0.88	0.83	0.87	0.91
Units	convert-1	0	0.16	0.13	0.45	0.39	0.48	0.62	0.88	0.92	0.83	0.86
	convert-2	0	0.19	0.2	0.41	0.25	0.44	0.58	0.68	0.83	0.87	0.82
	convert-3	0	0.13	0.13	0.24	0.36	0.44	0.58	0.76	0.88	0.78	0.73
	convert-4	0	0.03	0.17	0.48	0.21	0.33	0.62	0.6	0.71	0.78	0.77
	getSystem-1	0	0.1	0.33	0.66	0.61	0.67	0.58	0.96	0.83	0.87	1
	getSystem-2	0	0.16	0.43	0.72	0.71	0.85	0.77	0.84	0.88	1	1
	getUnits-1	0	0.97	0.9	0.9	0.89	0.89	0.88	0.88	0.88	0.87	0.86
	getUnits-2	0	0.87	0.7	0.28	0.79	0.96	0.96	0.96	0.96	0.96	0.95
	getValue-1	0	0.71	0.93	0.93	0.96	0.93	0.92	0.92	0.92	0.91	0.91
	getValue-2	0	0.61	0.97	0.93	0.96	0.96	0.92	0.96	0.96	0.96	0.95
Average:		0.03	0.54	0.62	0.71	0.77	0.81	0.84	0.87	0.87	0.88	0.89

Table 7 Aggregated expected costs per domain, problem and method (T^o, T^σ, T^δ and T^ϕ)

Domain	Task	T^o	T^σ	T^δ	T^ϕ
	addPunctuation-1	5.48	18.34	11.77	7.33
	addPunctuation-2	6.77	18.23	13.77	22.33
	changeFormat-1	7.58	20.49	11.26	13.59
	changeFormat-2	5.88	16.19	10.71	6.33
	changeFormat-3	9.46	19.41	19.43	16.45
	changeFormat-4	9.95	14.73	14.43	19.86
	changePunctuation-1	2.04	15.11	7.11	6.33
	changePunctuation-2	1.07	14.62	5	5.33
	getDay-1	6.82	18.45	19.2	19.65
	getDay-2	4.27	18.39	7.97	7.46
	getDay-3	2.33	17.25	7.12	5.33
	getDayOrdinal-1	7.85	18.86	25.62	24.85
	getDayOrdinal-2	6.11	18.45	12.53	10.33
	getMonthName-1	1.07	17.14	5.73	5.33
	getMonthName-2	2.07	17.38	5.9	5.33
	getWeekDay-1	2.07	17.42	7.61	7.4
	getWeekDay-2	2.04	17.62	5.17	6.33
	reduceMonthName-1	3.86	19.17	11.63	9.39
	reduceMonthName-2	11.62	21.15	24.69	25.44
	setFormat-1	6.87	18.85	18.79	19.06
Dates	setFormat-2	7	18.71	14.66	20.99
	afterSymbol-1	2.04	17.34	6.05	5.33
	afterSymbol-2	2.67	17.77	5.56	9.6
	betweenSymbols-1	1.07	17.12	5.04	5.33
	betweenSymbols-2	1.7	17.17	3.99	5.33
	brackets-1	4.02	17.28	6.92	5.33
	brackets-2	3.27	14.34	7.34	5.33
	deletePunctuation-1	5.31	17.65	10.33	18.13
	deletePunctuation-2	6.87	18.99	19.76	16.45
	deletePunctuation-3	14.75	21.85	20.45	40.45
	deletePunctuation-4	16.14	22.26	22.23	47.99
	deleteSpaces-1	6.17	18.58	14.17	16
	deleteSpaces-2	13.36	21.19	20.53	44.79
	digitToEnd-1	2.05	17.61	4.31	6.33
	digitToEnd-2	1.07	17.12	3.9	5.33
	firstCharacter-1	2.99	17.98	6.33	5.33
	firstCharacter-2	3.87	17.85	9.46	7.33
	getAfterComma-1	1.92	17.19	6.3	5.33
	getAfterComma-2	1.07	17.15	5.05	5.33
	getBetweenCommas-1	2.02	17.29	7.72	5.33
	getBetweenCommas-2	7.05	18.92	14.63	16.99
	getCaps-1	4.98	18.38	9.8	9.53
	getCaps-2	4.36	17.78	9.79	9.6
	getCaps-3	3.97	18.76	8.54	7.33
	toUpper-1	10.46	20.1	16.1	27.66

