Appunti Artificial Intelligence

CONTATTAMI:
- personale: gianluca[at]scarpellini.dev
- campus: g.scarpellini1[at]campus.unimib.it

PAPERS E MATERIALE

Slide:
- 1.IA1920 - 00 - Palmonari - Challenges in AI and this Course.pdf
- 2.IA1920 - 01.0 - Palmonari - Bandini Re-edit - History of AI.pdf
- 20190308-inter.pdf
- 20190309-env.pdf
- 3.IA1920 - 01.1 - Palmonari - Logics and AI.pdf
- 4.IA1920 - 01.2 - TextBook Selected - Simple Logic-based AI with Clauses - C5lect2.pdf
- 5.IA1920 - 01.3 - Palmonari and TextBook - FOL Datalog and Logic Programming.pdf
- 6.IA1920 - 01.4 - Palmonari - Applications - Transforming Big Data and the Web into Knowledge Bases.pdf
- 7.lect1.pdf
- 810 - Palmonari - Making Sense of Big Data + Data Linking - Part - 1.pdf
- AUTHOR_SLIDES_MAS_2nd_chapter03.pdf
- coordination1.pdf

Contents

Introduction (*30/09)*

Artificial intelligence is a intersection of many discipline (computer science, philosophy, biology, math, sociology, psychology, neuron science)
Human can acquire knowledge from experience, transmission and inference (generalizing and draw more conclusions)

Human-level AI
- *Human learn faster and without limited training "data"*
  - *Transfer learning, few-shot learning, combination of symbolic knowledge and learning*
- *Different humans' cognitive skills are highly connected and not learnt independently: we don't learn only one one specific task unlike machine that are focalized on one task*
  - *Transfer learning, multi-modal learning, multi-task learning, combinaton of analogical and logical reasoning, cognitively grounded architectures*
- *Human can imagine and reason about things and events that do not exist and at higher level of depth and abstraction*
  - ***Generative** models, imagination machines, **what-if question answering**, counterfactuals*
- *Humans learn, reason and acts*

Combining reasoning and learning → neuro-symbolic integration

History (01/10)

***Foundation** → 1956, but it have reference in the past*
Aristotele makes first attempts
***Turing test:** an interrogator has to make questions and must determine if the respondent is a machine or a human*

***First work on neural networks** → McCulloch and Pitt*
***Artificial neural networks** → learning by training (adjusting weights and thresholds)*

Physical Symbolic System Hypothesis

A physical symbolic system has necessary and sufficient instruments to perform general intelligent actions.

Manipulation of symbols

2 branch:
- One anti-logical, using methods similar to human reasoning
- *Logical*

***Cybernetics:** The idea was that the adaptation was a crucial aspect of artificial intelligence*
***Expert systems:** An expert system is a system that, on a determinate domain, has the same performance of a human expert*
***Semantic web**: the transformation of the web in*

Problem solving, reasoning, logic and AI
We have a problem that we want to solve with a computational approach, represent it, compute it in the space of the representation e give and output that is the solution of the problem
We want that the representation is close to the problem as possible and we have to consider the costs to reach the solution

In AI, more information could take to a better job but use more resources
Specify a problem is not easy and is very important
A symbol system can create, copy, modify and destroy symbols. To work with it we want

Interpretation *(07/10)*

Assign a value to all propositional variables. A model is an interpretation that satisfy constraint.
Representing knowledge, 2 main concepts:

Simple language: propositional definite clauses
- Atom: symbol starting with the lower causes
- Body: atom or in the form $b1 \wedge b2$ *where* $b1$ *and* $b2$ *are bodies*
- Knowledge base (KB): set of definite clauses

Propositional logic

Inference

Modus Pones

$$\frac{\alpha,\ \alpha \rightarrow \beta}{\beta}$$

Modus tollens

$$\frac{\neg\beta,\ \alpha \rightarrow \beta}{\neg\alpha}$$

First Order Logic

Definitions

- **Terms**: named for individuals:
  - b_obama, x, y, z, birthdate(b_obama), birthdate(x)
- **Predicates** to refer to relations among different individuals
  - PresidentOf(x,y,s,t)
- **Atomic formulas** to state that relations between individuals exist
  - PresidentOf(b_obama,usa,2009,2017)
- **Complex formulas** via connectives and quantifier
  - ∀x(∃y∃z,PresidentOf(x,usa,y,z) → USPresident(x))

Inference

FOL has *modus pones too:*

$$\frac{\forall x(\exists y \exists z, PresidentOf(x,usa,y,z) \rightarrow USPresident(x)) \quad PresidentOf(b\_obama,usa,2009,2017)}{USPresident(b\_obama)}$$

Interpretation

Some proposition will be always true or false, some are true in some interpretation and false in other.

- *A **model** of a set of clauses, is **an interpretation in which all the clauses are true***
- *A KB is true in **I**, if and only if every clause in KB is true in I*
- *If KB (knowledge base) is a set of clauses and g a conjunction of atoms, g is a logical consequence of KB, written $\models$, if g is true in very model of KB. There is no interpretation in which KB is true and g is false*

*A**xiomatization**: we are telling at the system what are the true clauses in the intended interpretation*

How can we compute consequences if we have a large number of propositions?
- *We can create a proof, a mechanically derivable demonstration that a formula logically follows from a knowledge base.*

Proof

Given a proof procedure, $\vdash$ *mean g con be derived from knowledge base KB*
Sound proof: *every time* $\vdash$ *implies* $\models$
Complete proof: *every time* $\models$ *implies* $\vdash$
Minimal model: *minimal set of formula of KB that let compute all the consequences*

Forward chaining

If $h \leftarrow b_1 \wedge \ldots \wedge b_n$ is a *clause* in KB, and each b_i can be derived, then h can be derived too **(per modus pones)**

Bottom-up procedure

$KB \vdash g$ if $g \in C$ at the end of this procedure:

$C := \{\}$;
repeat
  select clause "$h \leftarrow b_1 \wedge \ldots \wedge b_m$" in KB such that
    $b_i \in C$ for all i, and
    $h \notin C$;
  $C := C \cup \{h\}$
until no more clauses can be selected.

Fixed point

C is a **fixed point**

Let I be the interpretation in which every element of the fixed point is true and every other atom is false. **Then I is a model of KB.**
Proof: suppose $h \leftarrow b_1 \wedge \ldots \wedge b_n$ in KB is false in I. Then h is false and each b_i is true in I. Thus h can be added to C. **Contradiction** to C being the fixed point.

I is called a Minimal Model.

Completeness

If KB $\models g$ ***then KB*** $\vdash g$***.***

Suppose KB $\models g$*. Then* ***g*** *is true in all models of KB.*
- *Thus g is true in the minimal model.*
- *Thus g is in the fixed point.*
- *Thus g is generated by the bottom up algorithm.*
- *Thus KB* $\vdash g$

Datalog *(08/10)*

Assumptions

Query: *way to ask if a body is a logical consequence of the knowledge base*
2 possible answer:
- *An instance of the query, logical consequence of the KB*
- *No, if no instance is a logical consequence of the KB*

An **instance** of a clause is obtained by **uniformly substituting terms for variables in the clause**. *All occurrences of a particular variable are replaced by the same term.* The proof procedure extended for variables must account for the fact that a free variable in a clause means that any instance of the clause is true. A proof may have to use different instances of the same clause in a single proof. **The specification of what value is assigned to each variable is called a substitution.**
A **substitution** is a finite set of the form $\{V_1/t_1, \ldots, V_n/t_n\}$, where each Vi is a distinct variable and each ti is a term. The element Vi/ti is a binding for variable Vi. A substitution is in normal form if no V_i appears in any t_j.
The **application** of a substitution $\sigma = \{V_1/t_1, \ldots, V_n/t_n\}$ to expression e, written $e\sigma$, is an expression that is like the original expression e but with every occurrence of V_i in e replaced by the corresponding t_i.

A substitution $\sigma = \{V_1/t_1, \ldots, V_n/t_n\}$ is a unifier of expressions e_1 and e_2 if $e_1\sigma$ is identical to $e_2\sigma$.

Knowledge*:* described in terms of *individuals* and *relations* among individuals. Definite and positive statements
Environment: static

Syntax

- *A **variable** starts with upper-case letter.*
- *A **constant** starts with lower-case letter or is a sequence of digits (numeral).*
- *A **predicate** symbol starts with lower-case letter.*
- *A **term** is either a variable or a constant.*
- *An **atomic** symbol (atom) is of the form p or $p(t_1, \ldots, t_n)$ where p is a predicate symbol and t_i are terms.*
- *A **definite clause** is either an atomic symbol (a fact) or of the form: $a \leftarrow b_1 \wedge \ldots \wedge b_m$, where **h** is the head and b_i are atomic symbols composing **body***
- ***Query** is of the form $?b_1 \wedge \ldots \wedge b_m$*
- ***Knowledge base** is a set of definite clauses.*

Semantics

- *A **semantics** specifies the meaning of sentences in the language.*
- *An **interpretation** specifies:*
  - *What objects (individuals) are in the world*
  - *the correspondence between symbols in the computer and objects & relations in world*
    - ***Constants** denote **individuals***
    - ***Predicate symbols** denote **relations***

Example 12.2 The following is a knowledge base:

$$in(kim, R) \leftarrow teaches(kim, cs422) \wedge in(cs422, R).$$
$$grandfather(sam, X) \leftarrow father(sam, Y) \wedge parent(Y, X).$$
$$slithy(toves) \leftarrow mimsy \wedge borogroves \wedge outgrabe(mome, Raths).$$

From context, *kim*, *cs*422, *sam*, *toves*, and *mome* are constants; *in*, *teaches*, *grandfather*, *father*, *parent*, *slithy*, *mimsy*, *borogroves*, and *outgrabe* are predicate symbols and X, Y, and *Raths* are variables.

The first two clauses about Kim and Sam may make some intuitive sense even though we have not explicitly provided any formal specification for the meaning of sentences of the definite clause language. However, regardless of the mnemonic names' suggestiveness, as far as the computer is concerned, the first two clauses have no more meaning than the third. Meaning is provided only by virtue of a semantics.

Logical Consequence

*Atom **g** is a logical consequence of KB if and only if:*
- *g is a fact in KB, or*
- *there is a rule* $g \leftarrow b_1 \wedge \ldots \wedge b_k$ *in KB such that each* b_i *is a logical consequence of KB*

Interpretation

*An interpretation is a **triple*** $I = \langle D, \phi, \pi \rangle$ *where*
- *D, the **domain**, is a nonempty set. Elements of D are individuals.*
- ϕ *is a mapping that assigns to each constant an element of D. Constant c denotes individual* $\phi(c)$.
- π *is a mapping that assigns to each n-ary predicate symbol a relation: a function from* D^n *into {TRUE, FALSE}.*

We map the KB to real world concepts.

A ground atom is either true or false in an interpretation. Atom p(t1,…,tn) is true in I if

$$\pi(p)(\langle t_1, \ldots t_n \rangle) = T$$

User vs Computer interpretation

1. *Choose a task domain:* ***intended interpretation (computer doesn't have an intender interpretation)***
2. *Associate constants with individuals you want to name.*
3. *For each relation you want to represent, associate a predicate symbol in the language.*
4. *Tell the system clauses that are true in the intended interpretation:* ***axiomatizing*** *the domain.*
5. *Ask questions about the intended interpretation.*
6. *If KB* $\models g$*, then g must be true in the intended interpretation. (logical consequence of KB)*

Variables

Example 12.6 The clause

$$part\_of(X, Y) \leftarrow in(X, Y).$$

is false in the interpretation of Example 12.4 (page 497), because under the variable assignment with X denoting Kim and Y denoting Room 123, the clause's body is true and the clause's head is false.

The clause

$$in(X, Y) \leftarrow part\_of(Z, Y) \wedge in(X, Z).$$

is true, because in all variable assignments where the body is true, the head is also true.

- *Variables are universally quantified in the scope of a clause.*
- *A variable assignment is a function from variables into the domain.*
- *Given an **interpretation** and **a variable assignment**, each term denotes an individual and **each clause is either true or false**.*
- *A **clause** containing variables **is true in an interpretation if it is true for all variable assignments***

Debugging

To debug an answer g that is false in the intended interpretation:
- *If g is a fact in KB, the fact is wrong, we have to remove it*
- *If g was proved using the rule*

Problem solving in AI – KR system (Knowledge Representation System)
We want to represent and encode the problem in the system. The solution will be formulas that are
logical consequences of KB

Limitations

Suppose you had a database using the relation:
enrolled(S, C)

which is true when student S is enrolled in course C.

