First Order Logic with Logic Tensor Networks

Federico Bianchi
Bocconi University
f.bianchi@unibocconi.it

Logic Tensor Networks

Logic Tensor Networks

Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge*

Luciano Serafini[1] and Artur d'Avila Garcez[2]

[1] Fondazione Bruno Kessler, Trento, Italy, serafini@fbk.eu
[2] City University London, UK, a.garcez@city.ac.uk

Logic Tensor Networks

Logic Tensor Networks [Serafini+,2016] (LTNs)

LTNs = **Neural Networks** + **First Order Fuzzy Logic**

Key Aspects:

- LTNs **ground fuzzy logic in a vector space: continuous values in [0,1]**
- LTNs assign truth values to formulas using neural networks
- LTNs can learn from both data and rules
- LTNs can be used to do inferences over rules after training

Key Idea: LTNs provide a method to learn reasoning over vector spaces

Logic Tensor Networks

Logic Tensor Networks [Serafini+,2016] (LTNs)

LTNs = **Neural Networks** + **First Order Fuzzy Logic**

Key Aspects:

- LTNs ground fuzzy logic in a vector space: continuous values in [0,1]
- LTNs **assign truth values to formulas using neural networks**
- LTNs can learn from both data and rules
- LTNs can be used to do inferences over rules after training

Key Idea: LTNs provide a method to learn reasoning over vector spaces

Logic Tensor Networks

Logic Tensor Networks [Serafini+,2016] (LTNs)

LTNs = **Neural Networks** + **First Order Fuzzy Logic**

Key Aspects:

- LTNs ground fuzzy logic in a vector space: continuous values in [0,1]
- LTNs assign truth values to formulas using neural networks
- LTNs **can learn from both data and rules**
- LTNs can be used to do inferences over rules after training

Key Idea: LTNs provide a method to learn reasoning over vector spaces

Logic Tensor Networks

Logic Tensor Networks [Serafini+,2016] (LTNs)

LTNs = **Neural Networks** + **First Order Fuzzy Logic**

Key Aspects:

- LTNs ground fuzzy logic in a vector space: continuous values in [0,1]
- LTNs assign truth values to formulas using neural networks
- LTNs can learn from both data and rules
- LTNs **can be used to do inferences over rules after training**

Key Idea: LTNs provide a method to learn reasoning over vector spaces

Logic Tensor Networks: Plan

We need:

- A way to put logic in the vector space;
- A way to learn logical representation;
- A way to compute the truth score of predicate.

What we get:

- **A recursive logical language that works in the vector space.**

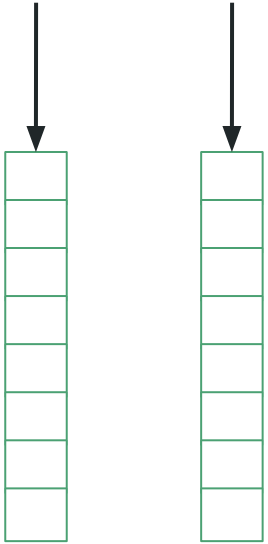
Logics in the Vector Space

Logic Tensor Networks: General Idea

parent(Susan, Ann)

Logic Tensor Networks: General Idea

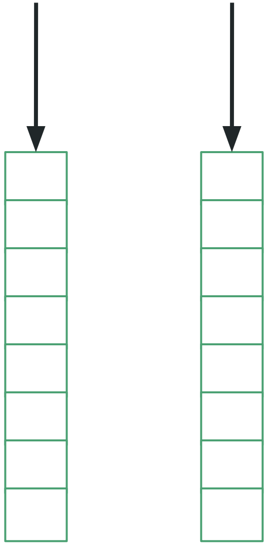
parent(Susan, Ann)



Constants are points in $\mathbf{R}^k$

Logic Tensor Networks: General Idea

parent(Susan, Ann)

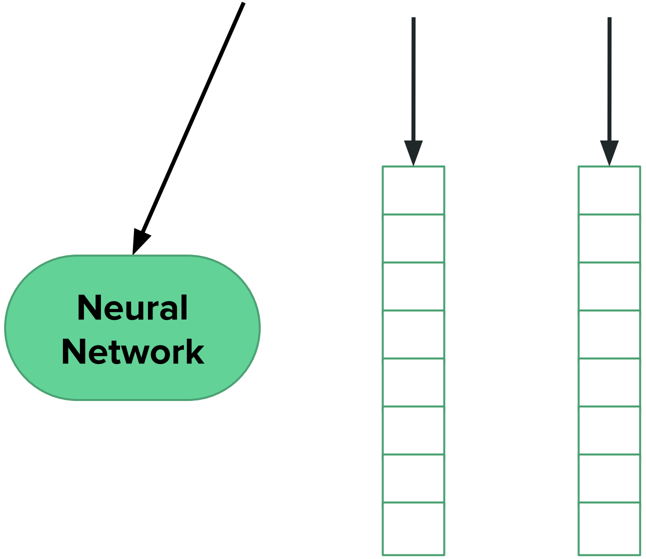


These are logical constants, but **we can learn these** vectors (similarly to what we do with word embeddings) or **initialize them** with pre-trained embeddings

Constants are points in $\mathbf{R}^k$

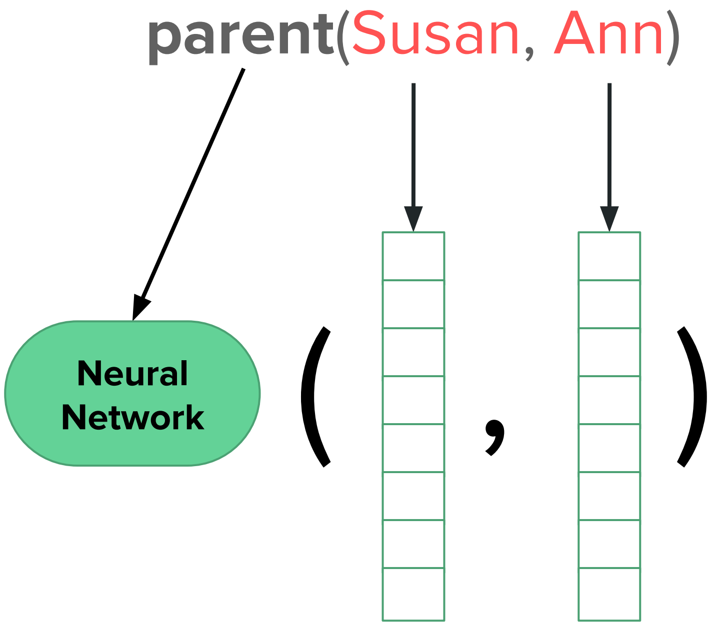
Logic Tensor Networks: General Idea

parent(Susan, Ann)

