Neuro-Symbolic Artificial Intelligence

Nils Holzenberger

February 15, 2024

Outline

1. Neuro-Symbolic AI
   - Symbolic AI
   - Neural AI
   - Neuro-Symbolic AI?
   - Neuro-Symbolic AI in Machine Learning
2. Course overview + logistics
3. Prolog
   - History
   - The Simpsons
   - Objects in Prolog
   - Unification in Prolog
   - Operators
   - Lists
   - Prolog resolution strategy

What is AI?

What is AI?

- Tasks
- Computers
- Cognitive objects

Examples

- I'm listening.

Examples

- I'm listening.
- Deep Blue (chess)
- Eliza (chatbot)

Characteristics

Characteristics

- Works with symbolic knowledge (e.g. logical rules, grammars)
- Time as discrete order (e.g. loop index)
- Explicit representations (e.g. predicates, phrases)
- Explicit inferences (e.g. logical deduction, rewriting)
- Perfect matches
- Combinatorial (algorithms)
- Definite errors
- Interpretability (generally, execution trace)
- Non-continuous (no topology)

Strengths

Strengths

- Processes structures
- Interpretability (XAI, find relevant explanations)
- (Re)-use of background knowledge (e.g. 80 km/h)
- Manages strict constraints
- May reach perfection
- Manages relations (right_of, borrow)
- Instant adaptation to context (*here*, *for a long time*)
- May imitate some cognitive processes (argumentation)
- May solve combinatorial problems (planning)

Weaknesses

Weaknesses

- Can't be fed with raw data
- Doesn't (always) scale up
- Highly sensitive to errors (*garbage in, garbage out*)
- Easy to fool
- Requires a lot of design

Examples

Examples

- GPT-3 (neural language model)
- AlphaZero (go)

Characteristics

- Also called distributed AI, connectionist AI

Characteristics

- Also called distributed AI, connectionist AI
- Uses a neural network to approximate a function
- Relies on data to fit the parameters of the neural network
- Uses probabilistic modeling, i.e. uses tools from statistical machine learning

Strengths

Strengths

- The neural network can approximate any function! In particular those hard to describe with symbols
- The approximation of the function gets better with more data (PAC learnability)
- In practice, it has proven more effective than symbolic systems, whenever the problem required the use of a lot of background knowledge and heuristics (see knowledge bottleneck)

Weaknesses

Weaknesses

- Doesn't avoid the problem of structure — the neural network's inputs and outputs must be processed
- Doesn't (always) scale up
- Sensitive to data quality
- Sensitive to input (adversarial inputs)
- Computationally intensive at inference time, and even more at training time

Examples

- IBM Watson (won at Jeopardy in 2012)
- Speech acquisition (a few slides from now)

Characteristics

- Uses symbolic AI as a framework
- Uses neural AI to approximate certain functions
- With that definition, almost every instance of neural AI is also neuro-symbolic (counter-example: GPT models)
- Neuro-symbolic AI explicitly involves symbols and neural networks. Sometimes neural networks manipulate symbols, sometimes symbols manipulate neural networks.