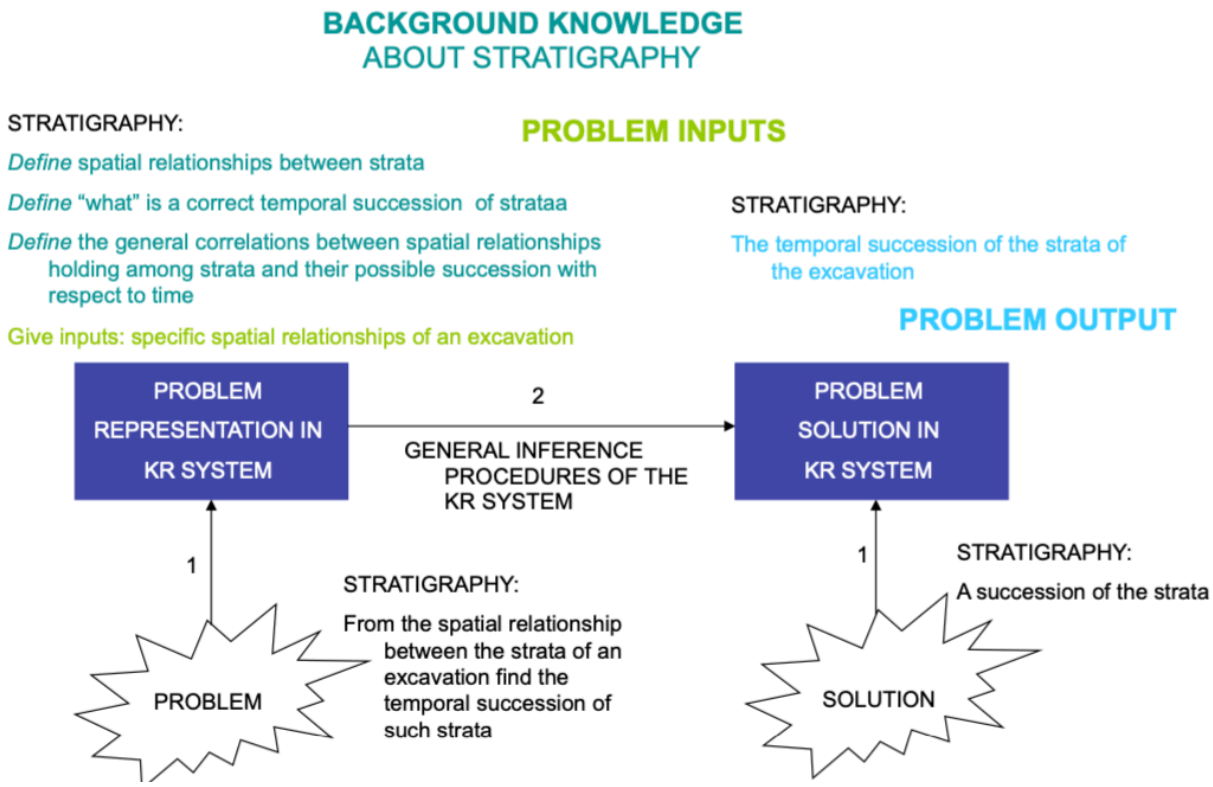
You can't define the relation:
empty course(C)

which is true when course C has no students enrolled in it. This is because empty course(C) doesn't logically follow from a set of enrolled relations. There are always models where someone is enrolled in a course!

Functions

*A knowledge base consisting of clauses with function symbols can compute any computable function. Thus, a knowledge base can be interpreted as a program, called a **logic program**.*

Example: ***Stratigraphy***

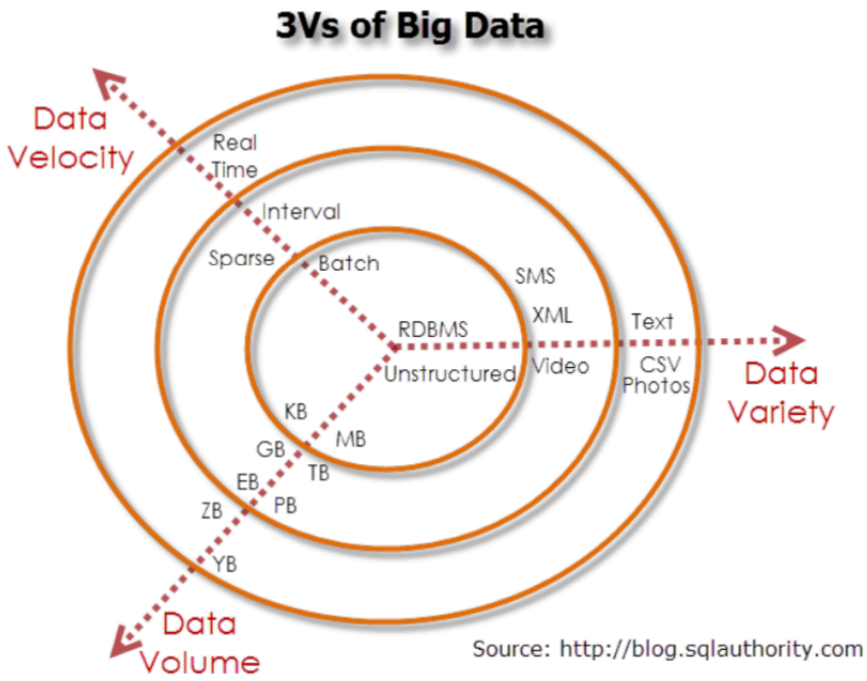


Spatial to temporal mapping

- *Define spatial relation between strata's*
- *Define "what" is a correct temporal succession of strata*
- *Define general correlation between spatial relationship*

```
posteriorTo(Y,W) :- leanOn(X,Y),leanOn(X,Z),cover(Y,Z),cover(X,W),leanOn(W,Z).

posteriorTo(X,W) :- leanOn(X,Y),leanOn(Z,W),cover(X,Z),cover(Y,W).
posteriorTo(Y,Z) :- leanOn(X,Y),leanOn(Z,W),cover(X,Z),cover(Y,W).

posteriorTo(Z,Y) :- contemporary(X,Y),posteriorTo(Z,X).
posteriorTo(Z,X) :- contemporary(X,Y),posteriorTo(Z,Y).

dirPostTo(Z,Y) :- equalTo(X,Y),cover(Z,X).
dirPostTo(Z,X) :- equalTo(X,Y),cover(Z,Y).

-contemporary(Z,Y) :- equalTo(X,Y),posteriorTo(X,Z).
-contemporary(Z,X) :- equalTo(X,Y),posteriorTo(Y,Z).

contemporary(X,Y) v posteriorTo(X,Y) v posteriorTo(Y,X) :- us(X),us(Y),not posteriorTo(X,Y),not -posterior(X,Y).
```

Big data *(08/10)*



Web data

1. *Semi-structured data (easy)*
   a. *Organized in attributes*
   b. *Well defined schema, follow some order*
2. *Semi-structured data (hard)*
3. *Unstructured data (Natural language text)*

Interpreting and connecting web data

- Experience to locations: To make sense of unstructured information you need well (well) structured information
- Factual information in search results
- Connecting people to locations

Knowledge graphs

To store entities connected by relations

Definition

A KG is a quintuple $< E, L, T, P, A >$
- E = set of entity id symbols
- L = set of literals
- T = set of type symbols
- P = set of relation symbols
- A = set of axioms in a language (usually subset of first order logic)

A-facts
- *Types of entities and literals*
- *Relation between entities*

A-general-axioms:
- *they depends on the language*
- *The minimal requirements is that we must be able to specify the relations between the types*

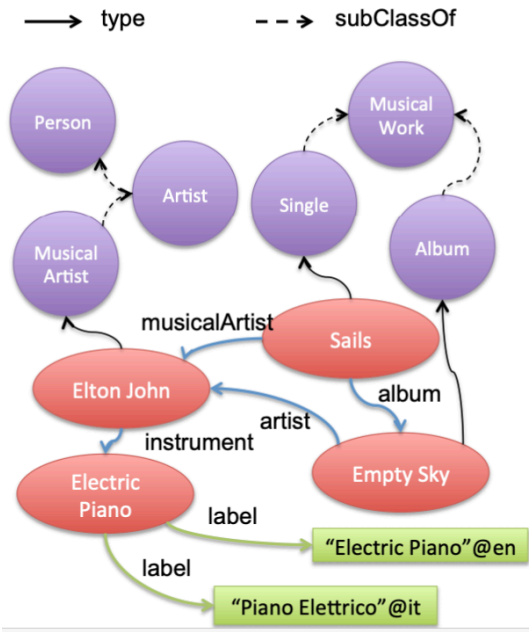
Applications

Common characteristics of the knowledge graphs.

	Data model	Size of the graph	Development stage
Microsoft	The types of entities, relations, and attributes in the graph are defined in an ontology.	~2 billion primary entities, ~55 billion facts	Actively used in products
Google	Strongly typed entities, relations with domain and range inference	1 billion entities, 70 billion assertions	Actively used in products
Facebook	All of the attributes and relations are structured and strongly typed, and optionally indexed to enable efficient retrieval, search, and traversal.	~50 million primary entities, ~500 million assertions	Actively used in products
eBay	Entities and relation, well-structured and strongly typed	Expect around 100 million products, >1 billion triples	Early stages of development and deployment
IBM	Entities and relations with evidence information associated with them.	Various sizes. Proven on scales documents >100 million, relationships >5 billion, entities >100 million	Actively used in products and by clients

RDF

RDF is a data model for representing data on the Web based on:
- **Triples** – basic unit to organise the information
- **Directed (labeled) graphs** – sets of triples
- **URIs** – unique resource identifiers



Agents *(14/10)*

Ai: synthesis and analysis of computation agents that act intelligently

*An agent is something that act in an **environment**, and it acts **intelligently** if:*
- *Its actions are appropriate to the **goal** and the **circumstance***
- *It is **flexible** to changing environment and goals*
- *It **learns** from experience*
- *It makes **appropriate choices** given perceptual and computational information*

An agent makes actions in the environment in accord to its abilities, goals/preferences, prior knowledge, stimuli (given by environment itself) and past experiences.

---

2 approaches

- ***Scientific goal:** understand the principles that make intelligent behaviour possible in natural or artificial system*
- ***Engineering goal**: design useful, intelligent artefacts*

Goals	Resources	Abilities	Type
Compatible	Sufficient	Sufficient	Independence
Compatible	Sufficient	Insufficient	Simple collaboration
Compatible	Insufficient	Sufficient	Obstruction
Compatible	Insufficient	Insufficient	Coordinated collaboration
Incompatible	Sufficient	Sufficient	Pure individual competition
Incompatible	Sufficient	Insufficient	Pure collective competition
Incompatible	Insufficient	Sufficient	Individual conflicts over resources
Incompatible	Insufficient	Insufficient	Collective conflicts on resources

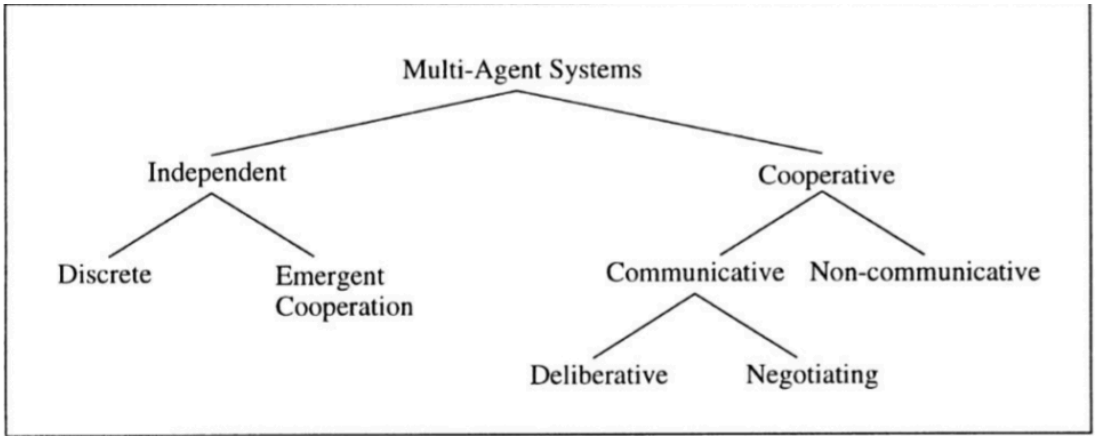
---

Multi-agent systems as distributed systems

Multi agent system can be seen as distributed system heterogenous, designed independently by different people. Protocols define how agents should **communicate:**
- **Modular**
- **Reusable**
- **Interoperability**

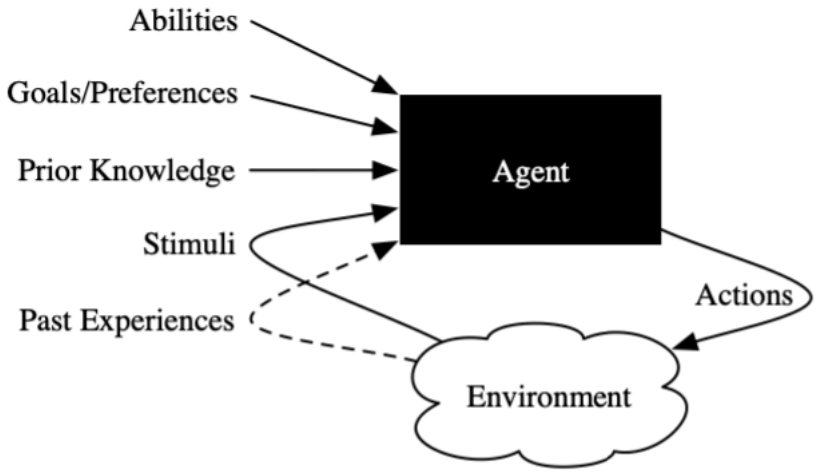
Traditional distributed systems:
- ignore autonomy and heterogeneity
- Implement low level messages

Autonomy: protocol do not over-constraint agents interaction



Agent interaction models *(15/10)*

Interaction: "an interaction occurs when two ore more agents are brought to a dynamic relationship through as set of reciprocal actions".