Table 7 continued

Domain	Task	T^{o}	T^{σ}	T^{δ}	T^{ϕ}
Freetext	toUpper-2	9.26	19.82	14.54	23.39
	addTitle-1	10.66	20.41	18.6	18.65
	addTitle-2	15.61	21.84	25.36	38.85
	getTitle-1	6.01	18.87	12.15	16
	getTitle-2	3.89	17.77	7.02	5.33
	login-1	26.13	26.51	31.2	58.45
	login-2	23.27	25.6	29.79	59.65
	reduceName-1	2.02	17.35	8.7	6.33
	reduceName-2	4.39	17.7	10.69	6.33
	reduceName-3	3.11	18.1	7.94	5.33
	reduceName-4	4.38	18.45	10.3	6.33
	reduceName-5	3.75	18.37	8.41	7.33
	reduceName-6	4.72	18.24	9.72	5.33
	reduceName-7	4.66	18.3	10.59	5.33
	reduceName-8	5.13	18.48	11.55	7.46
Names	reduceName-9	5.49	18.2	11.02	9.6
	generate-1	1.07	17.11	4.6	5.33
	generate-2	1.93	14.24	6.14	5.33
	generate-3	4.86	10.52	10.79	10.33
	getAfterAt-1	1.07	17.16	4.37	5.33
	getAfterAt-2	3.29	17.3	9.13	10.33
	getAfterAt-3	1.07	17.13	3.29	5.33
	getAfterAt-4	1.07	17.13	3.27	5.33
	getDomain-1	1.7	17.19	5.38	5.33
	getDomain-2	1.07	17.12	3.79	5.33
Emails	someBeforeAt-NA	6.3	18.56	13.67	19.06
	countryPrefix-1	3.75	17.77	7.75	6.33
	countryPrefix-2	3.28	17.7	5.91	7.4
	countryPrefix-3	25.88	26.2	30.78	62.91
	countryPrefix-7	25.61	26.11	31.56	61.85
	countryPrefix-8	3	17.75	5.55	6.33
	countryPrefix-9	2.99	17.74	5.71	6.33
	deleteParentheses-1	1.07	17.12	4.33	5.33
	deleteParentheses-2	1.07	17.11	3.66	5.33
	getNumber-1	2.5	17.23	10.2	5.33
	getNumber-2	5.89	18.12	15.58	9.6
	setPrefix-1	7.77	18.82	17.83	25.25
	setPrefix-2	2.67	17.37	6.92	5.33
	setPrefix-3	1.07	17.11	3.58	5.33
	setPrefix-4	2.33	17.22	5.41	5.33
	setPrefix-5	6.82	18.52	21.78	24.98
	setPrefix-6	3.2	17.95	6.16	5.33
	setPunctuation-1	2.97	17.41	5.45	9.6
Phones	setPunctuation-2	2.89	17.37	6.08	5.33
	addTime-1	6.97	19.07	14.38	14.79

Table 7 continued

Domain	Task	T^o	T^σ	T^δ	T^ϕ
	addTime-2	5.46	20.08	10.62	12.8
	appendTime-1	6.78	18.18	13.59	10.33
	appendTime-2	4.77	17.58	11.23	6.33
	appendTime-3	4.71	17.98	12.18	10.33
	appendTime-4	5.25	18.05	11.91	9.33
	convert-1	11.83	30.11	22.18	54.25
	convert-10	19.64	23.72	25.81	46.58
	convert-2	16.29	29.11	25.87	50.85
	convert-3	15.85	40.45	27.54	64.84
	convert-4	20.43	41.51	39.18	80.57
	convert-5	9.16	19.19	16.14	14.32
	convert-6	13.07	21.48	22.2	39.39
	convert-7	13.12	21.81	16.94	24.12
	convert-8	20.63	24.02	27.19	47.72
	convert-9	14.5	21.81	25.99	21.79
	deleteTime-1	5.84	18.33	14.89	7.46
	deleteTime-2	5.02	18.14	12.24	5.33
	getHour-1	3.86	17.79	9.61	5.33
	getHour-2	14.07	22.46	27.17	22.12
	getMinutes-1	4.3	17.68	11.09	5.33
	getMinutes-2	4.52	18.25	11.04	22.39
	getTime-1	1.93	9.22	8.21	5.33
Times	getTime-2	3.64	8.65	10.46	7.33
	convert-1	11.83	30.11	22.18	54.25
	convert-2	16.29	29.11	25.87	50.85
	convert-3	15.85	40.45	27.54	64.84
	convert-4	20.43	41.51	39.18	80.57
	getSystem-1	7.58	19.12	15.14	23.12
	getSystem-2	7.45	18.88	12.83	14.93
	getUnits-1	2.04	18.34	8.91	8.33
	getUnits-2	4.11	17.75	11.47	6.33
	getValue-1	3.85	18.4	9.06	7.33
Units	getValue-2	3.11	18.21	10.38	6.33
Average:		6.67	19.45	13.02	17.25