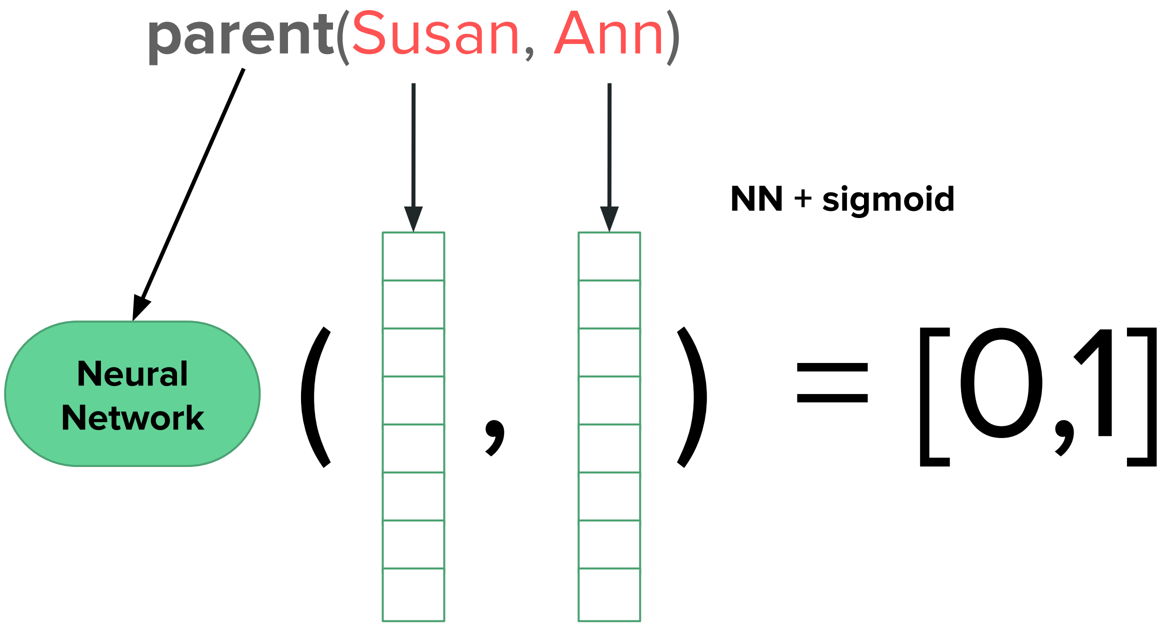


Each predicate
in LTN is a NN

Logic Tensor Networks: General Idea



Logic Tensor Networks: General Idea

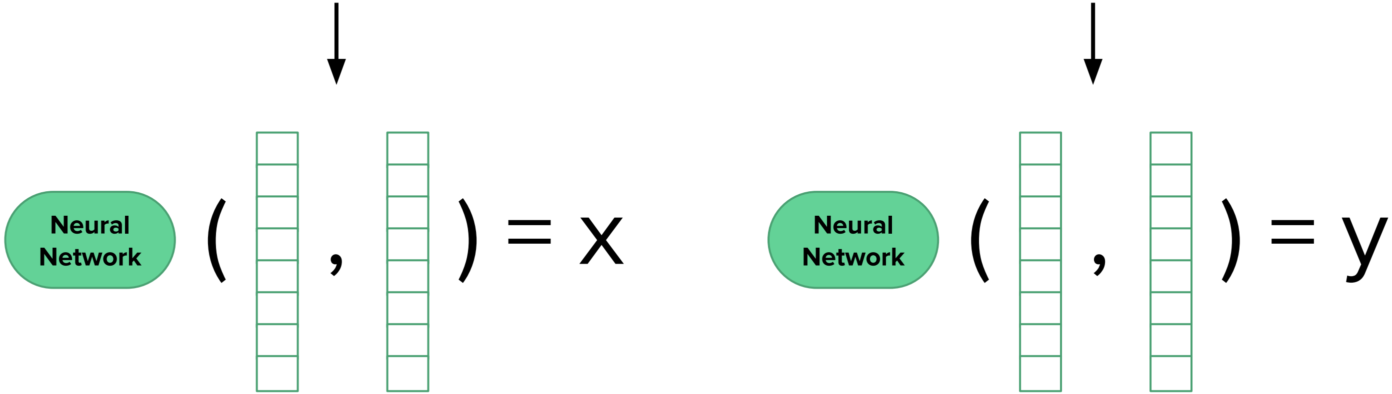


Logic Tensor Networks: General Idea

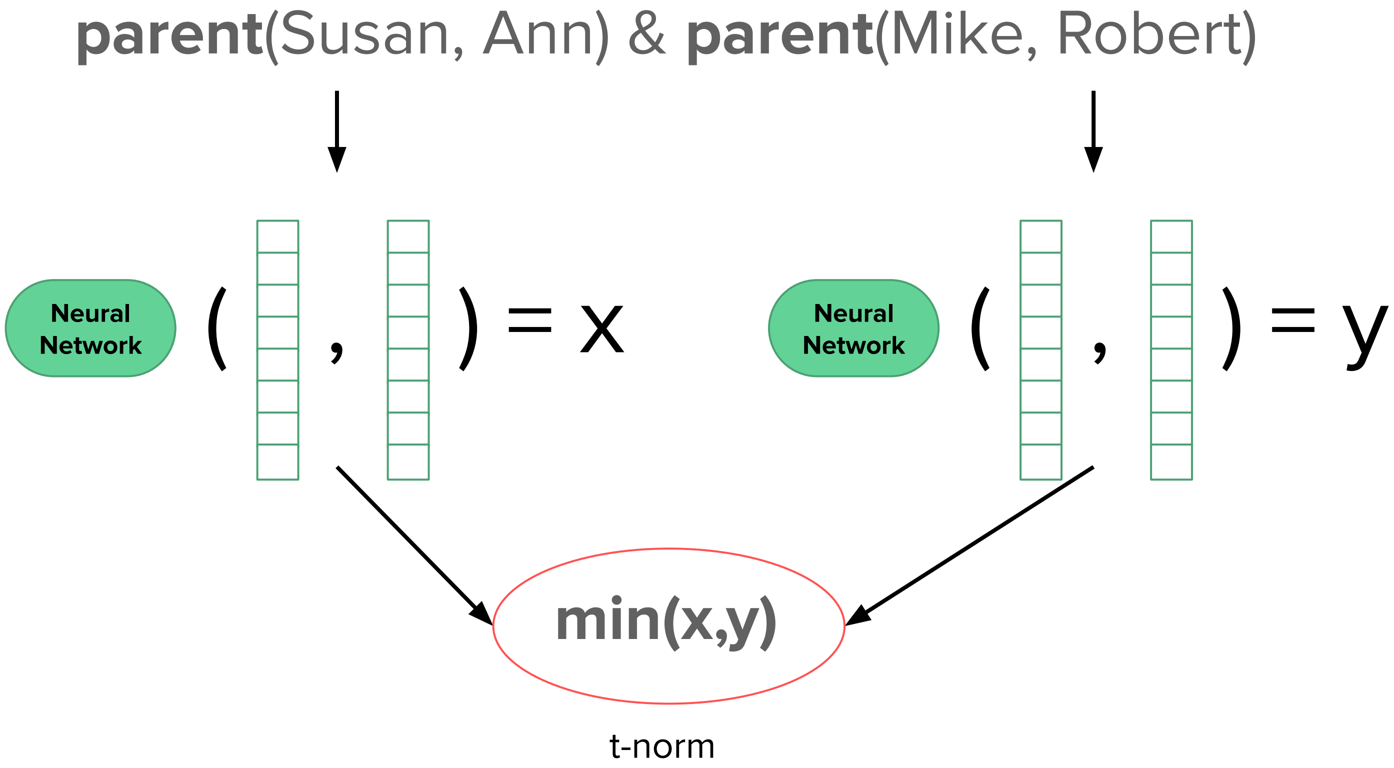
parent(Susan, Ann) & **parent**(Mike, Robert)

Logic Tensor Networks: General Idea