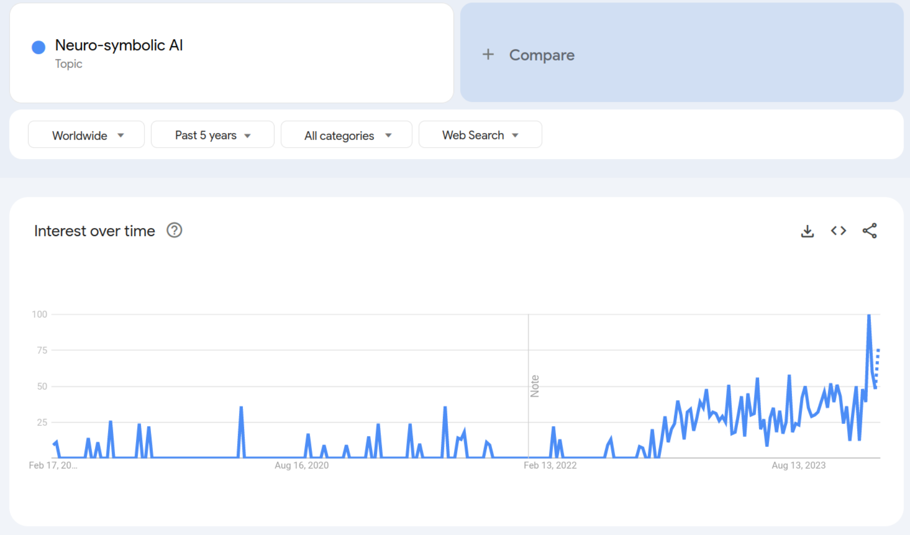
Strengths and Weaknesses

Strengths $=$ Strengths of symbolic AI $\cup$ Strengths of neural AI
Weaknesses $=$ Weaknesses of symbolic AI $\cap$ Weaknesses of neural AI

Challenges

- Knowing enough about both symbolic and neural AI
- Training the neural network
- Interface between discrete and continuous

Google searches



NeSy workshop

Neuro-Symbolic Artificial Intelligence

Workshop series on Neural-Symbolic Learning and Reasoning

Steering Committee (Neural-Symbolic Learning and Reasoning Association)

- Artur d'Avila Garcez (President)
- Daniel Silver (Vice President)
- Pascal Hitzler
- Peter Földiák (Treasurer)
- Kai-Uwe Kühnberger
- Luis C. Lamb
- Luc de Raedt

Further information pertaining to the Steering Committee and the Association.

NeSy Workshops and Seminars:

- NeSy 2023, 17th International Workshop on Neural-Symbolic Learning and Reasoning.
- NeSy 2022, Sixteenth International Workshop on Neural-Symbolic Learning and Reasoning at IJCLR 2022
- NeSy'20/21, Fifteenth International Workshop on Neural-Symbolic Learning and Reasoning at IJCLR-20/21 - online recordings
- NeSy'19, Fourteenth International Workshop on Neural-Symbolic Learning and Reasoning at IJCAI-19
- NeSy'18, Thirteenth International Workshop on Neural-Symbolic Learning and Reasoning at HLAI-18
- NeSy'17, Twelveth International Workshop on Neural-Symbolic Learning and Reasoning
- NeSy'16, Eleventh International Workshop on Neural-Symbolic Learning and Reasoning at HLAI 2016
- NeSy'15, Tenth International Workshop on Neural-Symbolic Learning and Reasoning at IJCAI-15
- NeSy'13, Ninth International Workshop on Neural-Symbolic Learning and Reasoning at IJCAI-13
- NeSy'12, Eighth International Workshop on Neural-Symbolic Learning and Reasoning at AAAI-12
- NeSy'11, Seventh International Workshop on Neural-Symbolic Learning and Reasoning at IJCAI-11
- NeSy'10, Sixth International Workshop on Neural-Symbolic Learning and Reasoning at AAAI-10
- NeSy'09, Fifth International Workshop on Neural-Symbolic Learning and Reasoning at IJCAI-09
- NeSy'08, Fourth International Workshop on Neural-Symbolic Learning and Reasoning at ECAI-08
- NeSy'07, Third International Workshop on Neural-Symbolic Learning and Reasoning at IJCAI-07
- NeSy'06, Second International Workshop on Neural-Symbolic Learning and Reasoning at ECAI-06
- NeSy'05, First International Workshop on Neural-Symbolic Learning and Reasoning at IJCAI-05

Speech acquisition

- Infants learn speech effortlessly

Speech acquisition

- Infants learn speech effortlessly
- Language is a strange object:
  - it manipulates discrete units (words, syllables, phonemes...)
  - that are realized in a continuous space (acoustics)
  - rules to combine units have strict results (grammatical or not)
  - but the rules are hard to capture: meaning of words/symbols change in context

