# Comparing Learning Systems with Machine Teaching

Cèsar Ferri

VRAIN, Universitat Politècnica de València
cferri@dsic.upv.es

*AI Initiative for Science*

Nov 25 - 26 , 2021
València, Spain

# Outline

# Introduction

## Priors and Patterns

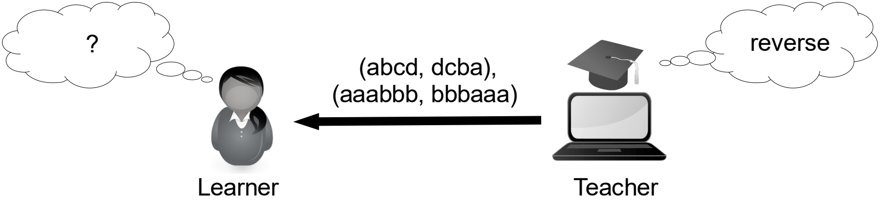
- An effective and reliable interaction between humans and machines depends on a proper understanding of their priors
  - **Extrinsic Pattern:** *twinkle twinkle little* $\longrightarrow$ *star*
  - **Intrinsic Pattern:** *on off on off on* $\longrightarrow$ *off*

## Language models

- Trained over massive natural language corpora that extrapolate sequences of symbols,
  - Very good at exploiting the **extrinsic patterns** borrowed from humans
  - Latest versions can also identify some **intrinsic patterns** in the prompts (few-shot learners)

## Questions:

- How elaborate can these abstractions be?
- Can they even have an algorithmic nature?
- What is the minimal set of examples for learning a concept?
- What priors these pretrained models incarnate for few-shot learning to work?




## Machine Teaching

- Area that looks for the optimal examples that a teacher should use to make a learner capture a concept
- Given a concept, find a *witness set* that allows the learner to uniquely identify the concept
- With rich representation languages similar to human language in expressive power:
  - Strong simplicity priors can make effective teaching possible with very few examples
  - The example set is usually shorter than the algorithmic expression of the concept itself

## Goals

- We use a machine teaching setting to **explore the number and length of the examples language models need to 'learn'** from prompts
- We **analyse whether a simplicity prior is acquired ('distilled')** as the result of training large language models
- We **compare them with other AI systems and humans** on the same task, which are known to have this prior

# Learning Systems

## Machine Learning

- The study of computer algorithms that can improve automatically through experience and by the use of data
- **Inductive inference**: inferring knowledge from data
- Many ML methods require vast amount of data:
  - Neural Networks
  - Deep Learning
- Since we want to compare systems with humans we can only employ methods that work with **few examples**

## Inductive Programming

- An inter-disciplinary domain of AI and cognitive science;
    - Studies the automatic synthesis of computer programs from examples and background knowledge (existing functions)
- Inductive inference makes generalisations from observations (evidence) by identifying general patterns in the data
- Usually IP is addressed from two different perspectives:
    - Logic paradigm: Inductive Logic Programming
    - Functional paradigm: Inductive Functional Programming
- Programming by example (PbE) is a sub-field where the specification comes in the form of input-output examples:
    - PbE can help to automate many data manipulation tasks, such as extracting terms from texts, transforming data format,etc

## Inductive Logic Programming

- ILP uses logic programming as a uniform representation for examples, background knowledge and hypotheses
- Positive examples ($E+$) + negative examples ($E-$)+ background knowledge ($B$) $\Rightarrow$ hypothesis ($h$).
- Given $E+, E-$ and $B$, an ILP system derives a hypothesised program $h$ which entails all the positive and none of the negative examples.
- Systems: Progol, Lousie

| | | | |
|---|---|---|---|
| Necessity: | $B$ | $\not\models$ | $E^+$ |
| Sufficiency: | $B \wedge h$ | $\models$ | $E^+$ |
| Weak consistency: | $B \wedge h$ | $\not\models$ | $false$ |
| Strong consistency: | $B \wedge h \wedge E^-$ | $\not\models$ | $false$ |

parent(a,b) parent(a,c) parent(d,b)

father(a,b) father(a,c) mother(d,b)

male(a) female(c) female(d)