parent(Susan, Ann) & **parent**(Mike, Robert)

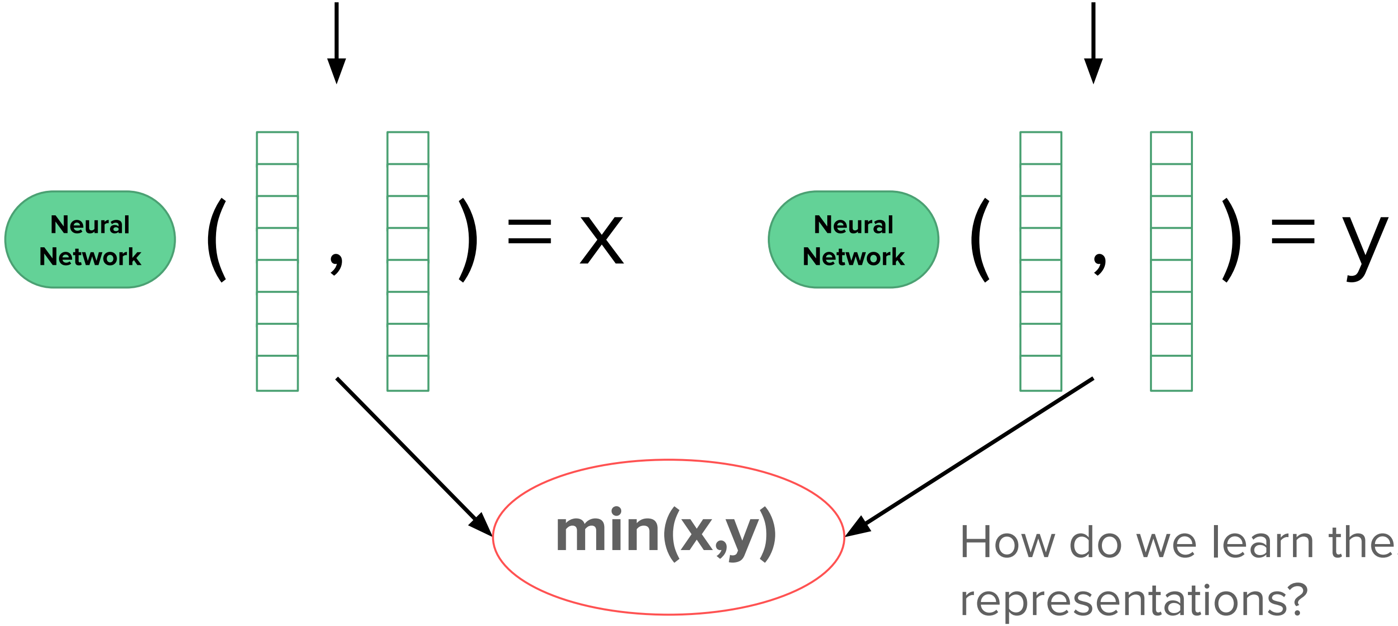


Logic Tensor Networks: General Idea



Logic Tensor Networks: General Idea

parent(Susan, Ann) & **parent**(Mike, Robert)



t-norm

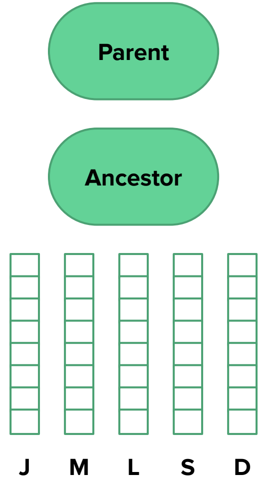
How do we learn these representations?
Backpropagation