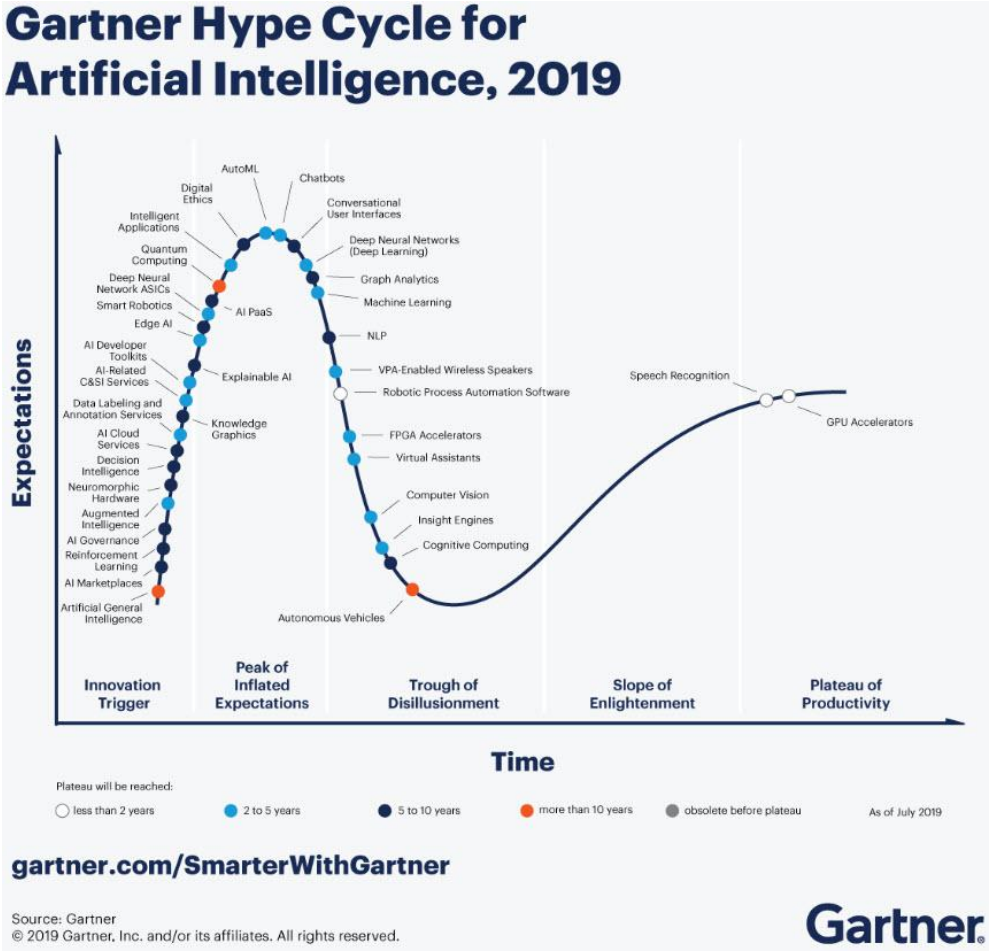
This form some communication path, is a point to point interaction
The interaction is not necessary the principal method in a communication and is required when it is not possible for a single agent to carry out its tasks.

Interaction model taxonomy

Direct interaction:
- *With a-priori acquaintance*
- *Agent discovery through middle agents*
- *Middle agents & acquaintance models*

Indirect interaction:
- *Guided/mediated by artefacts*
- *Spatially founded interaction*

Communicate act theory



Goes beyond traditional logic.

Different type of messages:
- informations: (e.g. I inform you about something)
- Directives (e.g. I request you to do something)
- Commissive (e.g. I promise …)

Agents communication

Multi-agent systems are distributed systems and they can be autonomous (independently acting) and heterogeneous (independently designed). The communication with each other happen with protocols that define how the agents ought to communicate and communities of practice define appropriate protocols. Typical distributed computing, ignore autonomy and heterogeneity. **Autonomy** is the facts that the agents are free to act as they please, so we need protocols that not over-constraint an agent's interactions.

Coordination

- Managing dependencies between activities
- Coordination + computation: separated components

- Parallel computing is a *consequence of coordination*

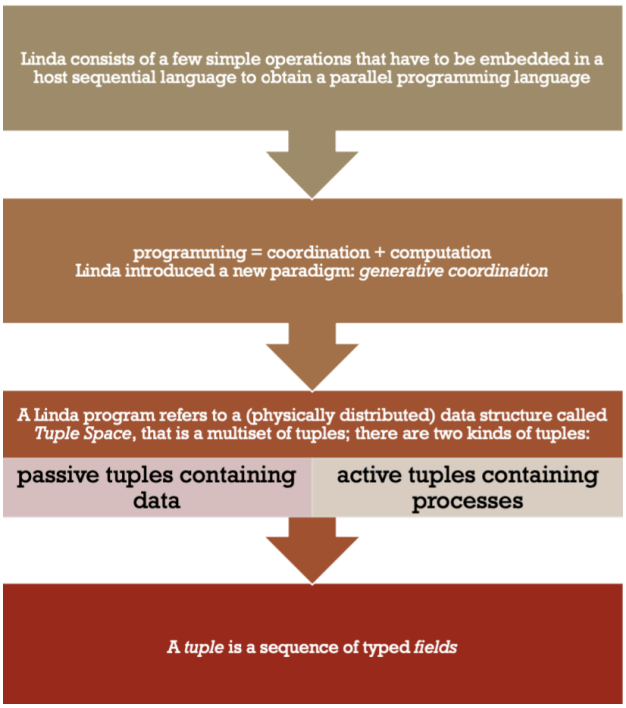
Distributed data structures: Linda

A **distributed data structure** is logically separated from process that can manipulate it; in Linda, for instance, the distorted data structure is contained in a Tuple Space, that is a multisite of tuples.Processes produce/consume tuples and create other processes

Linda is the most known language that provides a distrvugted data structure; other language that offer distributed data structures are Orca, Shared Prolog, Gamma

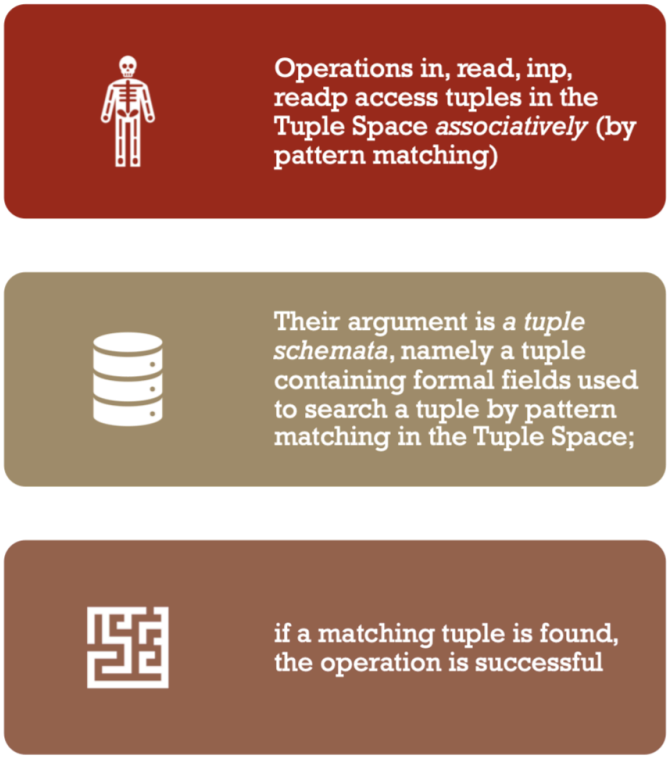
Tuple spaces (global computing environment). Include both data and agents. Linda specifies operation as the tuple space (READ, EVAL, IN, OUT).

Agents cannot communicate directly. All tuples are created by agents (OUT); they are atomic data that can be read (RD) or consume (IN). An agent an also create a new agent (EVAL) that when terminates becomes a data tuple.



Operations in Depth

- out(t) puts a new passive tuple in the Tuple Space; continues immediately
- eval(t) puts a new agent in the Tuple Space (each field containing a function to be computed starts a process); the caller agent continues immediately; when all active fields terminate the tuple becomes passive
- in(t) looks for a passive tuple in the Tuple Space; if not fond the agent suspends; when found, reads and deletes it
- rd(t) looks for a passive tuple in the Tuple Space; if not fond the agent suspends; when found, reads it
- inp(t) looks for a passive tuple in the Tuple; if not fond the agent return FALSE; if found, deletes it and return TRUE
- rdp(t) looks for a passive tuple in the Tuple if fond, copies it and return TRUE; if not, return FALSE



Coordination model (Tuple Space)

- Uncoupling of agents
- Associative addressing
- Not determinism
- Separation of concerns

Computation deals with *algorithms;* ***coordination*** *deals* with *architectures*

Definition
A **coordination model** is a triple (E, M, L) where:
- E: coordinative entities (active agents involved)
- M: coordinating media (media of the coordination, e.g. channels)
- L: coordination laws (laws for the coordinative entities actions)

Define mechanism for **communication, synchronisation, distribution** and **currency control**

LINDA EXAMPLES

Example:

```
out("string", 10.1, 24, "anothe:
        string")

real f; int i;
rd("string", ?f, ?i, "another
        string")
```

succeeds

```
in("string", ?f, ?i, "another
        string")
```

succeeds

```
rd("string", ?f, ?i, "another
        string")
```

does NOT succeed

AI: Hype? *(22/10)*

Every tasks could be tackled with different approaches: RL, DL, traditional ML. It's best practice to actually mixed together different approaches.

RL *(22/10)*

Applications

- ML
- CS
- Economic
- Math
- Psychology
- Engineering

Characteristics

No supervision during training; however, we define a *reward signal.* Data are correlated sequence, and agent's actions affect the subsequent data it receives. In addition, feedback is **delayed** and **time matters.**

Agent and environment

Agents act in an environment. At each step, the agent:

- execute an action A_t
- Receives and observation O_t
- Receives a scalar reward R_t

The environment:

- Receives an action A_t
- Emits observation O_{t+1}
- Emits scalar reward R_{t+1}

t increments at **env step**

What's a reward?

Scalar feedback signal. Indicates how well agent is doing at step t
The agent-s job is to *maximise* cumulative reward

```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  env.render()
  action = env.action_space.sample() # your agent here (this takes random action
s)
  observation, reward, done, info = env.step(action)

  if done:
    observation = env.reset()
env.close()
```

All **goals** can be described by the **maximisation** of **expected cumulative reward.**

Discrete case: start from state S and move to the next (especially when states are not enumerable). You need to put a signal to reward on each branch of a tree. You **explore the tree of states.**

In real environment, rewards **change.**

Components:
- Environment
- Action
- Reward
- Algorithm

Sequential decision making process

Goal: select actions to maximise total future reward

- Actions may have **long term consequence:** short-term reward is easy found but agent should project its actions for a highest future reward
- Reward may be **delayed**

History and state

History: sequence of observations, actions and rewards

State: is the information used to determine what happens next. Formally, state is a function of the history: $S_t = f(H_t)$. In fact, the current state is the result of a history (if the agent is the only subject influencing the environment).

Environment state

Environment state is only partially known by the agent: is virtually impossible for an agent to perceive everything. Environment state can also be any function of the history, and in general you don't have an analytical function.

Information state

Sometimes the state can be modelled as a **markovian process** (the state at time t depends only on state t-1, or $P(S_t \mid S_{t-1}) = P(S_t \mid S_{t-1}, \ldots, P_1)$.

You can make use of dynamic programming techniques, you can translate the learning process to a dynamic programming process (planning, not always applicable)

Partially observability

Agent *indirectly* observes environment:
- A robot doesn't known camera absolution location
- …

Terminology

An RL agent may include one or more of these components:
- **Policy**: agent's behaviour function
- **Value function**: how good is each state / actions
- **Model**: agent's representation of the environment

You map the rewards to the environment, to estimate the quality of the model. Sometimes policy is learning behaviour.

Policy: agent's behaviour. Function that select action from a state. Can be deterministic or probabilistic. It maps a state into an action.

Value function: evaluation of *feature rewards* starting from a given state. To select between actions. Estimate of a cumulative rewards for future states weighted given the current state. **Past is irrelevant. If** you have a model for the environment (e.g. chess), you can have a objective definition of rewards (e.g. position of pieces, number of pieces, …). The weight is a parameter (called *discount*) to evaluate future rewards.

Model: from an agent perspective, it's a way to forecasting the next environment state given the current state and and an action. Generally, you have the probability distribution

Examples

Regulate traffic light given **history**
Maze: right policy to escape a maze. You can define an algorithm to find a policy for a generic maze. You can define *exhaustibly.* You cannot do this in a large state space.

Categorisation

- Value Base: value function
- Policy base: policy and no value function
- Actor critic: police and value function

- Model free: no knowledge of the consequence on the environment
- Model based: policy/value functions **and** model

It's important to understand the **right approach.**

Q-learning

Q-learning uses temporal differences to estimate the value of $Q*(s, a)$. In Q-learning, the agent maintains a table of Q[S, A], where S is the set of states and A is the set of actions. Q[s, a] represents its current estimate of $Q*(s, a)$. An experience [s,a,r,s'] provides one data point for the value of Q(s, a). The data point is that the agent received the future value of

$$Q[s,a] \leftarrow Q[s,a] + \alpha \left(r + \gamma \max_{a'} Q[s',a'] - Q[s,a] \right)$$

Q-learning learns an optimal policy no matter which policy the agent is actually following (i.e., which action a it selects for any state s) as long as there is no bound on the number of times it tries an action in any state (i.e., it does not always do the same subset of actions in a state). Because it learns an opti mal policy no matter which policy it is carrying out, it is called an **off-policy method.**

Exploration vs exploitation

The Q-learning algorithm does not specify what the agent should actually do.
The agent learns a Q-function that can be used to determine an optimal action. There are two things that are useful for the agent to do:

- exploit the knowledge that it has found for the current state s by doing one of the actions a that maximises Q[s, a]
- explore in order to build a better estimate of the optimal Q-function. That is, it should select a different action from the one that it currently thinks is best.