$E^+$

Induction $h$

father(X,Y) :- parent(X,Y) & male(X)
mother(X,Y) :- parent(X,Y) & female(X)

## Inductive Functional Programming

- Based on the functional programming paradigm
  - Knowledge is expressed as functions
- Systems: Magichaskeller

$$
\begin{aligned}
typeTrain &= [Car] \\
typeCar &= (CShape, CLength, CRoof, CWheels, Load) \\
dataCShape &= Rect|Hexa|... \\
dataCLength &= Long|Short \\
dataCRoof &= None|Peak|... \\
typeCWheels &= Int \\
typeLoad &= (LShape, LNumber) \\
dataLShape &= Circ|Hexag|Recta|Tria|... \\
typeLNumber &= Int
\end{aligned}
$$

$$eastbound :: Train -> Bool$$

$$
\begin{aligned}
eastbound\ & ([(Rect, Short, None, 2, (Circ, 1)), \\
& (Rect, Long, None, 3, (Hexag, 1)), \\
& (Rect, Short, Peak, 2, (Tria, 1)), \\
& (Rect, Long, None, 2, (Recta, 3))]) = True
\end{aligned}
$$

## Language models

- Language models are conceptually simple systems
  - Estimate the probability $p(y|x)$ of a given sequence of characters $y$ following another sequence $x$
- These models are usually based on large deep learning architectures
  - Transformers (attention-based architectures)
- Systems: BERT, GPT-2, GPT-3

## Language models

- Trained over massive natural language corpora that extrapolate sequences of symbols,
  - Very good at exploiting the **extrinsic patterns** borrowed from humans
  - Latest versions can also identify some **intrinsic patterns** in the prompts (few-shot learners)
- Extensive experimental research is showing remarkable extrapolations from small prompts
  - Mathematical problem solving
  - Question answering
  - Language understanding
  - Code generation
  - Data wrangling

## Prompts

- Piece of text inserted in the input examples, so that the original task can be formulated as a (masked) language modeling problem

## Sentiment classifier

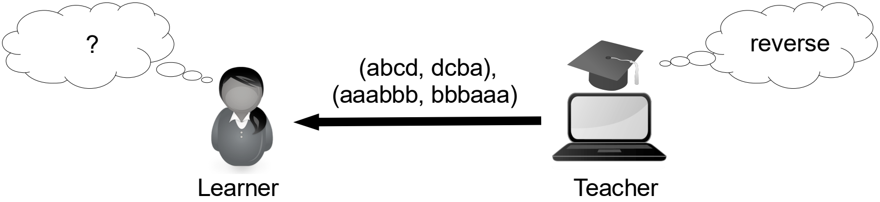
```
Tweet: "I loved the new Batman movie!"
Sentiment: Positive
###
Tweet: "I hate it when my phone battery dies."
Sentiment: Negative
###
Tweet: "This new music video blew my mind"
Sentiment:
```

## Airport code extractor

```
Text: "I want to fly form Los Angeles to Miami."
Airport codes: LAX, MIA
Text: "I want to fly from Orlando to Boston"
Airport codes:
```

## Movie to Emoji

# Machine Teaching




## Machine Teaching

- Area that looks for the optimal examples that a teacher should use to make a learner capture a concept
- Given a concept, find a *witness set* that allows the learner to uniquely identify the concept

## Machine Teaching

- Given a concept, find a set of examples, *witness set*, that allows the learner to uniquely identify the concept
- **Teaching dimension**: minimum cardinality of a witness set for the concept
  - This maximum will usually be unbounded
  - Even if we adopt a preference-based order on concepts, the maximum is still unbounded and incomputable

## Teaching Dimension

- Some concepts can be taught with very few examples, but those examples could be enormously large
  - Deviates from any intuitive notion of 'small teaching witness'
  - Ignores the time needed for the teacher to find the witness set and for the learner to identify the right concept
  - The teacher algorithm may not even terminate

## Goal

*Can we define a new teaching metric that is more reasonably related to how easy it is to teach a concept?*