Learning Logics in the Vector Space

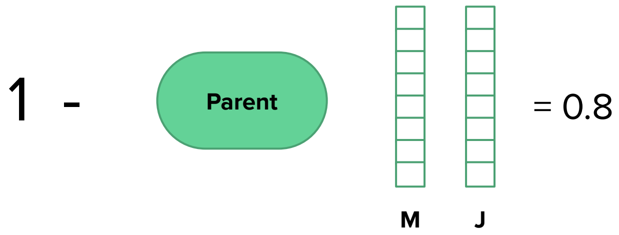
Example

KB:

- !parent(mark, john)
- parent(john,mark)
- ancestor(mark, lucas)
- parent(john, susan) | parent(john, dania)



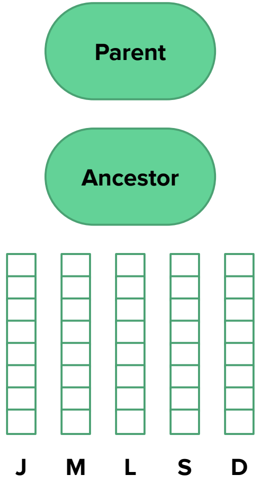
Example: Forward Pass



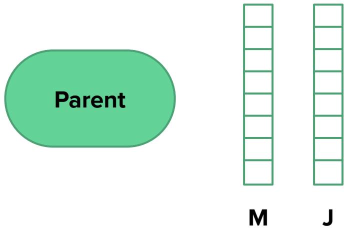
We want this axiom to be true, we need to maximize this

KB:

- **!parent(mark, john)**
- parent(john,mark)
- ancestor(mark, lucas)
- parent(john, susan) | parent(john, dania)



Example: Back Pass

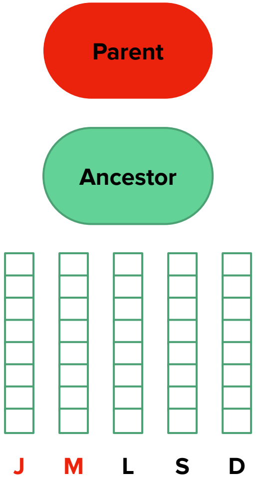
1 -  = **0.8**

Error is 0.2 (with MAE)

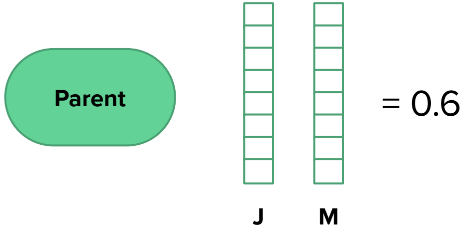
Update using backpropagation

KB:

- **!parent(mark, john)**
- parent(john,mark)
- ancestor(mark, lucas)
- parent(john, susan) | parent(john, dania)



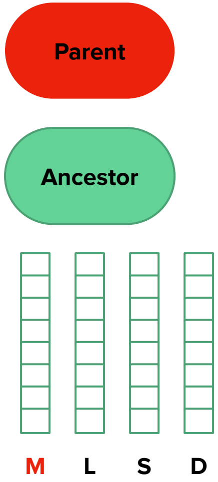
Example: Forward and Back Pass



We want to maximize this and thus we update the respective values

KB:

- !parent(mark, john)
- **parent(john,mark)**
- ancestor(mark, lucas)
- parent(john, susan) | parent(john, dania)



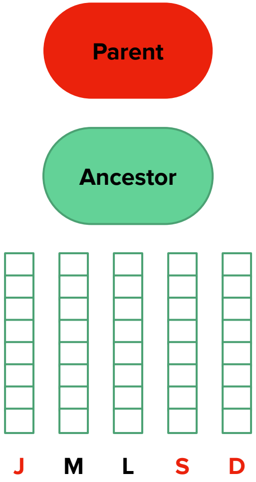
Example: Forward and Back Pass

max(0.2, 0.9) = 0.9

We want to maximize this and thus we update the respective values

KB:

- !parent(mark, john)
- parent(john,mark)
- ancestor(mark, lucas)
- **parent(john, susan)** | **parent(john, dania)**



A few more things

Logic Tensor Networks: Quantified Formulas

How do we interpret a quantified formula?

- **$\forall$x human(x)**

In LTN each variable (x) is associated to a **domain sample**

- S = {John, Paul} and **x** is defined over S, truth value of **$\forall$x human(x)** is
    - **min human(x), $\forall$ x in S.**
        - human(John) = 0.8, human(Paul) = 0.9, $\forall$x human(x) = 0.8
    - the truth value of the forall of a predicate is the value of the least true axiom.

Logic Tensor Networks: Data and Rules

LTNs can learn from both data and rules.

- Quantifiers are defined **over a domain sample.**

parent(Mark,Susan)
parent(Ron,Susan)

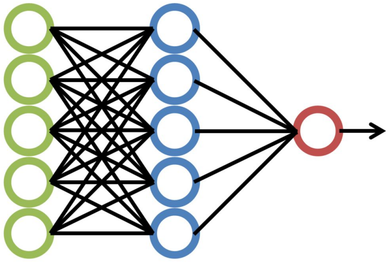
∀ x,y parent(x,y) → ancestor(x,y)

Quantifiers interpreted using an **aggregation function** (e.g., average):
∀ x P(x) = average value of P(x) in LTNs.

Logic Tensor Networks: Learning

The network is trained on a **best satisfiability task**:

- Learn the representations of:
  - **vectors** for the constants, **parameters** for the predicates in such a way that **the axioms are satisfied in the best possible way**.