The ϵ-greedy strategy is to select the greedy action (one that maximises Q[s, a]) all but ϵ of the time and to select a random action ϵ of the time, where $0 \leq \epsilon \leq 1$. It is possible to change through time. Intuitively, early in the life of the agent it should select a more random strategy to encourage initial exploration and, as time progresses, it should act more greedily. One problem with an ϵ -greedy strategy is that it treats all of the actions, apart from the best action, equivalently.

Reasoning agents *(28/10)*

In brief

A **URI** (a Uniform Resource Identifier) is used to uniquely identify a resource. A resource is anything that can be uniquely identified. A URI is a string that refers to a resource, such as a web page, a person, or a corporation. Often URIs use the syntax of web addresses.
RDF (the Resource Description Framework) is a language built on XML, providing individual-property-value triples.
RDF-S (RDF Schema) lets you define resources (and so also properties) in terms of other resources (e.g., using subClassOf). RDF-S also lets you restrict the domain and range of properties and provides containers (sets, sequences, and alternatives – one of which must be true). RDF allows sentences in its own language to be reified. This means that it can represent arbitrary logical formulas and so is not decidable in general. Undecidability is not necessarily a bad thing; it just means that you cannot put a bound on the time a computation may take. Simple logic programs with function symbols and virtually all programming languages are undecidable.
OWL (the Web Ontology Language) is an ontology language for the World Wide Web. It defines some classes and properties with a fixed inter pretation that can be used for describing classes, properties, and individuals. It has built-in mechanisms for equality of individuals, classes, and properties, in addition to restricting domains and ranges of properties and other restrictions on properties (e.g., transitivity, cardinality).

RDF: URified Graphs and Triples

RDF for describing communities
RDF can be used to describe persons, communities, online accounts. These properties and terms are defined in the **Friend of a Friend (FOAF) project.**

Ontologies

We're interested in models for a domain. Depending on the task, we can refer to ontology with different approaches.

Tesauri

Lexical ontologies. Not so popular anymore. Defines a dictionary for a language with focus on word and relationship between words.

Taxonomies

Classification structure for objects of a given domain. Trees with an hierarchy.
ex. Tree of

Ontologies as semantic technologies

More expressiveness leads to undecidability. We need a language to reason easily. The **primary purpose of an ontology** is to document what the symbols mean – the mapping between symbols (in a computer) and concepts (in someone's head).
The secondary purpose, achieved by the use of axioms, is to **allow inference or to determine that some combination of values is inconsistent.**

RDF Schema

More semantic to plain RDF
With RDFS we can model taxonomies

SubClasses: rules defined in RDFs to define hierarchies of concepts
Properties of properties, or *subPropertyOf*

	If S contains:	then S RDFS entails recognizing D:
rdfs1	any IRI aaa in D	aaa `rdf:type rdfs:Datatype .`
rdfs2	aaa `rdfs:domain` xxx . yyy aaa zzz .	yyy `rdf:type` xxx .
rdfs3	aaa `rdfs:range` xxx . yyy aaa zzz .	zzz `rdf:type` xxx .
rdfs4a	xxx aaa yyy .	xxx `rdf:type rdfs:Resource .`
rdfs4b	xxx aaa yyy.	yyy `rdf:type rdfs:Resource .`
rdfs5	xxx `rdfs:subPropertyOf` yyy . yyy `rdfs:subPropertyOf` zzz .	xxx `rdfs:subPropertyOf` zzz .
rdfs6	xxx `rdf:type rdf:Property .`	xxx `rdfs:subPropertyOf` xxx .
rdfs7	aaa `rdfs:subPropertyOf` bbb . xxx aaa yyy .	xxx bbb yyy .
rdfs8	xxx `rdf:type rdfs:Class .`	xxx `rdfs:subClassOf rdfs:Resource .`
rdfs9	xxx `rdfs:subClassOf` yyy . zzz `rdf:type` xxx .	zzz `rdf:type` yyy .
rdfs10	xxx `rdf:type rdfs:Class .`	xxx `rdfs:subClassOf` xxx .
rdfs11	xxx `rdfs:subClassOf` yyy . yyy `rdfs:subClassOf` zzz .	xxx `rdfs:subClassOf` zzz .
rdfs12	xxx `rdf:type rdfs:ContainerMembershipProperty .`	xxx `rdfs:subPropertyOf rdfs:member .`
rdfs13	xxx `rdf:type rdfs:Datatype .`	xxx `rdfs:subClassOf rdfs:Literal .`

Minimal reasoning

We can define a more general relationship then hierarchy. In fact, we define a POSET, a partial order relationship among entities.

SPARQL

Sparql was originally define only for **simple entailment (pattern matching).** There are incremental levels of entailment. We can complete the knowledge in a knowledge graph using **inference.**

Note:
We can generate inconsistency in the data graph if we define properties the wrong way. In fact, we get a contradiction.

Example: property is_member defined on a domain Groups. Then, an artist is also a Group!

We need a semantic construction to specify that 2 sets are disjoint (in OWL we can, in RDFs we cannot).

More inferences could be entailed, however the inference rules must be implemented. If we want a more semantical language, to define ontologies for example, we use OWL

OWL

OWL2 EL: limited to basic classification

OWL2 QL: designed to be translatable to relational database querying
OWL2 RL: designed to be efficiently implementable in rule-based systems.

.
OWL does not make the unique names assumption (page 534); two names do not necessarily denote different individuals or different classes. It also **does not make the complete knowledge assumption** (page 193); it does not assume that all relevant facts have been stated.

Axioms

Axioms declare general statement about concepts.

- Sub-class: relationship (like RDFs)

2 individuals are the same using the *sameAs* relationship. We distinguishes between 2 types of properties:
- **ObjectProperties**: *resources* as values
- **DatatypeProperties**: *literals* as values

If you define equivalence, you can query one knowledge graph with language of another one.

In OWL we can define important properties:

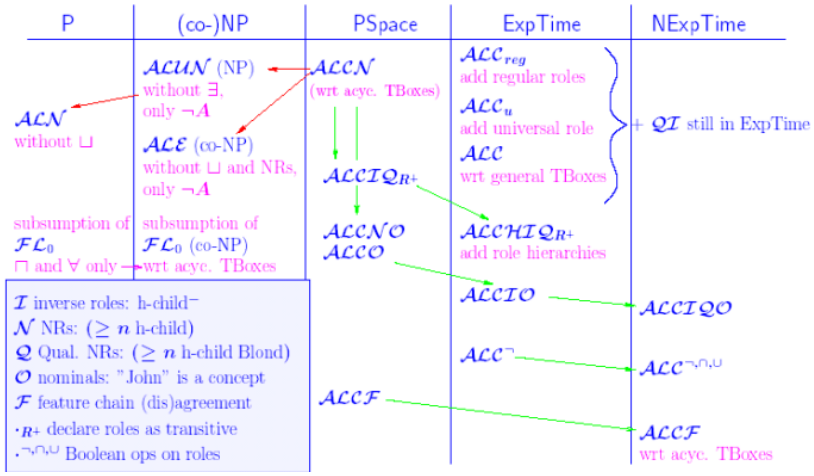
- Symmetry
- Transitivity
- Inverse
- Functional: a property is *functional* if it behaves like a function (at most a value)
- Inverse Functional

A tree is computationally efficient to represents hierarchies (ex. Spatial relationships, music genres, …). Inverse functional are useful to model these aspects.

Description logic *(29/10)*

Description logic is a subset of first order logic. Conceptual language, definition of complex concepts. We can model the concepts in a domain with a **modular** approach.

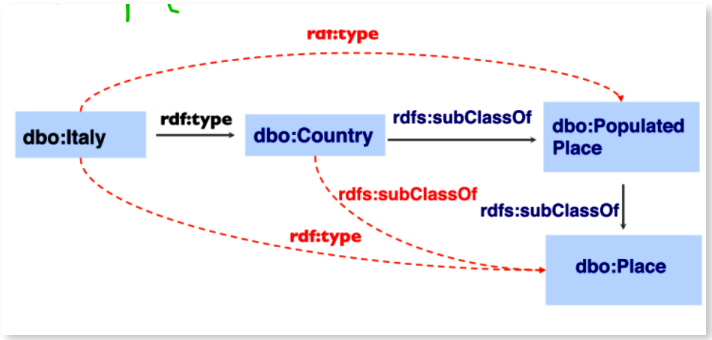
Taxonomy of DL



Grammar

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	¬Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	⩽1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	⩾2hasChild	$\exists^{\geqslant n} y.P(x,y)$

Limitation of RDFs



- No localised range and domain constraints
- No cardinality
- No transitive, inverse or symmetrical properties
- No Disjoint
- No boolean combinations: if we want to generalise association using conditions, we cannot do that in RDFs. We cannot join domains together, …

OWL

OWL is based on description logic, which allows to describe relationships among entities. It's a family of logics with incremental complexity.

It introduces:
- equivalent classes
- Disjointness of classes
- Equivalent of individuals
- Difference between individuals
- Object properties
- Datatype properties
- Equivalent properties

Can define properties characteristics: symmetry, transitivity, inverse, functional, inverse

OWL2 vs OWL1

Building blocks are the same. We have also backward compatibility. There're technical differences. Owl is the standard for representing ontologies. In Datalog, we focus on rules. In description logics instead there're few axioms. We focus on **definitions** of complex contexts. We model a context (what is a person, what is a wheel, …). Description logics became so popular because are **very modular.** In description logics we can define modules.

We use $SHIQ$ (equivalent to $SHOIN(D_n)$ DL). Owl benefit from DL research.
- Semantics
- Reasoning algorithms
- Implanted systems

OWL Syntax	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor

Description logic-bases systems

KB has **schema** and **data.**
The schema (also called T-box, or **Terminological** Box) define the meaning of your terminology (entities, relationships, …). We can also define complex classes (e.x. people with **at most** one child).

The data (**assertional** box) define concrete entities (data) using the previous define schema. We can say that an individual is member of a class or a tuple of individuals are member of a relationship.

Schema and data are separated. Data are extracted from different sources and mapped into our defined schema. Reasoning on cardinalities is impossibile in logic programming, because you have to quantify on the values and counts using functions (bad supported in Datalog).

We can build an infinite number of classes with a unique semantic. In ML and DL **composing class** isn't natural. Logic Tensor Network is trying to join ML and description logic.

Description logic can be mapped to First Order Logic or Modal Logics. Given a Description logic language, it's a subsample of first order logic. They have the same expressiveness.

Designing

An ontology does not specify the individuals not known at design time. For example, an ontology of buildings would typically not include actual buildings. **An ontology would specify those individuals that are fixed and should be shared, such as the days of the week, or colors.**

Description logic tasks *(04/11)*

Basic inference tasks

- Knowledge is **correct, if capture the intuitions**
- Knowledge is **minimally redundant:** two classes believed to not be equivalent are actually equivalent
- Knowledge is **meaningful**: knowledge base is satisfiable

If there's incoherence in KB happens if a concept is proved to be subclasses of **nothing (means isn't able to have any instance).**

There are also tools to **debug the KB**

Querying: classify instance of given class or relation

These problems are solved by optimal

Interpretation

- An interpretation $\mathcal{I}$ satisfies (models) an axiom A ($\mathcal{I} \models$ A):
  - $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ $\quad\quad$ $\mathcal{I} \models C \equiv D$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$
  - $\mathcal{I} \models R \sqsubseteq S$ iff $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ $\quad\quad$ $\mathcal{I} \models R \equiv S$ iff $R^{\mathcal{I}} = S^{\mathcal{I}}$
  - $\mathcal{I} \models x \in D$ iff $x^{\mathcal{I}} \in D^{\mathcal{I}}$
  - $\mathcal{I} \models \langle x,y \rangle \in R$ iff $(x^{\mathcal{I}},y^{\mathcal{I}}) \in R^{\mathcal{I}}$
  - $\mathcal{I} \models R^{+} \sqsubseteq R$ iff $(R^{\mathcal{I}})^{+} \subseteq R^{\mathcal{I}}$
- $\mathcal{I}$ satisfies a Tbox $\mathcal{T}$ ($\mathcal{I} \models \mathcal{T}$) iff $\mathcal{I}$ satisfies every axiom A in $\mathcal{T}$
- $\mathcal{I}$ satisfies an Abox $\mathcal{A}$ ($\mathcal{I} \models \mathcal{A}$) iff $\mathcal{I}$ satisfies every axiom A in $\mathcal{A}$
- $\mathcal{I}$ satisfies an KB $\mathcal{K}$ ($\mathcal{I} \models \mathcal{K}$) iff $\mathcal{I}$ satisfies both $\mathcal{T}$ and $\mathcal{A}$

Assumptions

Unique Name assumption (not default in DL): different names refer to different entities
Open world: if a fact isn't inferred by a KB, doesn't mean it's false

DL Reasoning

A concept C is subclass of D iff $C \cap \neg D$ is **unsatisfiable**

Construct a satisfiability formula (a form of concept normal form) and then apply tableau techniques: build a tree-model and apply expansion rules.