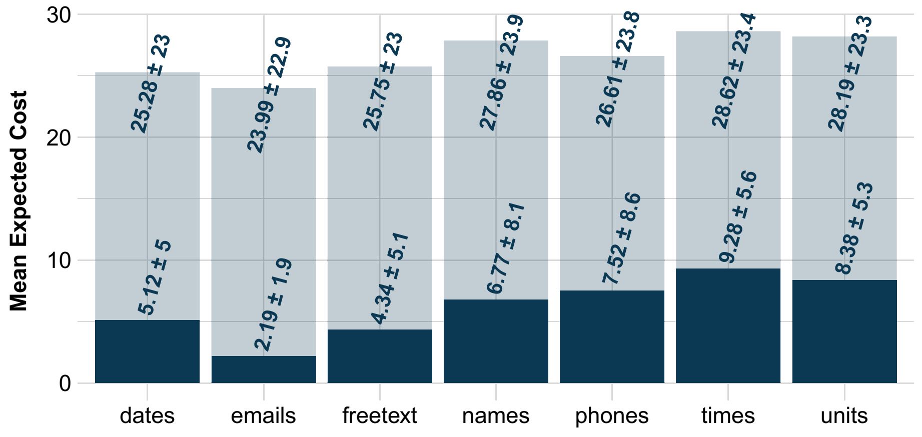


Fig. 9 Average expected costs per domain using the cost distributions from humans (using the costs derived including the human errors from Table 5) and with uniform $\mathcal{H}$ using an active learning approach

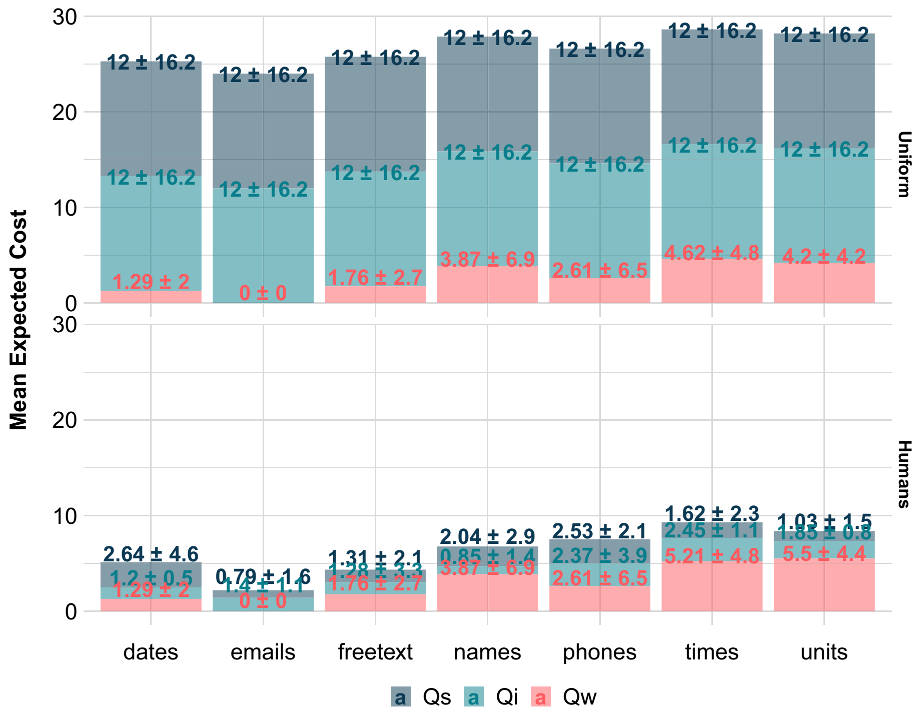


Fig. 10 Average expected costs per domain for the active learning. Top: uniform costs $\mathcal{H}$. Bottom: cost distributions from humans (using the costs derived including the human errors from Table 5). This is a disaggregation of Fig. 9. Compare with Fig. 7