Given **parent(Ann, Susan)** we expect the network to **learn representations** for **Ann**, **Susan** and **parent** in such a way that the predicted value is close to 1

Logic Tensor Networks: After Training Inference

- The trained network can be used to make novel inferences.
- Suppose we train using a dataset of ***parents*** and ***ancestors*** relationships.

Logic Tensor Networks: After Training Inference

- The trained network can be used to make novel inferences.
- Suppose we train using a dataset of ***parents*** and ***ancestors*** relationships.
- **After training** we can query LTNs on:
  - **∀ x,y ancestor(x,y) → parent(x,y)** has truth value close to 0

We get a deep recursive logical language to explore the vector space!

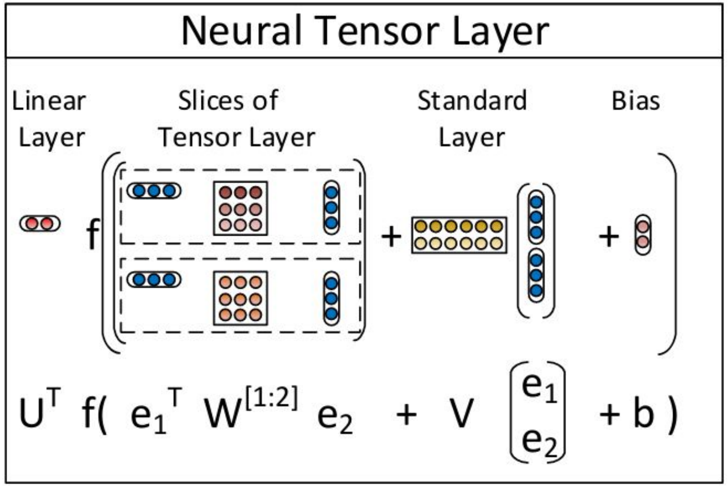
How to Compute The Truth Score

Logic Tensor Networks: Neural Tensor Network

$$g(e_1, R, e_2) = u_R^T f\left(e_1^T W_R^{[1:k]} e_2 + V_R \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + b_R\right)$$

Compute truth score

$R(e_1, e_2) \longrightarrow$



Socher, R., Chen, D., Manning, C. D., & Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. In Advances in neural information processing systems (pp. 926-934).

Logic Tensor Networks: LTN’s Neural Tensor Network

$$\mathcal{G}(P) = \sigma\left(u_P^T \tanh\left(\mathbf{v}^T W_P^{[1:k]} \mathbf{v} + V_P \mathbf{v} + B_P\right)\right)$$

- P is the axiom (e.g., **parent(Ann, Susan)**);
- **v** is **the concatenation** of the vectors of the arguments of the predicate.
  - Concatenation is order dependent => parent(Ann,Susan) != parent(Susan,Ann)

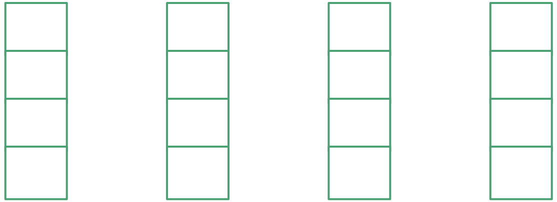
Logic Tensor Networks: LTN's Neural Tensor Network

$$\mathcal{G}(P) = \sigma\left(u_P^T \tanh\left(\mathbf{v}^T W_P^{[1:k]} \mathbf{v} + V_P \mathbf{v} + B_P\right)\right)$$

- P is the axiom (e.g., **parent(Ann, Susan)**);
- **v** is **the concatenation** of the vectors of the arguments of the predicate.
  - Concatenation can be used for higher arity-predicates like
    - **married(John, Susan, Saint Paul Church, 10/10/1984)**

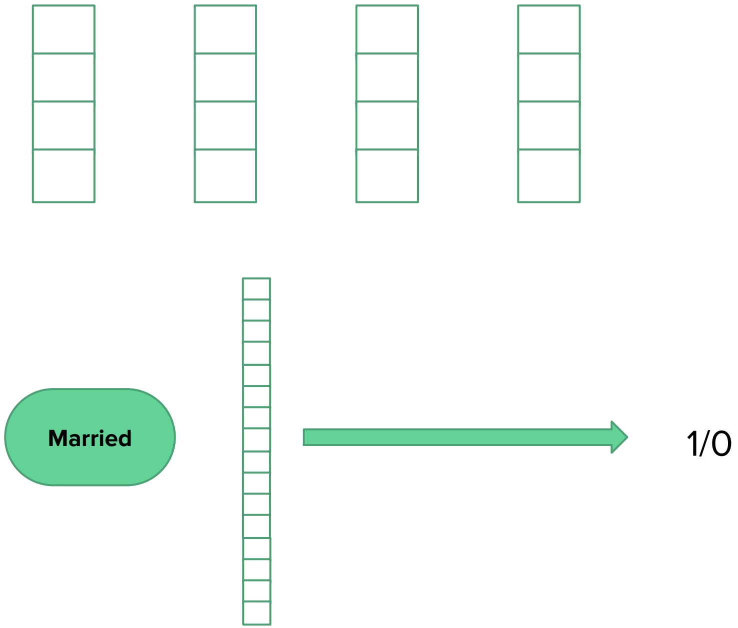
Logic Tensor Networks: LTN’s Neural Tensor Network

- married(John, Susan, Saint Paul Church, 10/10/1984)