Some rules are **nondeterministic** (Godel theorem)

C is satisfiable iff the tree built is fully expanse and **no clash occurred.**

OWL Reasoning

Semantics: RDF_based (use rdf) vs DL

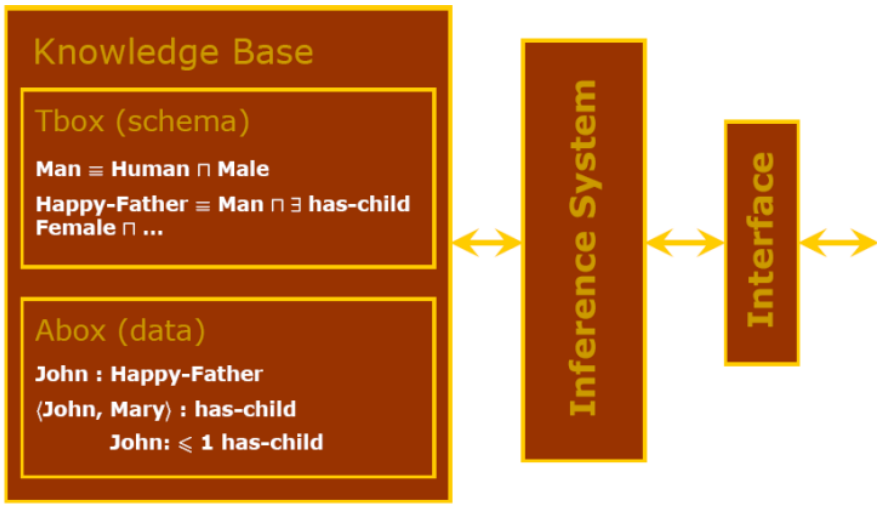
Powerful in its domain, lack of generalisation in problems different of classification (e.x. rules for total amount of a cart cannot be modelled because OWL doesn't have functions).

SWRL

Rule language combining OWL with RuleML to extend OWL capabilities: rule languages allow to define more powerful functions.

OWL AXIOMS WITH RULEML ISN'T DECIDABLE

OWL + Datalog is decidable and expressive.



Operators

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	¬Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	⩽1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	⩾2hasChild	$\exists^{\geqslant n} y.P(x,y)$

Reasoners (all available in protege)

- FaCT++
- Pellet
- HermiT
- Jena rules

Interpreting agents *(04/11)*

Interpret *different sources with different representations* in same model. Useful to get large data for AI applications.

Aligning

I receive data from one source and translate is representation to make one connected KG.

How to populate a KB?

If 2 KB use the same representation (rules, …) and objects, we can keep the kb separate and build a connection among the two (aligning them). Connections can be built using owl relations (equivalence, sameAs, equivalentClassOf, …)

sameAs is used to define the identity (same object of the real world), but can be misleading because of its general semantics.

We want to find mappings **automatically.** Usually classes and properties are much less than data, so automatically entity alignment is feasible.

Merge

Merge 2 KGs into one larger KG.

Linking data

- **Schema-level**: mappings between concepts and properties. Schema-mapping (or ontology matching).
- **Instance-level:** mappings between entity descriptors in heterogeneous sources. Record linkage, link discovery, entity co-resolution (we want to find entities linked with **sameAs** relation)

Ontology marching

We want to align conceptualisation. Complex problem, find correspondences (mapping) between entities of different ontologies (a source and a target ontology).

- Mapping = source, target, relation, similarity
- Alignment or matching = set of mappings

- Mapping cardinality: usually one to one (one concept of source is mapped at most to one concept of target). We don't want more than one *equivalence mapping arrow* because more classes mapped to the same target class could be redundant.

Ontology matching methods

Concept based:
- string based (edit distance)
- Language based (tokens)
- Constraint based (types, cardinality)
- Linguistic (vocabularies like WordNet)

Structure based:
- Taxonomy: when more classes match, if's useful to evaluate the structure of source and

KG integration and Construction *(14/11)*

Ontologies ***provide knowledge hubs (instances, relations, rich classification).*** Ontologies can be reused to a large extent when integration new data sources.

Instance matching

Two classes have the same instances. Doing instance matching require to have already matched the schema. A similarity function can be defined to measure the similarity of the values of each property of 2 instances (a source instance and a target instance) and use a threshold to define a match. So I'am assuming to have already linked properties and schemas.

Blocking approach: different phase to reduce the dimension of similarity matrixes.
Machine Learning can be exploited to lean similarity measures.

Semantic table annotation

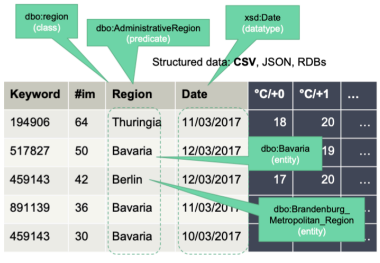
Semantic data enrichment
○ Intuitively: more data attached to input data
○ Depends on the kind input data …

Attach semantics to data
○ Metadata based on shared vocabularies
○ Transform data using a semantic model, e.g., RDF, JSON-LD, etc.

2. Use semantics to enrich the input data
○ Semantic used to support enrichment
○ May keep enriched data in the original format

3. Combination of 1 and 2

ASIA: assisted semantic interpretation and annotation of tabular data



Schema-level annotation:
- Mapping columns to types (suggest types (classes + datatypes) and properties based on their usage in)
- Validation rules

Instance-level annotation:
- reconciliation against external KBs
- Extension with external KBs
- Sufficient info to generate executable transformation to RDF

Relation extraction

Named Entity Recognition (NER)
Extraction of named entities mentioned in a text and classification by type (person, company, …)

Named Entity Linking (NEL)
Linking of named entities mentioned in a text to entities described in KB

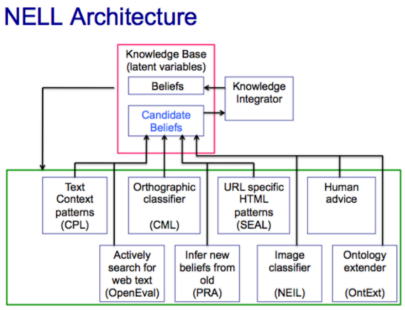


Figure 2: **NELL's software architecture.** NELL's growing knowledge base (KB) serves as a shared blackboard through which its various reading and inference modules interact. NELL's learning cycle iteratively retrains these software modules using the current KB, then updates the KB using these refined modules.

Challenges

Recognition

- Did I recognize it correctly?
  - ✓ @981THEBULL OMG. I want to win the **meet and greets** so 😱😭 I've been a **Swiftie** for 10 yrs.
- Out Of Vocabulary (OOV) entity mention identification problem
  - ✓ The Big Bang Theory being referred as *TBBT*
- Out of Knowledge base (OOKB) entity problem

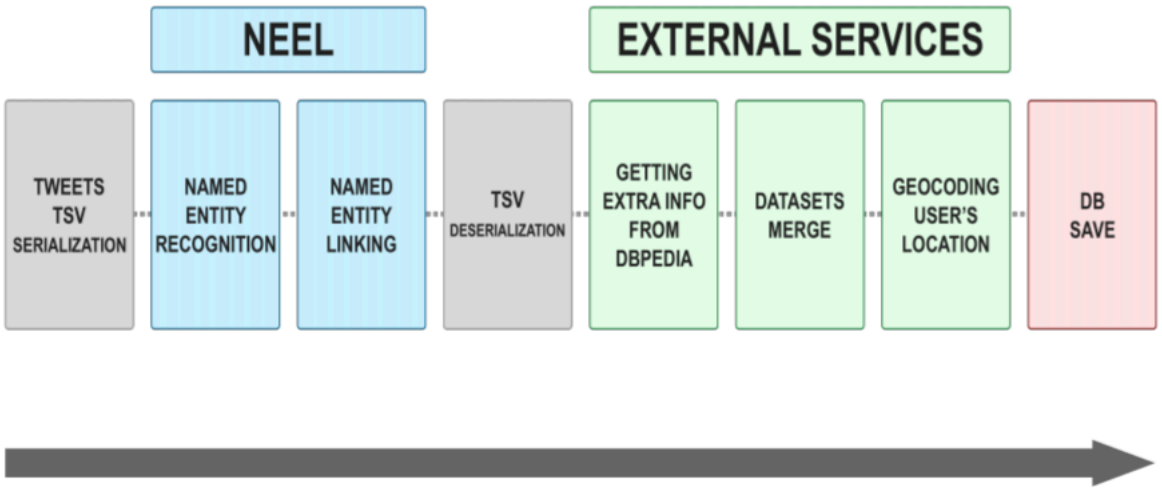
Linking

- Unlinkable entity -- Is it new? Is it misidentified/wrongly identified?
- What comprises new knowledge? Can we apply some measures for validating the credibility of new knowledge?

Classification

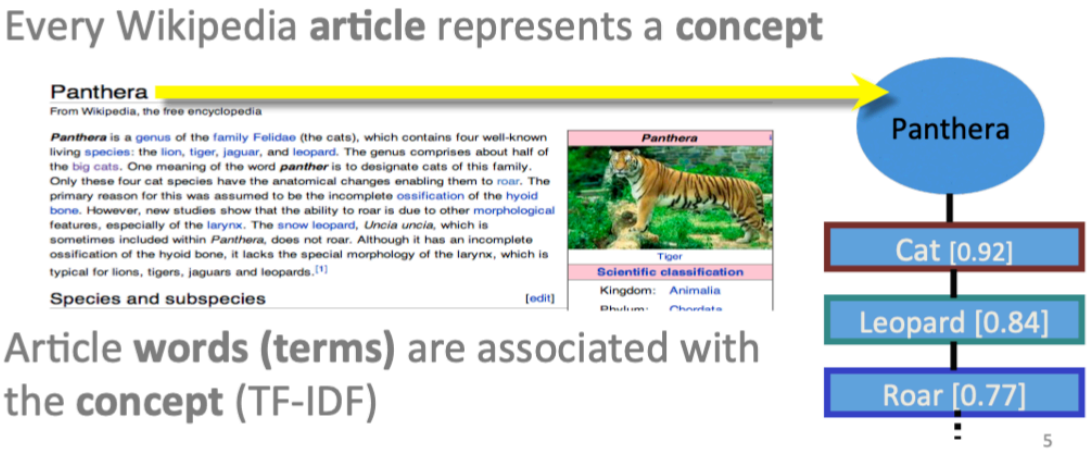
- Ambiguous named entities (e.g., Starbucks)
- Entity mention mis-classification
- Different NER systems have different classification ontologies. (e.g., Harry Potter – Person vs Character)
- New categories of named entities emerging in real-time on social media streams. Classifying them using an streamsClassifying them using an current ontology? Mapping it to a new current ontology?

TWINE: pipeline



Word Embeddings *(25/11)*

Explicit semantic analysis



Word embeddings

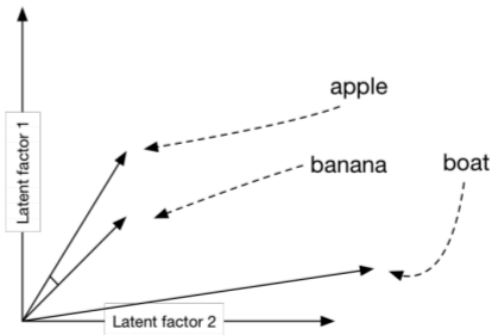
One-hot encoding

In detail: if we have N words in a dictionary, the i-th word is represented as a vector with dimension N full of zeros but with one 1 in the i-th position