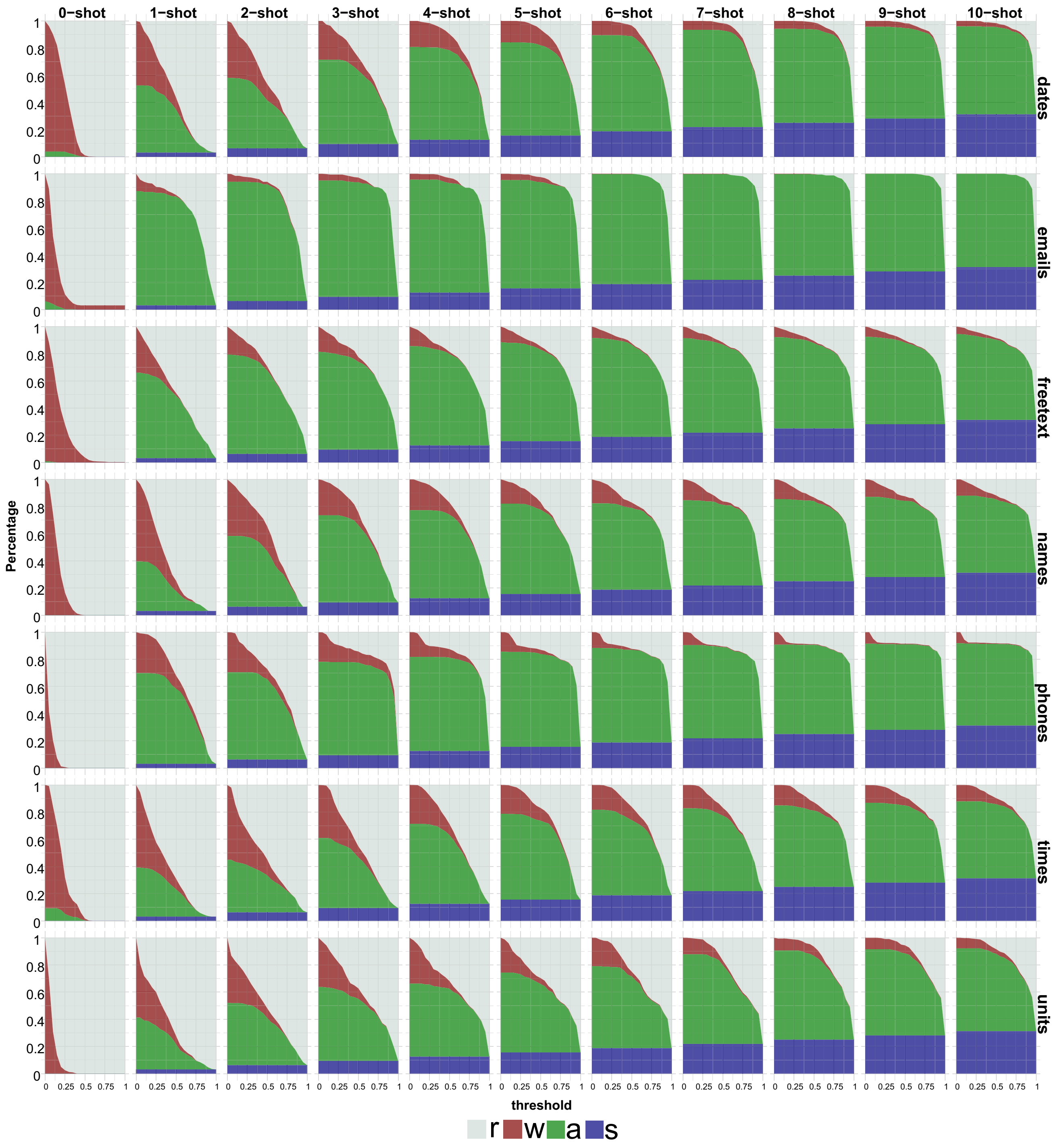


Fig. 11 Reject option behaviour for all domain problems. The curves show the proportion of examples of each category (supplied $\frac{n_s}{n}$ in blue, accurate $\frac{n_a}{n}$ in green, wrong $\frac{n_w}{n}$ in red and rejected $\frac{n_r}{n}$ in grey) as we increase the reject threshold t_r in the x-axis. The eleven plots show the evolution for different values of n_s in (0..10)

Appendix F

Here we include details about the questionnaires, the way they were distributed and how the estimates for c_s and c_i were obtained, as well as the aggregated distributions.