Logic Tensor Networks: LTN's Neural Tensor Network

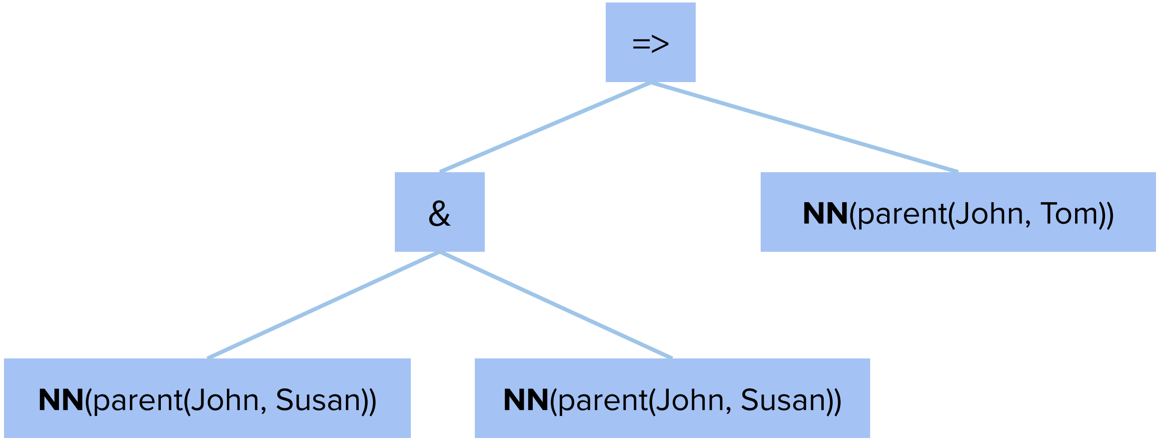
- married(John, Susan, Saint Paul Church, 10/10/1984)



Logic Tensor Networks: Complex Formulas

- **(parent(John, Susan) & parent(Susan, Tom)) => parent(John, Tom)**

Complex formulas are interpreted **recursively**.



Recap

- **LTNs** grounds logic in the vector space: each predicate is an NN that we train.
- **LTNs** gives us a recursive logical language
- We can **logically explore** the space after training with compositions of learned predicates.

Logic Tensor Networks with Embeddings

Support Logical Reasoning with Embeddings

Agent **axiomatic knowledge**:

species(**cat**)
mammal(**tiger**)
bird(**penguin**)
∀**x** (mammal(**x**) → animal(**x**))

logically Inferring something about
cat is not possible

Agent “**distributional knowledge**”: **cats** and **tigers** are more similar than **cats** and **penguins**.

Possible inferences with the support of distributional knowledge:

mammal(cat)

But also:

animal(cat)

Key Idea: combining standard logical reasoning with a embeddings

Sub-symbolic Commonsense: Distributional Semantics

Distributional Hypothesis:

similar words tend to appear in similar **contexts**

[Wittgenstein1953, Firth1957]

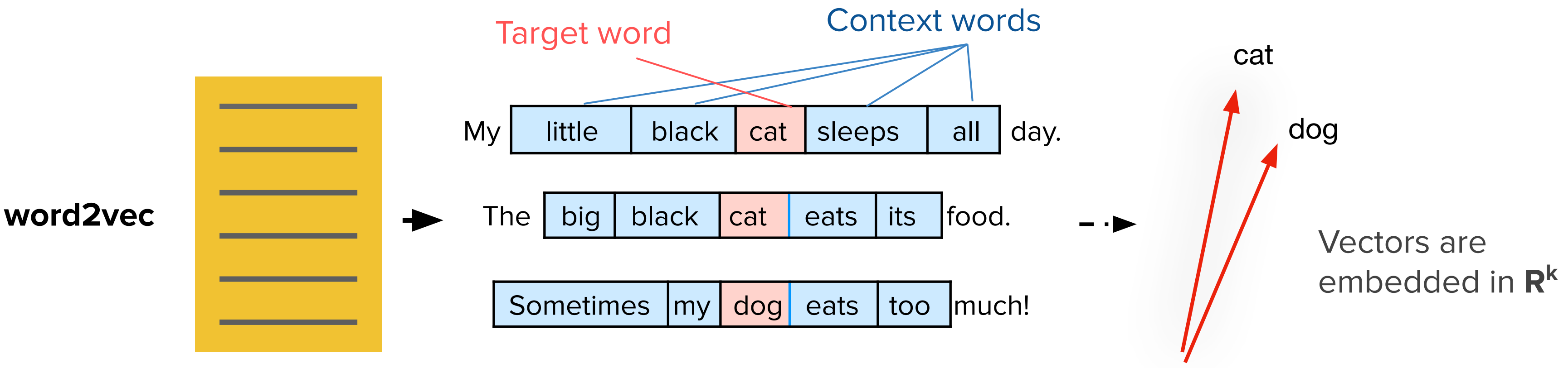
What is the meaning of **bardiwac?** [Lenci&Evert]:

- The drinks were delicious: blood-red **bardiwac** as well as light, sweet Rhenish
- He handed her a glass of **bardiwac**
- Beef dishes are made to complement the **bardiwacs**

Key Idea: meaning can be derived from language usage

Sub-symbolic Commonsense: Word2Vec

- **Grounded in Distributional Hypothesis [Harris,1954]: vector representations of language items**
- **i.e., embeddings, generated from a text corpus** [Mikolov+, 2013]



- Neural network trained on a prediction task
- Similar words appear in similar contexts and have similar vectors

Key Idea: neural models generate distributional vectors of a given dimension

Logical Reasoning with Common Sense:
Combining LTNs with Embeddings

LTNs and Commonsense

LTNs

- Learn from structured knowledge
- **Logic constants are vectors**
- Reasoning done combining fuzzy logics with neural networks