Problems

- **cannot be used to evaluate similarity**
- **Don't scale well**

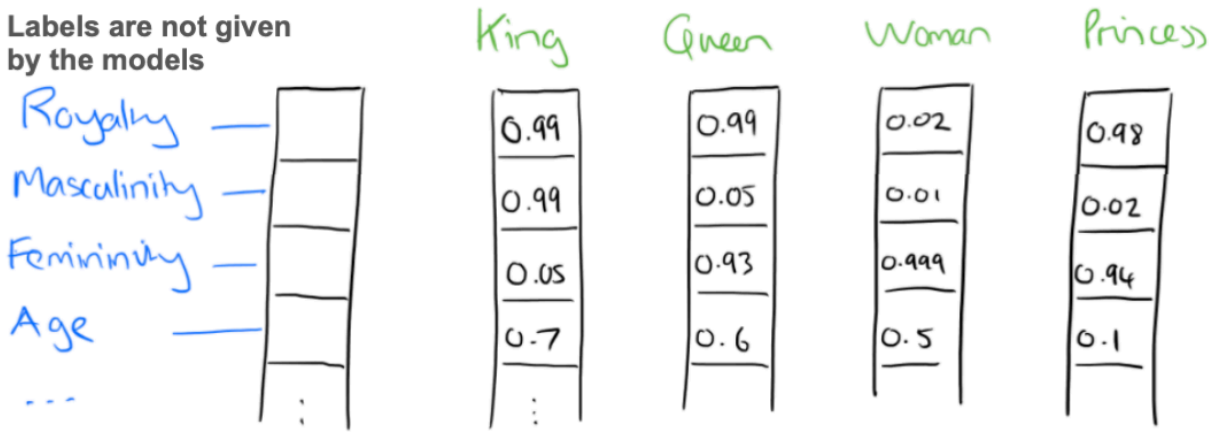
Similarity



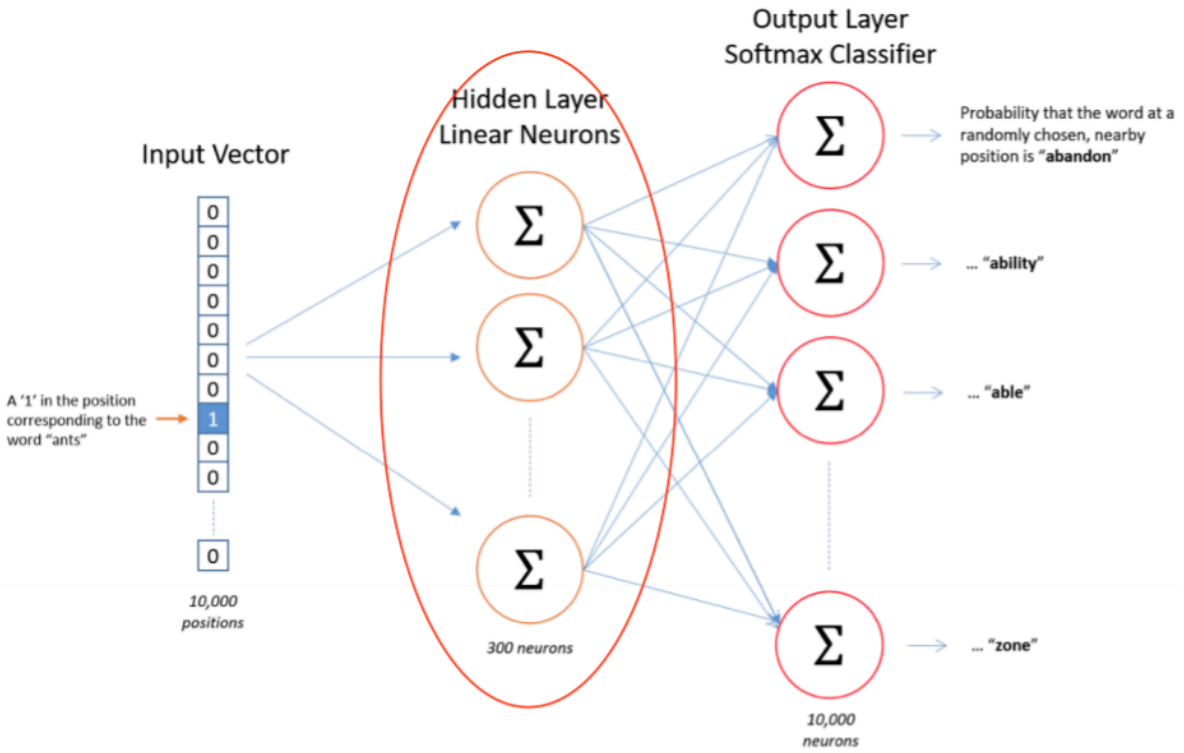
$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Word2Vec

We want a vector that contains information. One-hot encoding does not contain much information. Dense vectors have different components and share characteristics with other vectors



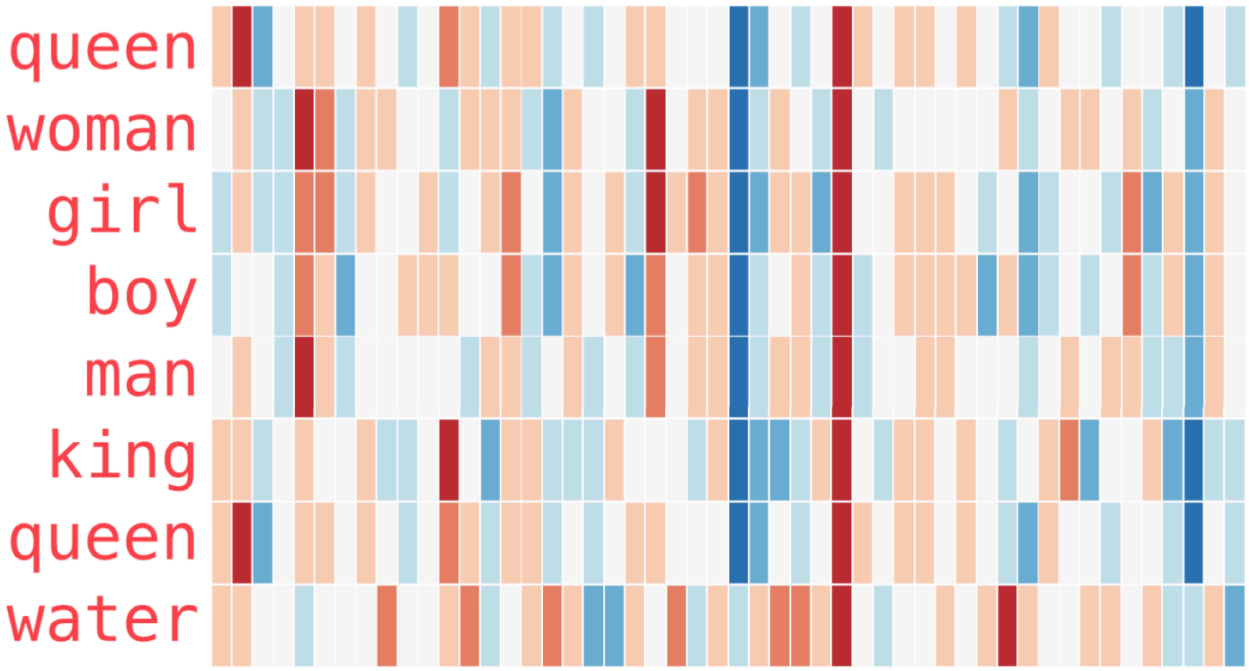
- **Word2vec** reduce the size of this matrix and embed vectors in a d-dimensional space
- Input is a one-hot word vector (all zeros but one 1, in the position representing the word)



- Desired output is one-hot vector (the nearby word)
- Model computes the probability of selecting each word in the dictionary given the input
- Update weights to correct values using back-propagation

The matrix representing the hidden layer **embeds the word vectors in a d dimensional space**

SImilarità: posso comparare due vettori, ciascuno rappresentate un concetto in forma **embedded**, con una metrica di **cosine similarity**, ossia il coseno dell'angolo che i vettori formano.



With Center of the Window (CBOW), continuous bag of words

Output: vector of probabilities that a word y is the center of a window in which input occurs

1. Presi i vettori hot-encoded delle parole di contesto, calcola il vettore medio
2. Moltiplica la media per la **matrice di input C (context)**
3. **Rimappa hidden layer moltiplicandolo per la matrice di target U**
4. **Output ottenuto con softmax**

Intuition: predict the target word given its context, so words before and after the target word

With skipgram

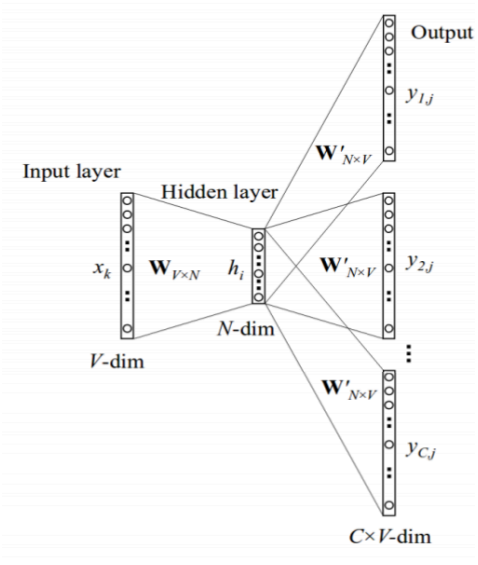
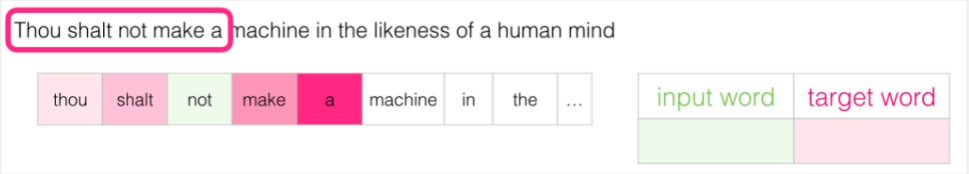
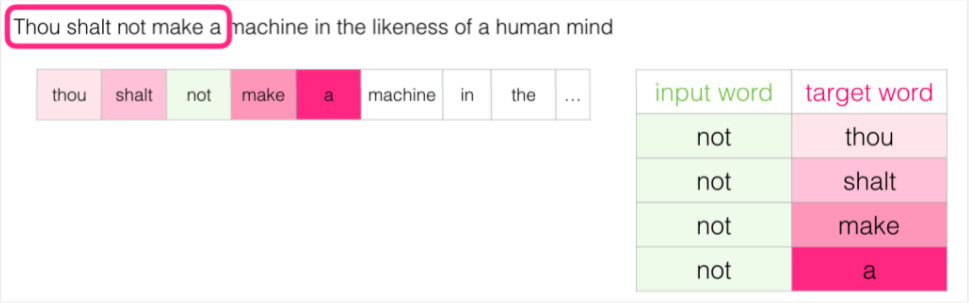


Figure 3: The skip-gram model.

Intuition: predict the context words given its target



This would add these four samples to our training dataset:



We then slide our window to the next position:

Training

Can be trained calculating an **error vector,** and using categorical cross entropy -> computational intense

Negative sampling

Switch task: train a model to output a probability of 2 words to be in the same context -> logistic regression model, easier then a categorical output
Issue: we have all the positive examples,
-> **noise-contrastive estimation:** we need negative examples too to get a much more robust mode (sampling words randomly),

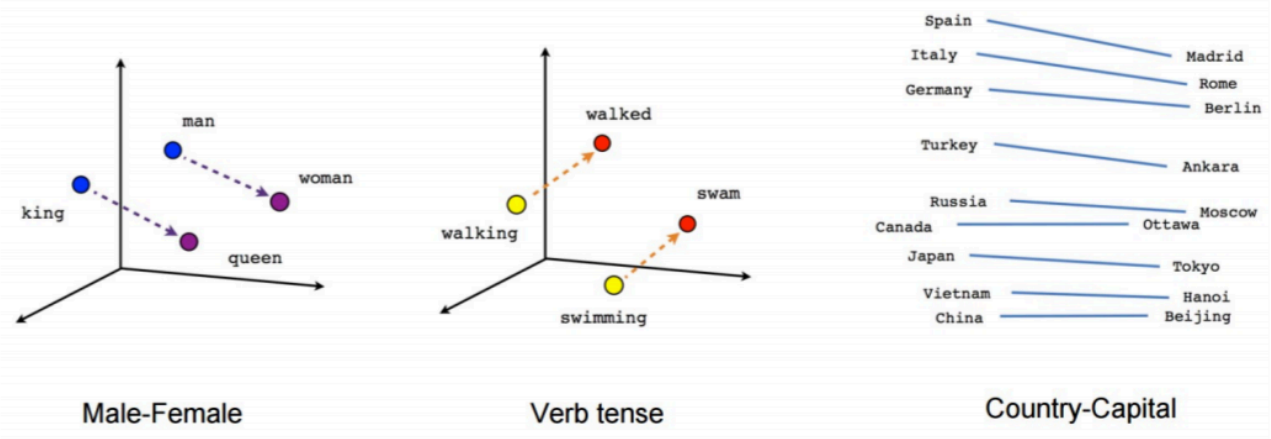
Skipgram with negative sampling: word2vec

- Preprocess text, determining the size of vocabulary (vocab_size), and words
- 2 matrices: **Embedding** and **Context:** matrices of **vocab_size x embedding_size** elements; both matrices are embeddings for the vocabulary

- In each training step, take one positive example and the associated negative examples; look up their embeddings: the input word in **Embedding** and the second words in **Context;**
- **Dot product** each couple to produce a number indicating similarity
- **Sigmoid** to get a probability value
- Evaluate the error, update both matrices representations according to the error

Recap

- Two main parameters to configure: window size, number of negative examples, number of dimensions to represent.
- The model optimises the probability of seeing one of nearby words of a given center word (or viceversa)
- **You can think of word2vec as a model that compress the output (input layer->hidden layer) and then decompress it to the original size (hidden layer-> output layer)**



Graph embeddings *(07/12)*

"Knowledge graph (KG) embedding is to embed components of a KG including entities and relations into continuous vector spaces, so as to simplify the manipulation while preserving the inherent structure of the KG."

A KG is a multi-relational graph composed of entities (nodes) and relations (different types of edges). Each edge is represented as a triple of the form (head entity, relation, tail entity), also called a fact, indicating that two entities are connected by a specific relation.

KG embedding aims to embed entities and relations into a low-dimensional continuous vector space, so as to simplify computations on the KG. Most of the currently available techniques use facts stored in the KG to perform the embed ding task, enforcing embedding to be compatible with the facts.