Speech acquisition

- Infants learn speech effortlessly
- Language is a strange object:
  - it manipulates discrete units (words, syllables, phonemes...)
  - that are realized in a continuous space (acoustics)
  - rules to combine units have strict results (grammatical or not)
  - but the rules are hard to capture: meaning of words/symbols change in context
- Is it possible to at least acquire the acoustic units that make up speech?
  - What is the problem? Allophones + speaker variability
  - Clustering?
  - Auto-encoder?
  - Auto-encoder with discrete units

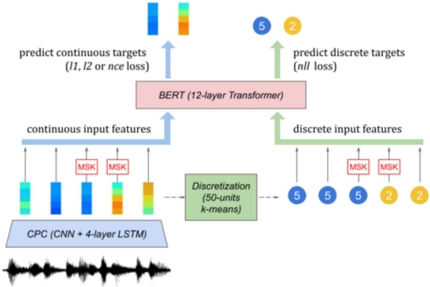
Speech acquisition

IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING, VOL. 16, NO. 6, OCTOBER 2022 1415

Are Discrete Units Necessary for Spoken Language Modeling?

Tu Anh Nguyen, Benoit Sagot, and Emmanuel Dupoux

Abstract—**Recent work in spoken language modeling shows the possibility of learning a language unsupervisedly from raw audio without any text labels. The approach relies first on transforming the audio into a sequence of discrete units (or pseudo-text) and then training a language model directly on such pseudo-text. Is such a discrete bottleneck necessary, potentially introducing irreversible errors in the encoding of the speech signal, or could we learn a language model without discrete units at all? In this work, we study the role of discrete versus continuous representations in spoken language modeling. We show that discretization is indeed essential for good results in spoken language modeling. We show that discretization removes linguistically irrelevant information from the continuous features, helping to improve language modeling performances. On the basis of this study, we train a language model on the discrete units of the HuBERT features, reaching new state-of-the-art results in the lexical, syntactic and semantic metrics of the Zero Resource Speech Challenge 2021 (Track 1 - Speech**



Speech acquisition

What was hard about this? Differentiating through the discretization

NeurSymAI in Cognitive Science

- Are humans more like a neural network or like a symbolic program?
- This debate is also fed by *System 1* and *System 2* in *Thinking, Fast and Slow* by Daniel Kahneman

Symbols and mental programs: a hypothesis about human singularity

Stanislas Dehaene,[1,2,*] Fosca Al Roumi,[1] Yair Lakretz,[1] Samuel Planton,[1] and Mathias Sablé-Meyer[1]

Natural language is often seen as the single factor that explains the cognitive singularity of the human species. Instead, we propose that humans possess multiple internal languages of thought, akin to computer languages, which encode and compress structures in various domains (mathematics, music, shape...). These languages rely on cortical circuits distinct from classical language areas. Each is characterized by: (i) the discretization of a domain using a small set of symbols, and (ii) their recursive composition into mental programs that encode nested repetitions with variations. In various tasks of elementary shape or sequence perception, minimum description length in the proposed languages captures human behavior and brain activity, whereas non-human primate data are captured by simpler nonsymbolic models. Our research argues in favor of discrete symbolic models of human thought.

Highlights

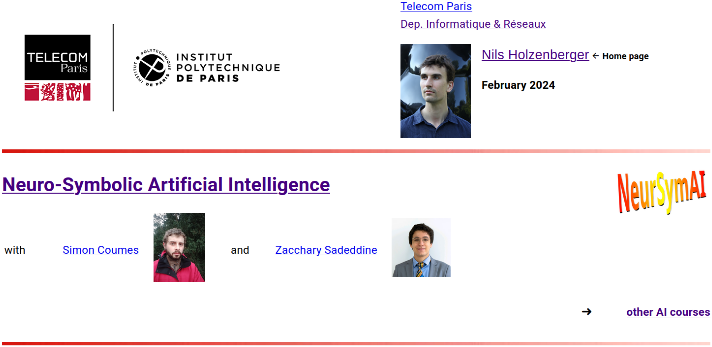
Accounting for human spatial memory requires the postulation of a mental language that can recursively compose primitives of number, space, and repetition with variations.

The same language accounts for the human perception of binary auditory sequences.