# Theoretical Framework

## Machine Teaching

- We have an infinite example space $X$ and an infinite concept class $C$ consisting of concepts that are a subset of $X$
- For any concept $c \in C$ the teacher must find a small witness set $w \subseteq X$ of positive examples from which the learner is able to uniquely identify the concept
- An example set $S = \{\langle i_1, o_1 \rangle, ..., \langle i_k, o_k \rangle\}$ is just a finite set of binary i/o pairs, used as witness
- We say that a concept $c$ satisfies $S$, denoted by $c \models S$, if $c(i) = o$ for all the pairs $\langle i, o \rangle$ in S (all concepts satisfy the empty set)

## Teaching Size

The **teaching size (TS)** of a concept $c$ could is defined as follows:

$$TS(c) = \min_{S}\{\delta(S) : \{c\} = \{c' \in C : c' \models S\}\}$$

i.e., as the size of the smallest (using the $\delta$ encoding) example set $S$ that allows a learner to uniquely identify $c$

- This minimal set $S$ is known as a witness set for the concept $c$
- TS depends on both a concept and a learner

## Order of programs: $\prec$

- Let $\prec$ be a total order of programs ordered by length
- Programs with fewer instructions preceding longer programs, and lexicographically for programs having the same number of instructions

## $f$-compatibility

- A program $p$ is $f$-compatible with example set $S$, denoted $p \models_f S$, if for all pairs $\langle i, o \rangle \in S$ the program $p$ on input $i$ will within a certain time steps have the specified behavior

## $K^f$-Teaching Size

For a time complexity function $f$, we define the first program:

$$\Phi_\ell^f(S) = \underset{p}{\operatorname{argmin}}^{\prec}\{\ell(p) : p \models_f S\}$$

We then use this to define the $K^f$-**teaching size** of a concept $c$ as:

$$TS_\ell^f(c) = \min_S\{\delta(S) : \Phi_\ell^f(S) \in Class_L(c)\}$$

## Teaching Book: Teacher

- $f$-**Teaching Book**:
  - A list of *entries* in the form of program/witness pairs $(p, w)$, with $p$ the smallest program $f$-compatible with $w$, and $w$ the smallest witness for which $p$ is $f$-compatible
- The teacher fills the $f$-Teaching Book
  - Generate example sets in non-decreasing size
  - For each example set try programs of increasing lengths until finding the first one with the correct behavior

## Teaching Book: Learner

- Given an witness set $w$, the learner wants to find $\Phi_\ell^f(w)$, the smallest program $f$-compatible with $w$
- This program has some fixed length, and therefore there is a finite number of programs preceding it in the $\prec$ order

## Teaching Book: Computability

- The learner is computable
  - Try programs in increasing order, i.e. $\prec$ order, and test if the program is $f$-compatible with $w$, with the first such program being the answer
- The teacher is computable
  - By filling what an $f$-Teaching Book with $p$ the smallest program $f$-compatible with $w$, and $w$ the smallest witness for which $p$ is $f$-compatible

## Teaching Book: Algorithm

- A brute-force algorithm to fill the $f$-Teaching Book:
  - Try all witness sets $w$ with no contradictory pairs but allowing $\langle i, \perp \rangle$ pairs, in order of non-decreasing $\delta$ encoding size
  - Run the learner algorithm on $w$ to find a program $p$
  - Test if the program $p$ found by the learner on this $w$ is already in the $f$-Teaching Book
  - If it is not then add the pair $(p, w)$ to the $f$-Teaching Book

# Experimental Validation

- We describe an experimental validation of our method for teaching a universal language, P3, a simple language for string manipulation
- As learning prior we take the preference order $\prec$ built on program size and complexity function $f$
- We compute the teaching size $TS_\ell^f$ of P3-concepts, i.e. concepts computable by P3 programs
- As $\delta$ we use the Elias Delta Coding, which encodes a number $x$ using $\lfloor \log_2(x) \rfloor + 2\lfloor \log_2(\lfloor \log_2(x) \rfloor + 1) \rfloor + 1$ bits

## P3

- We employ the universal language P3 with 7 instructions:
  - `<>+-[]o`
  - The tape alphabet has 3 symbols $\Sigma = \{0, 1, .\}$
- Operations on an input/memory tape of cells by moving a pointer left ($<$) and right ($>$)
- Two instructions overwrite the symbol in the pointer cell (+ and −)
- Instruction o outputs the content of the tape
- The bracket instruction [ loops to the corresponding bracket ] until the content of the pointer cell is '.'