Knowledge Graphs Embeddings

Why should we embed?
- *Fast and intuitive way to* ***compute similarity***
  - *Soft reasoning skills: analogical reasoning & similarity*
- ***Latent components***
- ***Features generation***
  - *Reasoning under uncertainty using latent components and features: e.g. what's the team of Cristiano Ronaldo (based on latent patterns that connect players and teams?*

Embedding techniques

- *Select the* ***representations for entities and relationships****: entities as vectors and relationships are matrices*
- *Select a* ***score function*** *for entities and relationships,* to define the **plausibility** of each fact
- *Define a loss function* and **solve an optimisation problem to maximise total plausibility given a KG representation**

Translational Distance Models

TransE

TransE is the most representative **translational distance model.** It represents both entities and relations as vectors in the same space, say R^d. **Given a fact (h, r, t), the relation is interpreted as a translation vector r so that the embedded entities h and t can be connected by r with low error, i.e., h+r ≈ t when (h, r, t) holds.**

The detailed optimization procedure is described in Algorithm 1. All embeddings for entities and relationships are first **initialized following the random procedure** proposed in [4]. At each main iteration of the algorithm, the embedding vectors of the entities are **first normalized.** Then, a **small set of triplets is sampled from the training set, and will serve as the training triplets of the minibatch**. For each such triplet, we then s**ample a single corrupted triplet**. The **parameters are then updated by taking a gradient step with constant learning rate.** The algorithm is stopped based on its performance on a validation set.

Algorithm 1 Learning TransE

input Training set $S = \{(h, \ell, t)\}$, entities and rel. sets E and L, margin γ, embeddings dim. k.

initialize $\boldsymbol{\ell} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each $\ell \in L$

$\boldsymbol{\ell} \leftarrow \boldsymbol{\ell} / \|\boldsymbol{\ell}\|$ for each $\ell \in L$

$\mathbf{e} \leftarrow \text{uniform}(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each entity $e \in E$

loop

  $\mathbf{e} \leftarrow \mathbf{e} / \|\mathbf{e}\|$ for each entity $e \in E$

  $S_{batch} \leftarrow$ sample(S, b) // sample a minibatch of size b

  $T_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets

  for $(h, \ell, t) \in S_{batch}$ **do**

    $(h', \ell, t') \leftarrow$ sample$(S'_{(h,\ell,t)})$ // sample a corrupted triplet

    $T_{batch} \leftarrow T_{batch} \cup \{((h, \ell, t), (h', \ell, t'))\}$

  end for

  Update embeddings w.r.t. $\sum_{((h,\ell,t),(h',\ell,t')) \in T_{batch}} \nabla \left[\gamma + d(\boldsymbol{h} + \boldsymbol{\ell}, \boldsymbol{t}) - d(\boldsymbol{h'} + \boldsymbol{\ell}, \boldsymbol{t'})\right]_+$

end loop

The optimisation is carried out by stochastic gradient descent (in mini-batch mode), over the possible h, and t, with the additional constraints that the **L2-norm of the embeddings of the entities is 1** (*no regularisation or norm constraints are given to the label embeddings)*. This constraint is important for our model, as it is for previous embedding-based methods, because **it prevents the training process to trivially minimise L by artificially increasing entity embeddings norms.**

Issues: TransE learns similar representation for each tail-entity of a 1-to-N relations
Solution: Introducing Relation-Specific Entity Embeddings. To over come the disadvantages of TransE in dealing with 1-to-N, N-to-1, and N-to-N relations, an effective strategy is to allow an entity to have distinct representations when involved in different relations. In this way, even if the embeddings of Psycho, Rebecca, and RearWindow might be very similar given the relation DirectorOf, they could still be far away from each other given other relations.

TransH

TransH models entities again as vectors, but each relation r as a vector **r** on a hyperplane with w_r as the normal vector. Given a fact (h, r, t), the entity representations h and t are first projected onto the hyperplane. T**he projections are then assumed to be connected by r on the hyperplane with low error if (h, r, t) holds.**

Relaxing Translational Requirement h + r ≈ t

TransM associates each fact (h, r, t) with a weight θr specific to the relation. By assigning lower weights to 1-to-N, N-to-1, and N-to-N relations, TransM allows t to lie farther away from h + r in those relations.

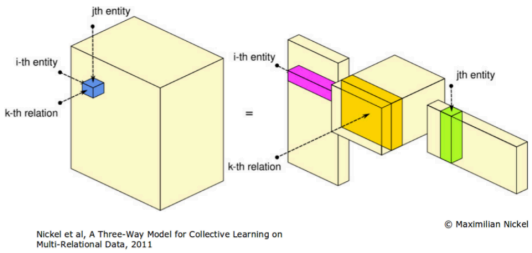
Semantic matching models

Semantic matching models exploit similarity-based scoring functions. They measure plausibility of facts by matching latent semantics of entities and relations embodied in their vector space representations.

RESCAL and Its Extensions

RESCAL associates each entity with a vector to capture its latent semantics. Each relation is represented as a matrix which models pairwise interactions between latent factors. The score of a fact (h, r, t) is defined by a bilinear function

$$f_r(h,t) = \mathbf{h}^\top \mathbf{M}_r \mathbf{t} = \sum_{i=0}^{d-1}\sum_{j=0}^{d-1} [\mathbf{M}_r]_{ij} \cdot [\mathbf{h}]_i \cdot [\mathbf{t}]_j,$$



Nickel et al, A Three-Way Model for Collective Learning on Multi-Relational Data, 2011

where **h, t** are vector representations of the entities, and $M \in R^{dxd}$ is a matrix associated with the relation.

DistMult

DistMult simplifies RESCAL by restricting M_r to diagonal matrices. For each relation r, it introduces a vector embedding **r** and requires $M_r = \text{diag}(r)$. The
scoring function is hence defined as

$$f_r(h,t) = \mathbf{h}^\top \text{diag}(\mathbf{r}) \mathbf{t} = \sum_{i=0}^{d-1} [\mathbf{r}]_i \cdot [\mathbf{h}]_i \cdot [\mathbf{t}]_i.$$

This score captures pairwise interactions between only the components of h and t along the same dimension, and reduces the number of parameters to O(d) per relation. However, this over-simplified model **can only deal with symmetric relations** which is clearly not powerful enough for general KGs.

HolE

HolE [62] combines the expressive power of RESCAL with the efficiency and simplicity of *DistMult*. It represents both entities and relations as vectors in R^d. Based on **circular correlation:**

$$f_r(h,t) = \mathbf{r}^\top (\mathbf{h} \star \mathbf{t}) = \sum_{i=0}^{d-1} [\mathbf{r}]_i \sum_{k=0}^{d-1} [\mathbf{h}]_k \cdot [\mathbf{t}]_{(k+i) \bmod d}.$$

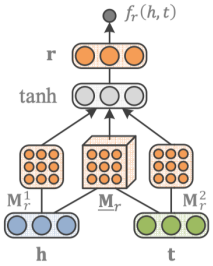
Circular correlation makes a **compression of pairwise interactions** (see also Fig. 2(c)). So HolE requires only O(d) parameters per relation, which is more efficient than RESCAL.
Meanwhile, since circular correlation **is not commutative**, i.e., h⋆ t/= t ⋆h, HolE **is able to model asymmetric relations as RESCAL does.**

ComplexE

Complex Embeddings (ComplEx). ComplEx [66] extends DistMult by introducing complex-valued embeddings so as **to better model asymmetric relations.**

Some approaches generate KG embeddings using structured data as a source, e.g., relations occurring in the KG, and are mainly targeted at predictive reasoning tasks such as link prediction. Other approaches generate the KG embeddings from text corpora using methods similar to the ones used to generate word embeddings, under the distributional hypothesis. These models referred to as text-based KG embeddings in the following, are mainly targeted at similarity evaluation tasks.

Neural Tensor Network



(b) NTN.

NTN is a **neural network** architecture. **Given a fact, it first projects entities to their vector embeddings in the input layer.** Then, the two entities h, t $\in R^d$ are combined by a relation-specific tensor $M_r \in R^{dxd}$ (along with other parameters) and **mapped to a non-linear hidden layer. Finally, a relation-specific linear output layer gives the score**

$$f_r(h,t) = \mathbf{r}^\top \tanh(\mathbf{h}^\top \underline{\mathbf{M}}_r \mathbf{t} + \mathbf{M}_r^1 \mathbf{h} + \mathbf{M}_r^2 \mathbf{t} + \mathbf{b}_r),$$

where $M_1^r, M_2^r \in R^{kxd}$ and $b_r \in R^k$ are relation-specific weight matrices and bias vectors, respectively.

NTN might be the most expressive model to date. But it requires $O(d^2k)$ parameters per relation, and is **not sufficiently simple and efficient to handle large-scale KGs.**

Training

The open world assumption (OWA) states that KGs contain only true facts and non-observed facts can be either false or just missing. **Negative examples can be generated by heuristics** such as the local closed world assumption.

It has been shown that minimising the logistic loss can help to find compact representations for some complex relational patterns such as transitive relations.

Pairwise ranking loss

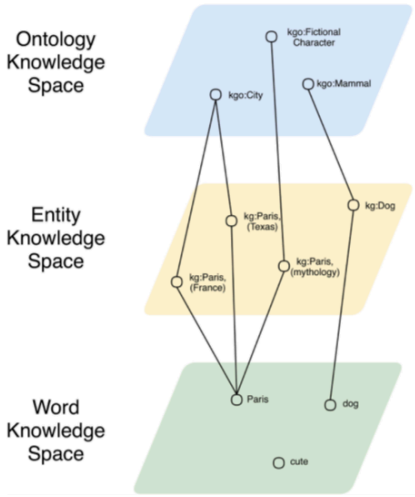
$$\min_{\Theta} \sum_{\tau^+ \in \mathbb{D}^+} \sum_{\tau^- \in \mathbb{D}^-} \max(0, \gamma - f_r(h,t) + f_{r'}(h',t'))$$

Note that in both types of optimisation, there are **constraints and/or regularisation terms** specified by different embedding models.

Trouillon et al. have shown that the **logistic loss** generally yields better results for **semantic matching models** such as DistMult and ComplEx, while the **pairwise ranking loss** may be more suitable for **translational distance models** like TransE.

Initialising Entity and Relation Embeddings.
Embeddings for entities and relations are usually initialised randomly from uniform distributions or Gaussian distributions.

Three Layer Vector Based Knowledge Representation



A unified model to represent knowledge in which words, entities and types are embedded in separated but linked spaces

Desired outcomes:
-Effective and efficient similarity between entities and/or types
-Improved word similarity via disambiguation
-Multi-aspect similarity (e.g., temporal similarity)

What do KGE learn?
- Latent types for entities and latent representation of relational types

What do KGE not learn?
- A relational model (X is true if city(Rome, Italy)

Majority of KGE approaches are outperformed by simple relational baseline

---

Limits of KGE

KGE are evaluated on HITS@10. This means that we look for the answer in the top-10 ranked list.

- *Still far from being a good performance.*
- *Also, are corrupted triples a good way to learn the embeddings? Open World assumption is still a challenge*
  - *Is (Barack Obama, president, United States) false? => temporal link prediction*
  - *KB: (Mark, child, Mary) and (Mark, child, Paul).*
- *Predict (Mark, child, ?). Which is the correct answer to consider in HITS@10?*
  - *Both? Only one?*

Enhanced Knowledge Graph Embeddings

- People have also tried to extend knowledge graph embeddings with information coming from **ontological knowledge, text, images and rules.**
- It is still hard to understand the behaviour of these models.

Ontology

Nice Idea: i**nterpret ontological concept has spheres in the space.**
Instance of relation => if entity vector is contained inside the sphere.

WordNet (Miller, 1995). Hierarchical concepts in these knowledge bases provide a natural way to categorize and locate instances. Therefore, the common simplification in previous work will lead to the following two drawbacks:

Insufficient concept representation: Concepts are essential information in knowledge graph. A concept is a fundamental category of existence (Rosch, 1973) and can be reified by all of its actual or potential instances. Figure 1 presents an example of concepts and instances about university staffs. Most knowledge embedding methods encode both concepts and instances as vectors, cannot explicitly represent the difference between concepts and instances.

Lack transitivity of both isA relations: `instanceOf` and `subClassOf` (generally known as isA) are two special relations in knowledge graph. Different from most other relations, isA relations exhibit transitivity, e.g., the dotted lines in Figure 1 represent the facts inferred by isA transitivity. The indiscriminate vector representation for all relations in previous work cannot reserve this property well (see Section 5.3 for details).

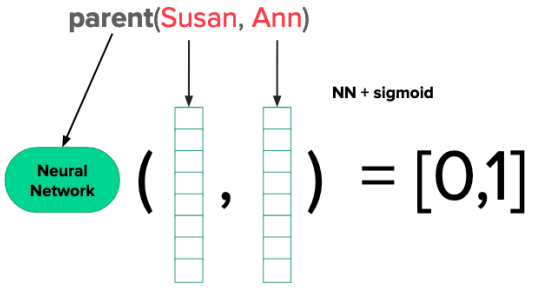
Logic tensor network

Logic Tensor Networks [Serafini+,2016] (LTNs) => neuro-symbolic [Garcez+,2008;Garcez+,2012] combine neural network a symbolic AI.