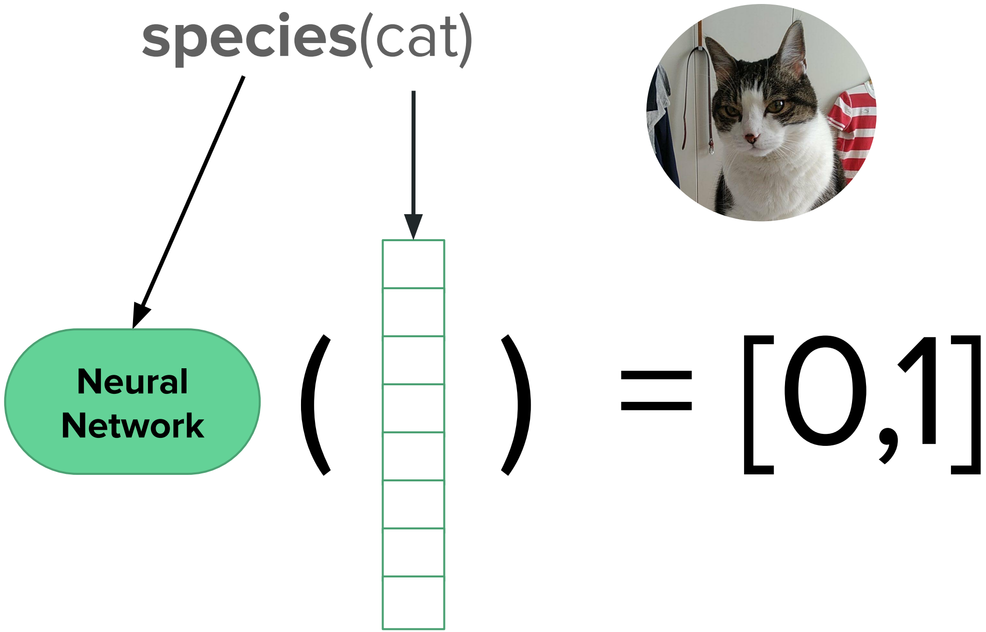
Knowledge Embeddings

- Learned from text
- Capture latent semantics
- **Entities are vectors**

We can use sub-symbolic representations as the constants in LTNs

Logic Tensor Networks: General Idea

No need to learn the representation of “dbr:cat”. We already have its **distributional entity vector.**



LTNs and Commonsense

Commonsense Knowledge

- Text annotated with entity linking
- Entities **dbr:cat** and **dbr:tiger** appear in similar contexts.
- **Entity Embeddings**:
- v(dbr:cat) ≈ v(dbr:tiger)

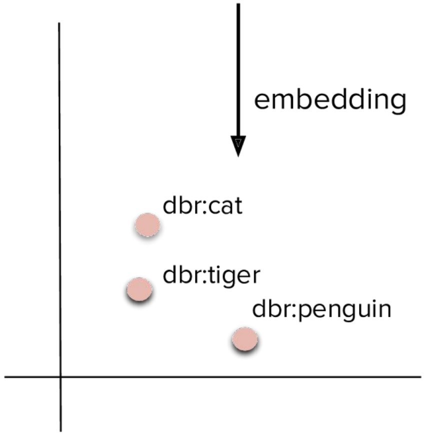
Axiomatic Knowledge

species(**dbr:cat**)
mammal(**dbr:tiger**)
bird(**dbr:penguin**)
[...]

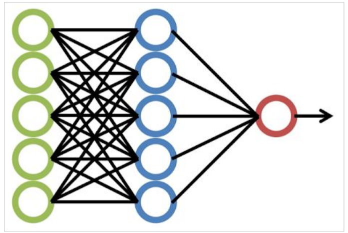
Instantiated atoms

∀ **x** (mammal(**x**) → animal(**x**))

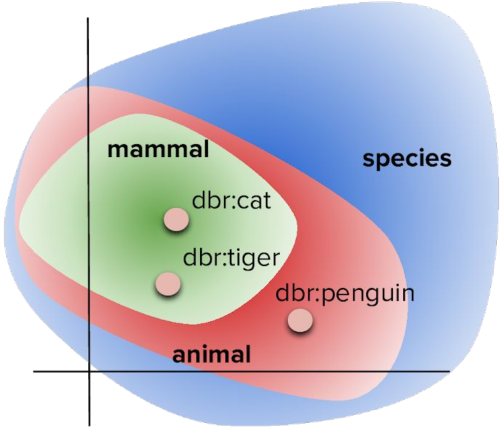
Universally quantified formulas



learning predicates with commonsense and axiomatic knowledge



LTN



After Training Inferences:
animal(**dbr:cat**) ?
∀ **x** (species(**x**) → animal(**x**)) ?

Sub-symbolic commonsense knowledge

Sub-symbolic commonsense knowledge with learned predicates

Experimental Evaluation

We created a simple KB with predicates about **species and sub-classes.**

Task: KB completion.

Inputs to LTNs:

- **Entity Embeddings** from Wikipedia
- **Axiomatic Knowledge (both inst. atoms and 22 quantified rules)**

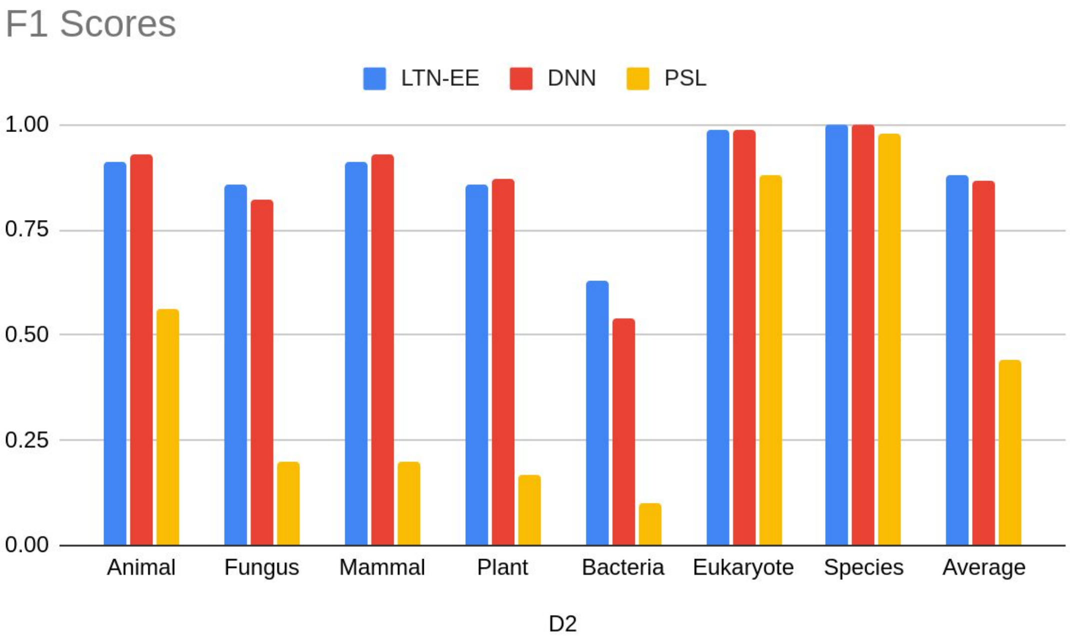
species(**cat**)
mammal(**tiger**)
bird(**penguin**)
∀**x** (mammal(**x**) → animal(**x**))