## Example (P3 Programs)

Table: Sample of pairs of programs and witnesses $(p, w)$ found by the teacher

| **Example set** | **Program** | **Description** |
|---|---|---|
| $\{\langle 0, 0\rangle, \langle 10, 10\rangle\}$ | [o>] | identity |
| $\{\langle 010000, 000010\rangle, \langle 1000, 0001\rangle\}$ | [>]+[<o] | reverse |
| $\{\langle 011, 11\rangle, \langle 10001, 11\rangle\}$ | [-[+o+]>] | filter 0 |
| $\{\langle 011, 0\rangle, \langle 10001, 000\rangle\}$ | [+[-o-]>] | filter 1 |
| $\{\langle 01, 10\rangle, \langle 0011, 1100\rangle\}$ | [+[o>+]+o] | swap 1 and 0 |
| $\{\langle 01, 11\rangle, \langle 0011, 1111\rangle\}$ | [[+]-o>] | convert 0 to 1 |
| $\{\langle 01, 00\rangle, \langle 0011, 0000\rangle\}$ | [[+]+o>] | convert 1 to 0 |
| $\{\langle 0100, 00\rangle, \langle 001, \rangle, \langle 00, 00\rangle\}$ | [+>]<[-o<] | remove before last 1 |
| $\{\langle 0100, 1000\rangle, \langle 10010, 00101\rangle\}$ | >[o>]<[<]>o | left shift |

## Experimental scenarios

- $D_{\geq 1}$-**teaching book**: Witness cardinality greater than or equal to 1
- $D_{=1}$-**teaching book**: Witness cardinality equal to 1

## Example (P3 $D_{\geq 1}$-teaching book)

Table: Some programs and witnesses belonging to the teaching book

| **Witness** | **Program** |
|---|---|
| $\{\langle 010, 010 \rangle\}$ | [o>] |
| $\{\langle , 0 \rangle, \langle 0, \rangle\}$ | -[-o] |
| $\{\langle , 111 \rangle, \langle 1, \rangle\}$ | +[+ooo] |
| $\{\langle 101, 0011 \rangle\}$ | [>oo] |

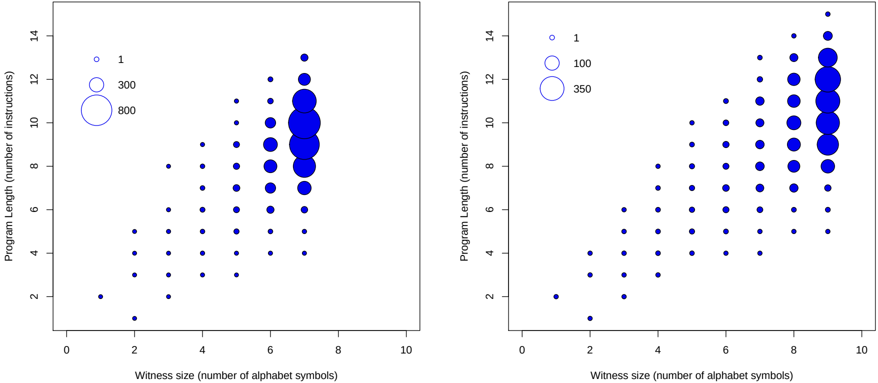



Figure: Scatter plot of programs according to their witness size, in the $D_{\geq 1}$-teaching book (left, witness cardinality greater than or equal to 1) and the $D_{=1}$-teaching book (right, witness cardinality equal to 1). Sizes of the circles are proportional to the number of entries