Minimum description length, rather than actual sequence length, predicts human working memory for auditory and visual sequences.

Course website

Have you received the login info?
Have you done the first lab session?



`https://ailab.r2.enst.fr/LKR`

The course website will be updated with slides, lab sessions, answers to FAQ...

Course overview

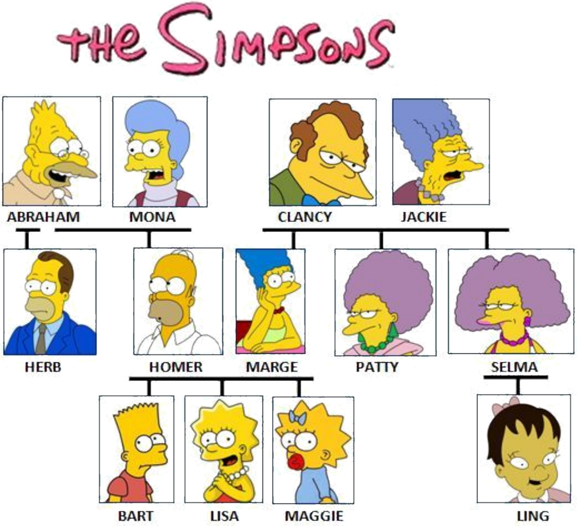
	Topics
	Overview
13 feb 2024 13:30 ➜ 27 feb 2024 13:30	First steps in Prolog
27 feb 2024 13:30 ➜ 5 mar 2024 13:30	Problem solving and Knowledge representation
5 mar 2024 13:30 ➜ 12 mar 2024 13:30	Propositional Logic
12 mar 2024 13:30 ➜ 19 mar 2024 13:30	Predicate Logic
19 mar 2024 13:30 ➜ 26 mar 2024 13:30	Symbolic machine learning
26 mar 2024 13:30 ➜ 2 apr 2024 13:30	ProbLog: Probabilistic Prolog
2 apr 2024 13:30 ➜ 9 apr 2024 13:30	Deep ProbLog: Neural Probabilistic Prolog
9 apr 2024 13:30 ➜ 16 apr 2024 13:30	Neuro-Symbolic Programming

Validation

- lab sessions: answers are recorded and graded
- final quizz: 90 min, no documents, no functioning devices

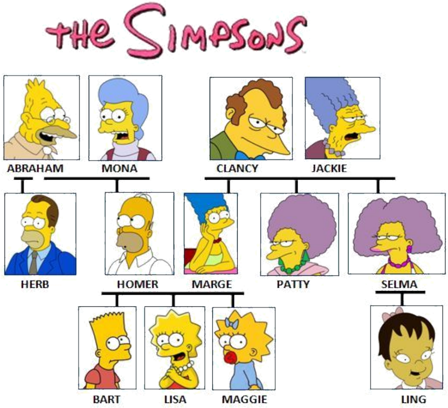
- 1965: resolution (Robinson)
- 1972: **Prolog** created by A. Colmerauer and P. Roussel in Luminy.
- 1980: Prolog acknowledged as a major A.I. language
- Now various versions (e.g. **SWI-Prolog**), some of them used in Constraint Programming
- 1977: Datalog
- pyDatalog, **ProbLog**, ProGol...

The Simpsons' genealogy



The Simpsons' genealogy

```
parent(marge, lisa).
parent(marge, bart).
parent(marge, maggie).
parent(homer, lisa).
```



The Simpsons' genealogy

```prolog
parent(marge, lisa).
parent(marge, bart).
parent(marge, maggie).
parent(homer, lisa).
parent(homer, bart).
parent(homer, maggie).
parent(abraham, homer).
parent(abraham, herb).
```

```prolog
parent(mona, homer).
parent(jackie, marge).
parent(clancy, marge).
parent(jackie, patty).
parent(clancy, patty).
parent(jackie, selma).
parent(clancy, selma).
parent(selma, ling).
```

Some predicates

```
child(X, Y) :- parent(Y, X).
grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
```