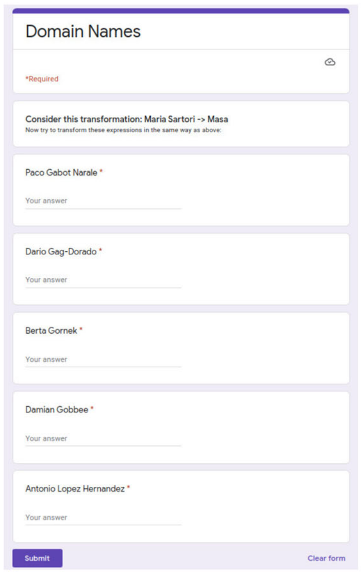
The questionnaires employed in this work are based on the Data Wrangling Dataset Repository,[9] a benchmark integrating many data wrangling tasks in the literature as well as new manually gathered tasks dealing with daily transformations [24]. Overall, the repository contains 123 different tasks divided into 7 different domains (*dates*, *emails*, *freetext*, *names*, *phones*, *times* and *units*). For every task we have 32 annotated examples where an input string is converted into a corrected or transformed version.

For each domain we selected randomly one task for 'supply' examples, and a different task for 'inspect' responses. For each task, we also selected randomly five instances to be completed by the humans who were performing the questionnaire. These first two questionnaires measured the actual time for solving an instance (its supply time τ_s being a proxy for c_s) and the actual time for verifying an instance (its inspect time τ_i being a proxy for c_i), averaged over five instances per question. An example of the questionnaire for the domain *names* and can be seen in Fig. 12a (supply) and in Fig. 12b (inspect).

This was followed by a third subjective question asking the cost unit per time unit of a person (χ/τ), so that times could be converted into costs, and a fourth subjective question that asked about the cost of each error χ_w directly. We just derived $\chi_s = \tau_s \frac{\chi}{\tau}$ and $\chi_i = \tau_i \frac{\chi}{\tau}$. Finally, we divided both by χ_w to have the normalised costs in $\mathbf{c}$. That is, the estimate of the operating condition $\hat{\mathbf{c}}$ is given by $\hat{c}_s = \frac{\chi_s}{\chi_w}$ and $\hat{c}_i = \frac{\chi_i}{\chi_w}$. Before these questions, we used an introductory text to give some context. In Fig. 12c, we show the questionnaire employed in the domain *names*.

In order to improve the estimation of costs and reduce the effect of respondent fatigue [60] (i.e., poor performance and efficiency for the later items of a questionnaire when respondents get bored, tired, or uninterested with the task), we produced a second version of the questionnaires in which we used the same tasks but swapping problems. The task used for the the supply problem in the first questionnaire was used for the inspect problem in the second version. Likewise, the task used for the inspect problem in the first questionnaire was used in the supply problem in the second version. Additionally, we reversed the order of the domains in the second questionnaire. The first version of the questionnaire was filled by 17 respondents, while the second version was filled by 14, 31 in total.

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



Domain Names

*Required

Consider this transformation: Maria Sartori -> Masa
Now try to transform these expressions in the same way as above:

Paco Gabot Narale *
Your answer

Dario Gag-Dorado *
Your answer

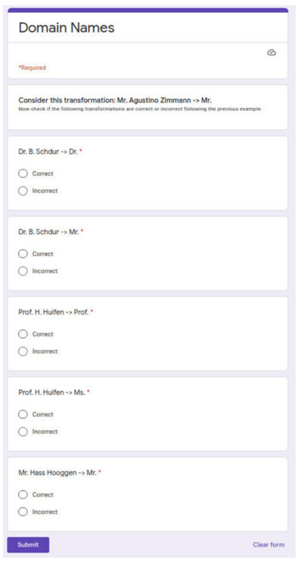
Berta Gornek *
Your answer

Damian Gobbee *
Your answer

Antonio Lopez Hernandez *
Your answer

Submit | Clear form

(a) Questionnaire for supply instances of the *names* domain used to estimate τ_s



Domain Names

*Required

Consider this transformation: Mr. Agustino Zimmann -> Mr.
Now check if the following transformations are correct or incorrect following the previous example.