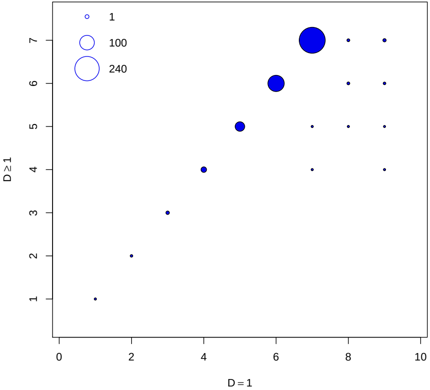



Figure: Comparing size of witnesses in experiment $D_{=1}$ ($x$-axis) and experiment $D_{\geq 1}$ ($y$-axis). Sizes of the circles are proportional to the number of entries

## Example ( $D_{\geq 1}$ vs $D_{=1}$)

Table: Some programs in both teaching books where the witness size is smaller in $D_{\geq 1}$ than in $D_{=1}$

| **Program** | $D_{\geq 1}$ | Size | $D_{=1}$ | Size |
|---|---|---|---|---|
| +[o>]-o | $\{\langle, 01\rangle, \langle 1, 1\rangle\}$ | 4 | $\{\langle 0000, 10001\rangle\}$ | 9 |
| -[ooo>]+o | $\{\langle, 1110\rangle, \langle 0, 0\rangle\}$ | 6 | $\{\langle 11, 0001110\rangle\}$ | 9 |
| -[o>]+o | $\{\langle, 10)\rangle, \langle 0, 0\rangle\}$ | 4 | $\{\langle 1001, 00010\rangle\}$ | 9 |

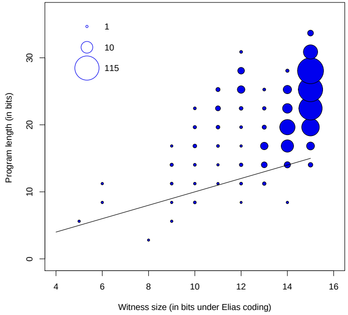



Figure: Scatter plot of program length in bits versus witness size in bits, for the new teaching book using Elias coding. Size of circles proportional to number of programs. The straight line is the unit diagonal

## Teaching Concepts vs Teaching Examples

If a teacher wants to send a concept it is frequently more efficient to send its optimal witness set under our schema than to send the program itself

## Observations

- The computability of the theoretical setting is translated into a universal language, P3
- Teaching books have a high number and high diversity of programs.
  - No collapsing phenomenon of a few programs capturing almost all witness sets
- The experiment with limited witness cardinality equal 1 shows higher overfitting than the general setting
- It is common to find witness sets that are smaller than the programs they identify, more pronounced for efficient encodings (Elias delta)

# Applications

- Traditional and new applications of machine teaching can be strengthened by the teaching size paradigm:
  - Human-computer interaction, education, trustworthy AI and coding
- Interaction with artificial agents can benefit from a better understanding of how humans teach concepts

## Example

- A user can tell an assistant that she likes peas with roasted chicken and potatoes, but dislikes peas with fish, and the system can infer that this is sufficient to infer the kind of recipes the user likes
- A model of teaching where these short examples are put at the same level of examples having millions of ingredients does not make sense

# Experimental Setting

## Choice of Concepts and Examples

- Concepts of different complexity:
  - **VL-C** (very low complexity): 2 programs with 1-2 operators
  - **L-C** (low complexity): 2 programs with 3-4 operators
  - **H-C** (high complexity): 2 programs those with 5-6 operators
  - **VH-C** (very high complexity): 2 programs with 7-8 operators
- Three batches with increasing evidence:
  - **WS:** Minimal witness sets between 1 and 3 examples
  - **AS I:** First batch with 2 extra examples
  - **AS II:** Second batch with 3 extra examples

## Example (Choice of Concepts and Examples)

Table: Experimental setting for the learning of concepts $c_1$ and $c_4$.

| Id | P3 | Description | WS | AS I | AS II | Test |
|---|---|---|---|---|---|---|
| $c_1$ | o | *Print the first character of a string* | {('0','0')} | {('111001','1'), ('110101','1')} | {('100110','1'), ('01010','0'), ('111100','1')} | {('00000','0'), ('11100','1'), ('00111','0'), ('11010','1'), ('0010', '0')} |
| $c_4$ | o+oo | *If input is '', print '00'; else if input starts with '0', print '011'; else print '1'* | {('0','011')} | {('10','1'), ('001','011')} | {('00','011'), ('0001','011'), ('000','011')} | {('01011','011'), ('0101','011'), ('0010','011'), ('100','1'), ('1','1')} |