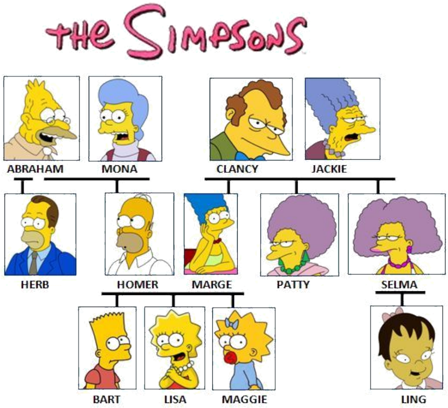
Some more facts

```
female(marge).
female(lisa).
male(bart).
female(maggie).
male(homer).
male(abraham).
male(herb).
```

```
female(mona).
female(jackie).
male(clancy).
female(patty).
female(selma).
female(ling).
```

Some more predicates

```
child(X, Y) :-
    parent(Y, X).
grandparent(X, Y) :-
    parent(X, Z),
    parent(Z, Y).
mother(X, Y) :- ?
sister(X, Y) :- ?
ancestor(X, Y) :- ?
cousin(X, Y) :- ?
```



Basic objects

Integers	`9`
Floats	`1.5`
Strings	`"this is a string"`
Atoms — definite objects	`marge_simpson`
Variables — objects that can take on a definite value	`X`

More complicated objects

Predicates — a bit like a boolean function	`parent/2`
Operators — a special case of predicates	`=`
Terms — *constant*	`marge_simpson`
Terms — *structure*	`mother(marge_simpson,X)`
Terms — *variable*	`Person`
Clauses — *facts*	`parent(homer, bart).`
Clauses — *rules*	`child(X,Y) :- parent(Y, X).`

Objects

Definitions:

`http://www.projog.org/prolog-introduction.html`

Definition + example

- *Unification* takes two terms and checks whether they may be equivalent. Unification may succeed or fail. If it succeeds, variables may be bound together.

Definition + example

- *Unification* takes two terms and checks whether they may be equivalent. Unification may succeed or fail. If it succeeds, variables may be bound together.
- Examples
  - Unification of `X` and `2` succeeds and returns `X=2`
  - Unification of `parent(marge, lisa)` and `parent(Who, lisa)` succeeds and returns `Who=marge`
  - Unification of `parent(Xvar, lisa)` and `parent(Who, lisa)` succeeds and returns `Xvar=Who`
  - Unification of `parent(marge, lisa)` and `parent(bart, lisa)` fails
  - Unification of `parent(marge, lisa)` and `child(lisa, marge)` fails
  - Unification of `a(X, X)` and `a(2, 3)` fails

Definition + example

- *Unification* takes two terms and checks whether they may be equivalent. Unification may succeed or fail. If it succeeds, variables may be bound together.
- Examples
  - Unification of `X` and `2` succeeds and returns `X=2`
  - Unification of `parent(marge, lisa)` and `parent(Who, lisa)` succeeds and returns `Who=marge`
  - Unification of `parent(Xvar, lisa)` and `parent(Who, lisa)` succeeds and returns `Xvar=Who`
  - Unification of `parent(marge, lisa)` and `parent(bart, lisa)` fails
  - Unification of `parent(marge, lisa)` and `child(lisa, marge)` fails
  - Unification of `a(X, X)` and `a(2, 3)` fails
- It is a basic building block of Prolog
- Unification is performed using this operator: `=`

Exercises

- Can `a(B,C)` and `a(2,3)` be unified?

Exercises

- Can `a(B,C)` and `a(2,3)` be unified?
- `a(X,Y,L)=a(Y,2,carole)`?

Exercises

- Can `a(B,C)` and `a(2,3)` be unified?
- `a(X,Y,L)=a(Y,2,carole)`?
- `a(X,X,Y)=a(Y,u,v)`?

Exercises

- Can `a(B,C)` and `a(2,3)` be unified?
- `a(X,Y,L)=a(Y,2,carole)`?
- `a(X,X,Y)=a(Y,u,v)`?
- `p(X,b(Z,a),X)=p(Y,Y,b(V,a))`?