Dr. B. Schdur -> Dr. *
○ Correct
○ Incorrect

Dr. B. Schdur -> Mr. *
○ Correct
○ Incorrect

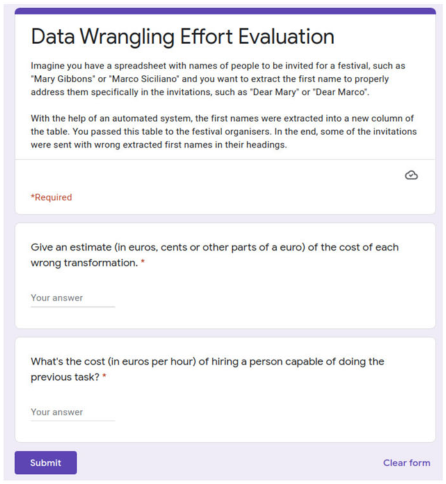
Prof. H. Huifen -> Prof. *
○ Correct
○ Incorrect

Prof. H. Huifen -> Ms. *
○ Correct
○ Incorrect

Mr. Hass Hooggen -> Mr. *
○ Correct
○ Incorrect

Submit | Clear form

(b) Questionnaire for inspect instances of the *names* domain used to estimate τ_i



Data Wrangling Effort Evaluation

Imagine you have a spreadsheet with names of people to be invited for a festival, such as "Mary Gibbons" or "Marco Siciliano" and you want to extract the first name to properly address them specifically in the invitations, such as "Dear Mary" or "Dear Marco".

With the help of an automated system, the first names were extracted into a new column of the table. You passed this table to the festival organisers. In the end, some of the invitations were sent with wrong extracted first names in their headings.

*Required

Give an estimate (in euros, cents or other parts of a euro) of the cost of each wrong transformation. *
Your answer

What's the cost (in euros per hour) of hiring a person capable of doing the previous task? *
Your answer

Submit | Clear form

(c) Questionnaire of the *names* domain used to estimate cost unit per time unit of a person (χ/τ), and the cost of each error χ_w

Fig. 12 Questionaries used for the domain "names"

Acknowledgements We thank the anonymous reviewers for their comments.

Funding The funding has been received from ValgrAI - Valencian Graduate School and Research Network for Artificial Intelligence; the Norwegian Research Council with Grant no. 329745 (Machine Teaching for Explainable AI); Generalitat Valenciana with Grant nos. CIPROM/2022/6 (FASSLOW) and IDIFEDER/2021/05 (CLUSTERIA); the European Commission under H2020-EU with Grant no. 952215 (TAILOR); US DARPA with Grant no. HR00112120007 (RECoG-AI); the Future of Life Institute with Grant no. RFP2-152; the Spanish Ministry of Science and Innovation (MCIN/AEI/10.13039/501100011033) with Grant no. PID2021-122830OB-C42 (SFERA) and "ERDF A way of making Europe"; and the Spanish Ministry of Universities with Grant no. PID2022-140110OA-I00 (FISCALTICS) funded by MICIU/AEI/10.13039/501100011033 and by ERDF, EU.

Data availability All the code and results, questionnaires and responses can be found in https://github.com/nandomp/Trade-OffsFew-Shot.git.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no Conflict of interest.