## Learners

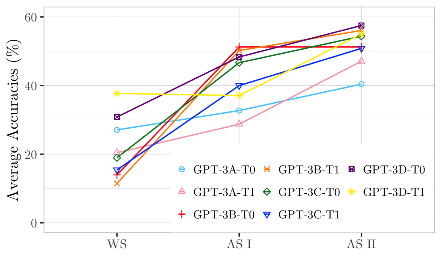
- **Humans**: 30 human participants with ages between 18 and 54
- **MagicHaskeller (MH)**: A general-purpose inductive functional programming system
- **Louise**: An inductive logic programming system
- **GPT2**: A transformer-based language model (774M-parameter version)
  - **GPT2-ens**: The most frequent result (among 40 extrapolations with the same query)
  - **GPT2-exp**: The average result (among 40 extrapolations with the same query)
- **GPT3**: Next generation system from the GPT family (from 2.7B to 175B parameters) (T0 deterministic, T1 non-determinisc like GPT2-ens)
  - **GPT-3A** Ada
  - **GPT-3B** Babbage
  - **GPT-3C** Curie
  - **GPT-3D** Da Vinci

## Example (Prompts)

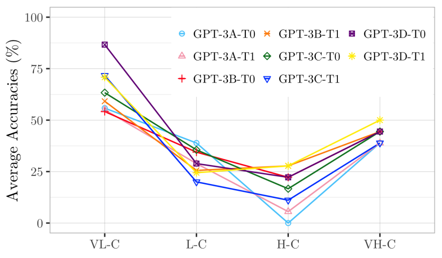
| System | Prompt | Observations |
|---|---|---|
| GPT-2 | Input1: 0, Output1: 0; Input2: 00000, Output2: | The numbering aims to logically connect and distinguish the input-output pairs. |
| GPT-3 | Input: 0<br>Output: 0<br>Input: 00000<br>Output: | The separator is the line break with no numbering, following the prompt style guidelines from OpenAI API. |

# Results

## Results for eight GPT-3 configurations.

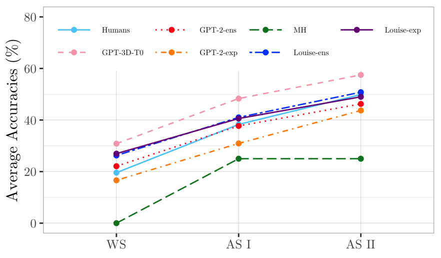



(a) Mean GPT-3 accuracy by teaching batch

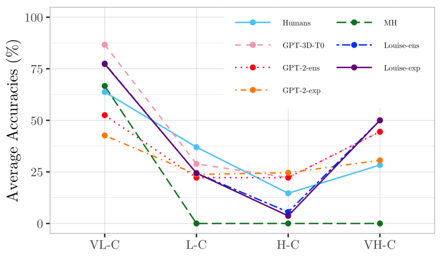



(b) Mean GPT-3 accuracy by concept complexity level.

## Comparison against Humans using 8 Concepts

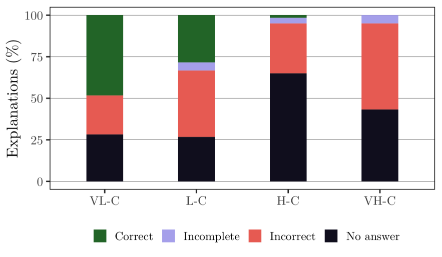



(c) Mean accuracy by teaching batch.

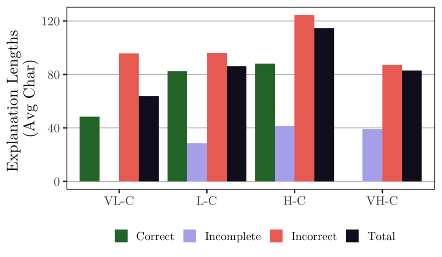



(d) Mean accuracy by concept complexity level.