Arithmetic

- `+`, `-`, `*`, `/`, `div`, `mod`
- Usage: `X is 6-2`, results in `X=4`
- This is a special case of unification; it is in fact a structure `-(6,2)` that can be evaluated

Comparison

- `<`, `>`, `>=`, `=<`, `=:=`, `=\=`
- The first 4 are most useful, and can be used on integers and strings
- `X = Y` succeeds if `X` unifies with `Y`
- `X \= Y` succeeds if `X` fails to unify with `Y` (i.e. `X = Y` fails)
- `X == Y` succeeds if `X` and `Y` are syntactically identical
- `X \== Y` succeeds if `X == Y` fails
- You can think of *succeed* and *fail* as *true* and *false*

Lists

- `[a, b, c, d]`
- `[bart_simpson, homer, Z, pred(A,Z)]`
- `[bart_simpson, homer, Z, parent(A,Z)]` unifies with `[Person, homer, X, parent(homer, marge)]` with `Person=bart_simpson, X=Z, Z=marge, A=homer`
- `[bart_simpson, homer, parent(A,Z)]` unifies with `[H|T]` with `H=bart_simpson, T=[homer, parent(A,Z)]`
- `[X,Y|T]` unifies with `[bart_simpson, homer, parent(A,Z)]` with `X=bart_simpson, Y=homer, T=[parent(A,Z)]`
- `[a,b,c]` does not unify with `[b,c]`
- For two lists to unify, every single one of their elements must unify. Otherwise the whole unification fails.

Exercises

- `contains(X,L)` checks whether a list contains an element
- `list_length(L,N)` computes the length of a list
- `split(List,Left,Right)` splits a list into 2 equally-sized parts
- `extract(X,List,Remainder)` takes an element from a list
  `extract(a,[a,b,c],[b,c])` succeeds
  `extract(b,[a,b,c],[b,c])` fails
- `permutation(L,X)` permutes a list
- `attach(L1,L2,L3)` appends two lists

Some solutions

`list_length(L,N)` is true when the list `L` contains `N` elements

```
list_length([],0).
list_length([_|T],N) :-
    list_length(T, N1),
    N is N1+1.
```

Note that the following version failed:

```
list_length([],0).
list_length([_|T],N) :-
    N1 is N-1,
    list_length(T, N1).
```

Some solutions

`contains(X,L)` checks whether a list `L` contains an element `X`

```
contains1(X,[Y]) :-
    X=Y.
contains1(X,[X|T]).
contains1(X,[H|T]) :-
    contains1(X,T).
```

A more "Prology" version:

```
contains2(X, [X|_]).
contains2(X, [_|T]) :-
    contains2(X, T).
```

The query `contains(a,[a,b,c])` should succeed, `contains(a,[b,c])` should fail, and `contains(X,[b,c])` should return `X=b; X=c`
Note that `contains1` and `contains2` behave slightly differently: compare all possible solutions enumerated by `contains(X,[b,c])`.

Some solutions

`extract(X,List,Remainder)` takes an element from a list: it succeeds if `Remainder` is obtained by removing `X` from `List`

`extract(a,[a,b,c],[b,c])` succeeds

`extract(b,[a,b,c],[b,c])` fails

Solution:

```
extract(X, [X|T], T).
extract(X, [H|T], [H|Remainder]) :-
    extract(X, T, Remainder).
```

Note that this can called in different ways:

`extract(b,[a,b,c],L).`

`extract(b,L,[b,c]).`

With `extract` it is possible to both insert and extract elements from a list. This property is known as *reversibility*.

More exercises

- Duplicate each element of a list
- Intertwine two lists
- Palindrome test
- Palindrome building
- Remove redundant elements
- Test prime numbers
- Find repeated patterns in a list
- Interlace an unspecified number of lists
- Generate lists containing the terms `"A"`, `"C"`, `"T"`, `"G"` without identical consecutive terms

Ideas

- Declarativity - Reversibility
- Depth-first strategy
- Backtracking
- Recursivity
- Unification

Details

Next class.