General Idea

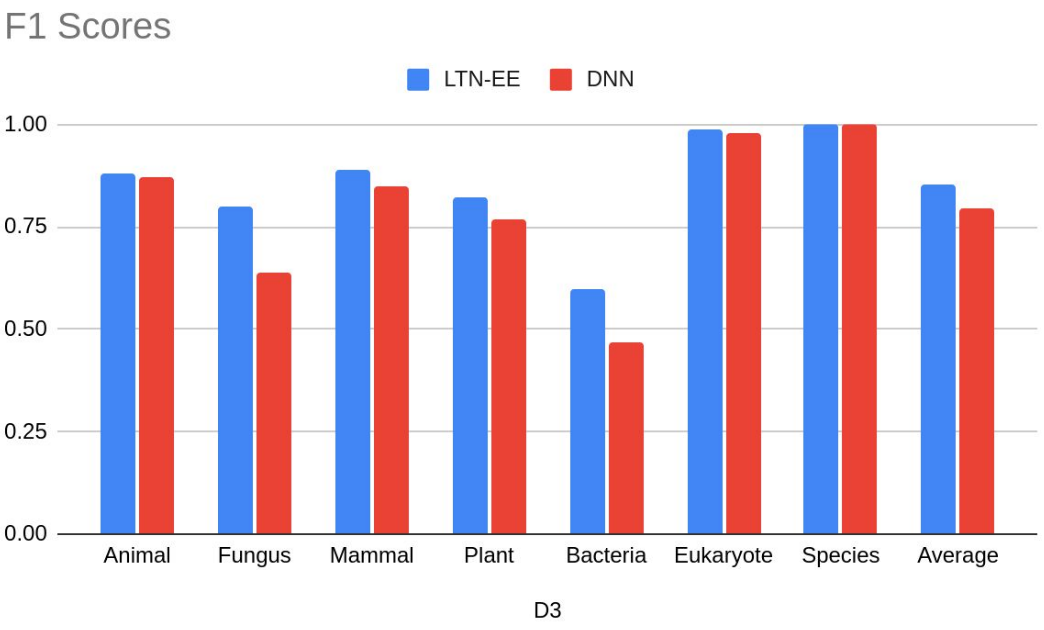
mammal(cat)

animal(cat)

Experiments: Standard Training/Testing



Experiments: Zero-Shot on Novel Entities



LTN: Testing New Predicates

We collected species and trained LTNs to recognize the classes of each species

Again: we can **logically explore** the vector space

We query the vector space with logical rules.

Axiom	Truth
$\forall x(species(x) \rightarrow animal(x))$	0
$\forall x(eukaryote(x) \rightarrow \neg bacteria(x))$	0.73
$\exists x(eukaryote(x) \wedge \neg plant(x))$	1

LTN: Testing New Predicates

We collected species and trained LTNs to recognize the classes of each species

Again: we can **logically explore** the vector space

We query the vector space with logical rules.

Axiom	Truth
$\forall x(species(x) \rightarrow animal(x))$	0
$\forall x(eukaryote(x) \rightarrow \neg bacteria(x))$	0.73
$\exists x(eukaryote(x) \wedge \neg plant(x))$	1

LTN: Testing New Predicates

We collected cities and people from Wikipedia and learned some predicates like **bornIn, locatedIn** and **nationality**

Again: we can **logically explore** the vector space

Axiom	**Truth**
$\forall x, y, z(nationality(x,y) \wedge locatedIn(z,y) \rightarrow bornIn(x,z))$	0.33
$\exists x(nationality(x, Canada) \wedge bornIn(x, Montreal))$	1
$\forall x(bornIn(x, New\ York) \rightarrow nationality(x, United\ States))$	0.88

Thank you!

Backup Slides

Terminological Recap: First Order Logic

A **constant** is an element of a domain (set) taken in consideration

S : {John, Paul, ...}
T : {BMW, Toyota, ...}

A **function** is a relation f: S → T between sets that associates to every element of a first set exactly one element of the second set.

car: T → S
car(Paul) = BMW

A **predicate** is a Boolean-valued function P: S → {1 (= True), 0 (=False)}.

male: S → {1, 0}
male(John) = 1

drives: S x T → {1, 0}
drives(Paul, BMW) = 1

Terminological Recap: First Order Logic

An **axiom** is a statement in a logical language:

$$R(a, b)$$

A **grounded axiom** contains grounded constants:

$$parentOf(John, Paul)$$

A **quantified axiom** is an axiom that contains quantified variables:

$$\forall\ x\ Human(x)$$

A **formula** is a combination of grounded and quantified axioms:

$$\forall\ x,y\ parentOf(x, y) \Rightarrow parentOf(y, x)$$

Terminological Recap: Operators

- **&** - conjunction: parent(Ann, Susan) & Female(Susan)
- **|** - disjunction: parent(Ann, Susan) | parent(Susan, Ann)
- **!** - negation: !Male(Susan)
- **=>** - implication: parent(Ann,Susan) => children(Susan,Ann)
- **∀** - forall: ∀x female(x) => human(x) *[forall x if x is a female than x is a human]*
- **∃** - exists: ∃x human(x) *[exists one human]*