## Human Explanations

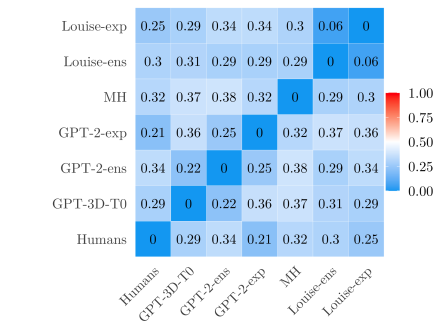



(e) Types of human explanations.

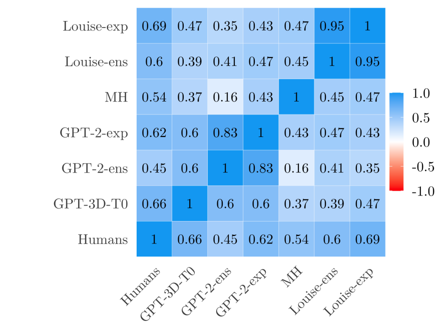



(f) Size of human explanations.

## Behaviour similarity




(g) Error disagreement matrix.




(h) Phi coefficient matrix.

## Performance of Language Models for a Larger Concept Sample

Table: Average accuracy obtained by GPT-3D-T0 with 300 concepts

| GPT-3D-T0 | Batch | VL-C (%) | L-C (%) | H-C (%) | VH-C (%) | Row Mean |
|---|---|---|---|---|---|---|
| | WS | 37.00 | 20.05 | 6.71 | 10.02 | 18.44 |
| | AS (I) | 47.00 | 38.56 | 18.76 | 19.16 | 30.87 |
| | AS (II) | 75.00 | 46.70 | 29.55 | 22.35 | 43.40 |
| Column Mean | | 53.00 | 35.10 | 18.34 | 17.18 | |

# Conclusions

- The concept of teaching dimension is counter-intuitive for languages where examples are structured and can be arbitrarily long
  - A small teaching dimension could be obtained with a very long example
  - Teaching dimension leads to incomputability for universal languages
- We have introduced the teaching size and analyzed the computability of the teaching procedure
- The experimental section supports the intuition that more examples usually pave the way for witnesses of smaller size
  - The size of the program is almost always larger than the optimal witness set used to identify it
- More realistic and powerful analysis of machine teaching and closer connections with learnability theory and communication

- Machine teaching can be employed to compare machine learning systems and humans
    - Language models
    - IP systems
    - Humans
- We have compared the accuracy of the learners from examples generated with Machine Teaching:
    - Different complexity of Concepts
    - Different size of evidence
    - Agreement in behaviour

## Main Finding

- Large models are good at capturing low-complexity patterns that depend on simplicity priors, **distilling some kind of Occam's razor for few-shot inference**
- Results with complex concepts or those including loops are poor
  - Larger models in the future may not be able to solve complex algorithmic concepts
  - **This distillation may be limited to sequential (if-then-else) algorithmic patterns**

## Limitations and Future Work

- MH and Louise follow simplicity priors but high dependency on background knowledge.
    - Not only test more powerful language models but other AI approaches!
- The use of humans for comparison limits the number of concepts and test examples
    - Keep on performing studies comparing humans and language models!
- The MT setting we have used assumes perfect alignment and identification
    - Turn MT probabilistic where more evidence increases the confidence of identification

## Take-Away

- Machine teaching can bring very important insights about what language models can do and the number —or the size— of examples they need.

## Papers

- **Think Big, Teach Small: Do Language Models Distil Occam's Razor?** G Jaimovitch-Lopez, D Castellano Falcón, C Ferri, J Hernández -Orallo *Advances in Neural Information Processing Systems 34*
- **The teaching size: computable teachers and learners for universal languages** JA Telle, J Hernández-Orallo, C Ferri *Machine Learning 108 (8), 1653-1675*
- **Teaching and Explanation: Aligning Priors between Machines and Humans** Hernández-Orallo, C Ferri, *Human-Like Machine Intelligence. Oxford University Press*

- Questions?