References

1. Wang Y, Yao Q, Kwok JT, Ni LM (2020) Generalizing from a few examples: a survey on few-shot learning. ACM Comput Surv (CSUR) 53(3):1–34
2. Tao H, Cheng L, Qiu J, Stojanovic V (2022) Few shot cross equipment fault diagnosis method based on parameter optimization and feature mertic. Measur Sci Technol 33(11):115005
3. Wang K, Liew JH, Zou Y, Zhou D, Feng J (2019) Panet: Few-shot image semantic segmentation with prototype alignment. In: proceedings of the IEEE/CVF international conference on computer vision. p. 9197–9206
4. Yang B, Liu C, Li B, Jiao J, Ye Q (2020) Prototype mixture models for few-shot semantic segmentation. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16. Springer. p. 763–778
5. Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, et al. (2020) Language Models are Few-Shot Learners. In: Advances in Neural Information Processing Systems. p. 1877–1901
6. OpenAI. GPT-4 technical report. ArXiv. 2023;abs/2303.08774
7. Zeng W, Ren X, Su T, Wang H, Liao Y, Wang Z, et al (2021) PanGu-α: large-scale autoregressive pretrained chinese language models with auto-parallel computation. arXiv preprint arXiv:2104.12369
8. Chowdhery A, et al (2022) PaLM: scaling language modeling with pathways. arXiv:2204.02311 [cs]
9. BigScience, et al (2023) BLOOM: A 176B-parameter open-access multilingual language model. https://doi.org/10.48550/arXiv.2211.05100. arXiv:2211.05100 [cs]
10. Touvron H, Lavril T, Izacard G, Martinet X, Lachaux MA, Lacroix T, et al (2023) Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971
11. Schellaert W, Martínez-Plumed F, Vold K, Burden J, Casares PA, Loe BS et al (2023) Your prompt is my command: on assessing the human-centred generality of multimodal models. J Artif Intell Res 77:377–394
12. Franc V, Prusa D, Voracek V (2023) Optimal strategies for reject option classifiers. J Mach Learn Res 24(11):1–49
13. Pugnana A, Ruggieri S (2023) A model-agnostic heuristics for selective classification. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37. p. 9461–9469
14. Hendrickx K, Perini L, Van der Plas D, Meert W, Davis J (2023) Machine learning with a reject option: a survey. arXiv preprint arXiv:2107.11277
15. Kumar P, Gupta A (2020) Active learning query strategies for classification, regression, and clustering: a survey. J Comput Sci Technol 35:913–945
16. Rattenbury T, Hellerstein JM, Heer J, Kandel S, Carreras C (2017) Principles of data wrangling: practical techniques for data preparation. O'Reilly Media, Inc
17. Jaimovitch-López G, Ferri C, Hernández-Orallo J, Martínez-Plumed F, Ramírez-Quintana MJ (2023) Can language models automate data wrangling? Mach Learn 112(6):2053–2082
18. Charoenphakdee N, Cui Z, Zhang Y, Sugiyama M (2021) Classification with rejection based on cost-sensitive classification. In: International Conference on Machine Learning. PMLR. p. 1507–1517
19. Zhou L, Martínez-Plumed F, Hernández-Orallo J, Ferri C, Schellaert W (2022) Reject before you run: small assessors anticipate big language models. In: 1st AI Evaluation Beyond Metrics Workshop (EBEM), CEUR Proceedings, volume 3169
20. Lu Y, Bartolo M, Moore A, Riedel S, Stenetorp P (2021) Fantastically ordered prompts and where to find them: overcoming few-shot prompt order sensitivity. arXiv preprint arXiv:2104.08786
21. Flach PA (2016) ROC analysis. In: Encyclopedia of Machine Learning and Data Mining. Springer, p. 1–8
22. Nakas C, Bantis L, Gatsonis C (2023) ROC analysis for classification and prediction in practice. CRC Press
23. Tian Y, Si L, Zhang X, Cheng R, He C, Tan KC et al (2021) Evolutionary large-scale multi-objective optimization: a survey. ACM Comput Surv (CSUR) 54(8):1–34
24. Contreras-Ochando L, Ferri C, Hernández-Orallo J, Martínez-Plumed F, Ramírez-Quintana MJ, Katayama S (2019) Automated data transformation with inductive programming and dynamic background knowledge. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, ECML PKDD 2019. Springer, p. 735–751
25. Contreras-Ochando L, Ferri C, Hernández-Orallo J, Martínez-Plumed F, Ramírez-Quintana MJ, Katayama S (2019) BK-ADAPT: dynamic background knowledge for automating data transformation. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer. p. 755–759
26. Srivastava A, Rastogi A, Rao A, Shoeb AAM, Abid A, Fisch A, et al. (2022) Beyond the imitation game: In: Quantifying and extrapolating the capabilities of language models. arXiv preprint arXiv:2206.04615
27. Burnell R, Schellaert W, Burden J, Ullman TD, Martinez-Plumed F, Tenenbaum JB et al (2023) Rethink reporting of evaluation results in AI. Science 380(6641):136–138
28. Virani N, Iyer N, Yang Z (2020) Justification-based reliability in machine learning. In: Proc. of the AAAI Conf. on Artificial Intelligence. vol. 34. p. 6078–6085
29. Cabitza F, Campagner A, Balsano C (2020) Bridging the "last mile" gap between AI implementation and operation:"data awareness" that matters. Ann Transl Med 8(7). https://doi.org/10.21037/atm.2020.03.63
30. De A, Koley P, Ganguly N, Gomez-Rodriguez M (2020) Regression under human assistance. In: Proc. of the AAAI Conf. on Artificial Intelligence. vol. 34. p. 2611–2620
31. Pan Z, Yu W, Yi X, Khan A, Yuan F, Zheng Y (2019) Recent progress on generative adversarial networks (GANs): a survey. IEEE Access 7:36322–36333
32. Harshvardhan G, Gourisaria MK, Pandey M, Rautaray SS (2020) A comprehensive survey and analysis of generative models in machine learning. Comput Sci Rev 38:100285
33. Saxena D, Cao J (2021) Generative adversarial networks (GANs) challenges, solutions, and future directions. ACM Comput Surv (CSUR) 54(3):1–42
34. Sung F, Yang Y, Zhang L, Xiang T, Torr PH, Hospedales TM (2018) Learning to compare: relation network for few-shot learning. In: Proc. of the IEEE Conf. on computer vision and pattern recognition. p. 1199–1208
35. Xu S, Semnani S, Campagna G, Lam M (2020) AutoQA: from databases to Q&A semantic parsers with only synthetic training data. In: Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing (EMNLP). p. 422–434
36. Izacard G, Grave E (2020) Leveraging passage retrieval with generative models for open domain question answering. arXiv preprint arXiv:2007.01282

37. Hendrycks D, Burns C, Basart S, Zou A, Mazeika M, Song D, et al (2020) Measuring massive multitask language understanding. In: International Conf. on Learning Representations
38. Reynolds L, McDonell K (2021) Prompt programming for large language models: beyond the few-shot paradigm. arXiv preprint arXiv:2102.07350
39. Scao TL, Rush AM (2021) How many data points is a prompt worth? arXiv preprint arXiv:2103.08493
40. Bommasani R, Hudson DA, Adeli E, Altman R, Arora S, von Arx S, et al. (2021) On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258
41. Chow C (1970) On optimum recognition error and reject tradeoff. IEEE Trans Inform Theory 16(1):41–46
42. Herbei R, Wegkamp MH (2006) Classification with reject option. In: Canadian Journal of Statistics/La Revue Canadienne de Statistique. p. 709–721
43. Bartlett PL, Wegkamp MH (2008) Classification with a reject option using a hinge loss. J Mach Learn Res 9(59):1823–1840
44. Wegkamp M, Yuan M (2011) Support vector machines with a reject option. Bernoulli 17(4):1368–1385
45. Denis C, Hebiri M, Zaoui A (2020) Regression with reject option and application to kNN. arXiv preprint arXiv:2006.16597
46. Lee JK, Bu Y, Rajan D, Sattigeri P, Panda R, Das S, et al (2021) Fair selective classification via sufficiency. In: International Conf. on Machine Learning. PMLR. p. 6076–6086
47. Tortorella F (2000) 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. p. 611–620
48. Pietraszek T (2007) On the use of ROC analysis for the optimization of abstaining classifiers. Mach Learn 68(2):137–169
49. Settles B (2011) From theories to queries: active learning in practice. In: Active Learning and Experimental Design workshop In conjunction with AISTATS 2010. JMLR Workshop and Conf. Proc. p. 1–18
50. Chen X, Price E (2019) Active regression via linear-sample sparsification. In: Beygelzimer A, Hsu D, (eds) Proc. of the Thirty-Second Conf. on Learning Theory. vol. 99 of Proc. of Machine Learning Research. PMLR. p. 663–695
51. Margineantu DD (2005) Active cost-sensitive learning. In: Proc. of the 19th International Joint Conf. on Artificial Intelligence. p. 1622–1623
52. Settles B, Craven M, Friedland L (2008) Active learning with real annotation costs. In: Proc. of the NIPS workshop on cost-sensitive learning. vol. 1. Available at https://api.semanticscholar.org/CorpusID:16285026
53. Haertel RA, Seppi KD, Ringger EK, Carroll JL (2008) Return on investment for active learning. In: Proc. of the NIPS Workshop on cost-sensitive learning. vol. 72
54. Culotta A, McCallum A (2005) Reducing labeling effort for structured prediction tasks. In: Proc. of the AAAI Conf. on Artificial Intelligence,. vol. 5. p. 746–751
55. Fu Y, Zhu X, Li B (2013) A survey on instance selection for active learning. Knowl Inform Syst 35(2):249–283
56. El-Yaniv R, Wiener Y (2012) Active learning via perfect selective classification. J Mach Learn Res 13(2):255–279
57. Shah K, Manwani N (2020) Online active learning of reject option classifiers. In: Proc. of the AAAI Conf. on Artificial Intelligence. vol. 34. p. 5652–5659
58. Zhou C, Tao H, Chen Y, Stojanovic V, Paszke W (2022) Robust point-to-point iterative learning control for constrained systems: A minimum energy approach. Int J Robust Nonlinear Control 32(18):10139–10161
59. Hernández-Orallo J, Flach P, Ferri C (2012) A unified view of performance metrics: translating threshold choice into expected classification loss. J Mach Learn Res 13(Oct):2813–2869
60. Jeong D, Aggarwal S, Robinson J, Kumar N, Spearot A, Park DS (2023) Exhaustive or exhausting? Evidence on respondent fatigue in long surveys. J Dev Econ 161:102992