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



Training

The network is trained on a **best satisfiability task**:
- Learn the **representations**: (vectors for the constant, parameters for the predicates) in such a way that the **axioms are satisfied in the best possible way.**

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

Sub-symbolic Common sense: Distributional Representations

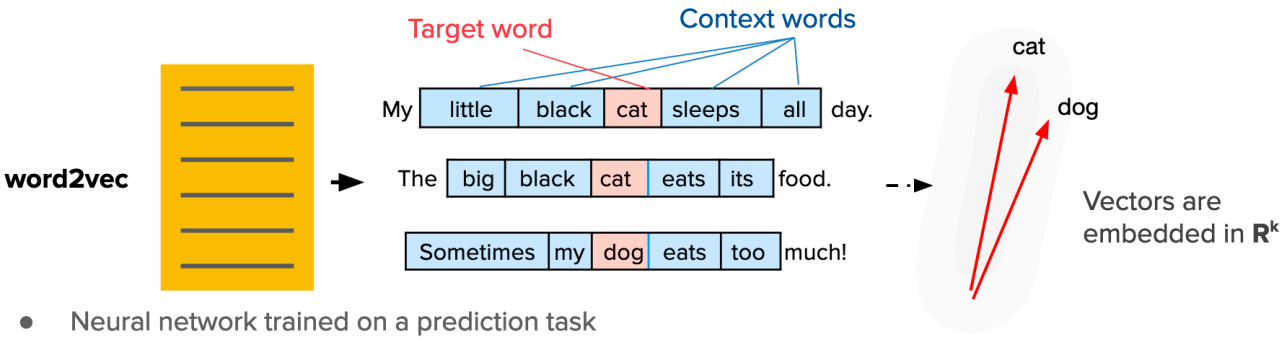
- Logical Reasoning: **Logic Tensor Networks**
- Sub-symbolic Commonsense: **Entity Embeddings**

*Distributional Hypothesis: **similar** words tend to appear in similar **contexts** [Wittgenstein1953, Firth1957]*

Key Idea: meaning can be derived from language usage

Sub-symbolic Commonsense: Word2Vec

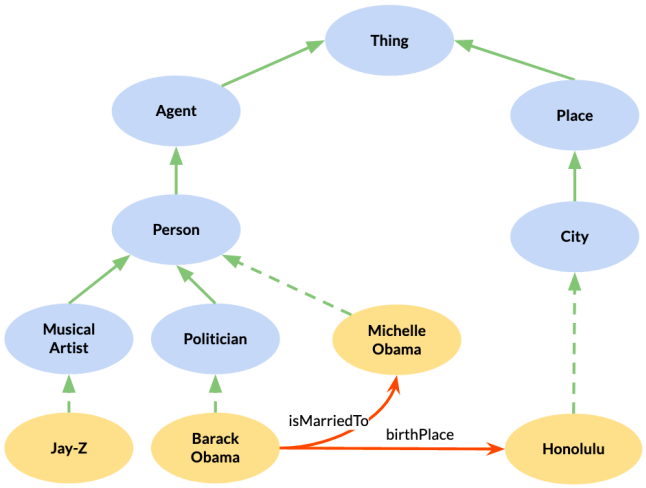
- **Grounded in Distributional Hypothesis [Harris,1954]: vector representations of language items**
- **i.e., embeddings, generated from a text corpus** [Mikolov+, 2013]



- 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

Sub-symbolic Commonsense: From Words to Entities



- Knowledge Graphs:
  - large representations of structured knowledge
  - < subject, predicate, object >
  - ~1.3 billion triples in DBpedia
  - symbols to refer to **entities**, **types**, and **relations**
  - types organized in sub-types graphs

DBpedia is a Knowledge Graph derived from Wikipedia

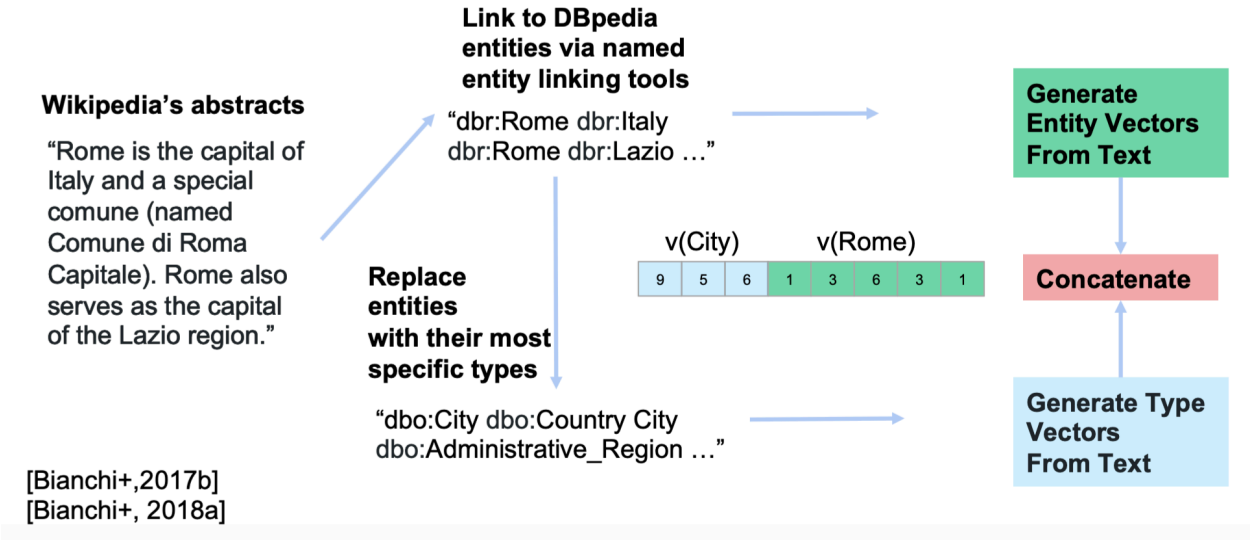
We can apply entity linking in text to create an **annotated text**.

Considerations

- **Important**: when does this combination fail? Are embeddings always the solution?
- The current version of LTNs **does not scale well** with the size of the KB and the number of axioms
- **Nice new paper** by Marra&Giannini&Diligenti&Gori **[ECML-PKDD2019]** on a framework for neuro-symbolic reasoning (LYRICS) based on tensorflow.

Typed entity in embeddings from text *(16/12)*

"Generate the KG embeddings from text corpora using methods similar to the ones used to generate word embeddings, under the distributional hypothesis. These models referred to as text-based KG embeddings in the following, are mainly targeted at similarity evaluation tasks."



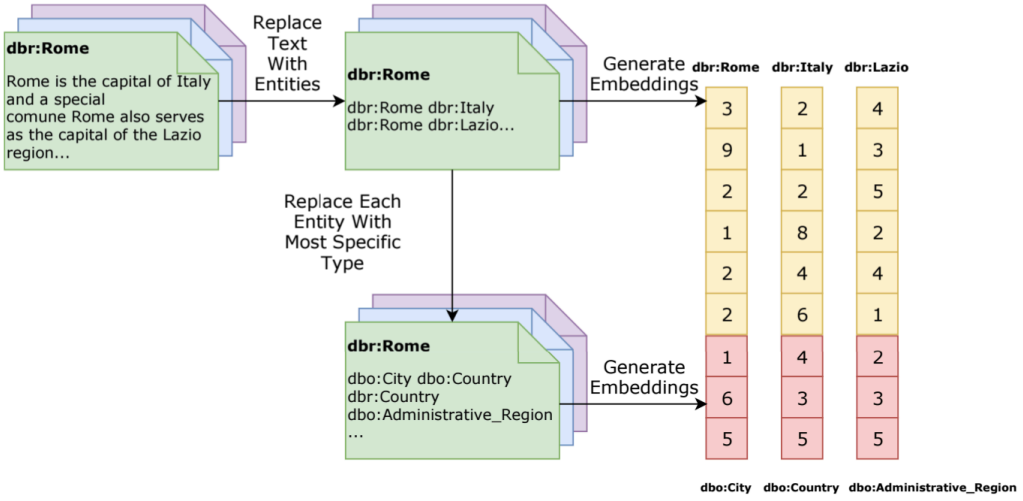


Fig. 1. Entity embedding process: textual content is replaced by entities. Each entity is replaced with its own type. Embeddings can be then generated using word2vec. Finally each entity is concatenated to its own type.

TEE: Analogical Reasoning with Entities

- *Word embeddings (e.g., [Mikolov+, 2013])*
- *Text-based Entity Embeddings*
- *Text as main source vs. Graph as main source*
- *Typed Entity Embeddings (TEE): use word embeddings algorithms on documents where entities and types replace words*
- ***Pros: good for similarity evaluation***
- ***Cons: no embedding of relations, just entity***

Hypothesis

1. *Analogical reasoning with entities should have better results than analogical reasoning with words*
2. *Types can be used to improve the accuracy of the models*

Time aware similarity

"Time can sneak into entity similarity in a way that cannot be controlled, because entities that share a temporal context are more likely to co-occur in the text (we refer to this implicit effect of time on entity similarity as to the time effect hypothesis)"

We **encode representations of years,** i.e., we **embed regular time periods with a yearly granularity.** We build **year representations from the textual description of events occurring during each year**, which are available in different web sources. We generate **year representations by aggregating the representations of the entities that that take part in events occurring in those particular years.** These representations are then used to define two parametric time-aware similarity functions: time-flattening and time-boosting similarities.

Time flattened similarity: *to reduce the impact of time in the similarity.*
E.g., make US presidents similar independently from their temporal context.

Time boosted similarity: *to boost the impact of time in the similarity.*
E.g., make politicians that share temporal contexts more similar

Implementations

Our main hypothesis is that discrete periods of times can be embedded in a vector space, where each period is represented by a vector, in such a way that years that are near in time have similar vectors. A second hypothesis that drives our approach is that a period of time, e.g., a year, can be described by the entities that take part in the events that occur during the time period. For example, years in the first half of the 40s are characterised by World War II events and by the entities that had a relevant role in these events. For the experiments conducted in this paper, we **consider textual descriptions of events that appear in the Wikipedia pages that describe years**.

To generate the representation of a year, we extract entities from the corresponding Wikipedia page and compute the average vector of the entity vectors defined in the EE space.

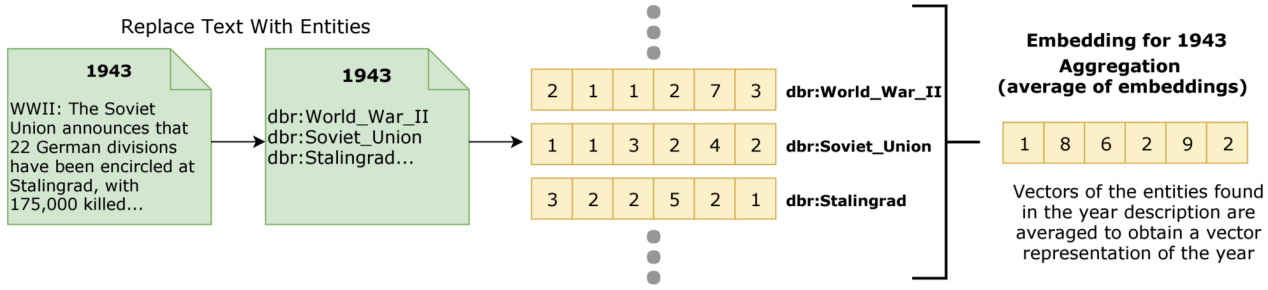


Fig. 2. Year embedding process.

Similarities

Given an entity e ∈ E is it possible to find its most representing year in Eτ by considering the functions defined in the model. To get the most representative year for a given entity we select the most similar year in E^t to a given entity.

$$\rho(e) = \underset{e^\tau \in E^\tau}{\operatorname{argmax}}\, cos(\phi(e), \omega(e^\tau))$$

Time-flattened similarity

$$\psi(e_1, e_2) = \alpha\eta(\mathbf{e_1}, \mathbf{e_2}) - (1-\alpha)\eta_n(\mathbf{e_1^\tau}, \mathbf{e_2^\tau})$$

Time flattening is obtained by **subtracting the temporal similarity η_n (normalised)** of the most representative temporal periods (i.e., years) of both entities.

Time-boosted similarity

$$\psi(e_1, e_2) = \alpha\eta(\mathbf{e_1}, \mathbf{e_2}) + (1-\alpha)\eta_n(\mathbf{e_1^\tau}, \mathbf{e_2^\tau})$$

Evaluation

Evaluate if entity representations share (implicitly) time similarities

"Our dataset contains 152 battles linked to Wikipedia (and thus DBpedia) from the two different periods 1914-1918 and 1939-1945. 63 battles are from World War I while 89 are from World War II. Results. In Table 1 we show the confusion matrix that we obtained after **clustering the embedded vectors of the battles with K-Means. Out of 152 samples, 146 were correctly associated to the same cluster, while 8 were classified in the wrong one."**

Time order

"For each entity pair, we compare the manually determined relative order with the order of their most representative years according to our model. The most representative year of an entity is the the closest year to the entity in the vector space."

Applications

Potential applications can be found in **time-aware entity recommendations** (e.g., find "related contemporary entities" vs. "related entities in the past" vs. "time-independent related entities") and in **temporal information retrieval**, where it is important to keep track of the time factor.

Conclusions

- Time can be represented in the vector space using events descriptions
- Time sneaks into entity similarity (time bias)
- Time bias can be controlled by considering explicit representations of time periods

Future Work

- Study compositionality of time periods representations
- Comparison with Doc2Vec
- Improve time-aware similarity measure
- Comparison with other KG embeddings models

Training Temporal Word Embeddings with a Compass

Goals
- Design a model to represent word meanings across time
- Entity machine in text from a specific period of time
- Automatically detect language dynamics and trends

Temporal word embeddings
- Temporal word embeddings are vector representations of words during specific temporal intervals (e.g. the year 2001, the day 3/28/2018)
- They are learned from diachronic text corpora, divided in multiple temporal slices (e.g. news articles, social posts)

Motivation for the approach

Most of the proposed TWEMs align multiple vector spaces by enforcing word embeddings in different time periods to be similar (Kulkarni et al. 2015; Rudolph and Blei 2018). This method is based on the assumption that the majority of the words do not change their meaning over time.

Model

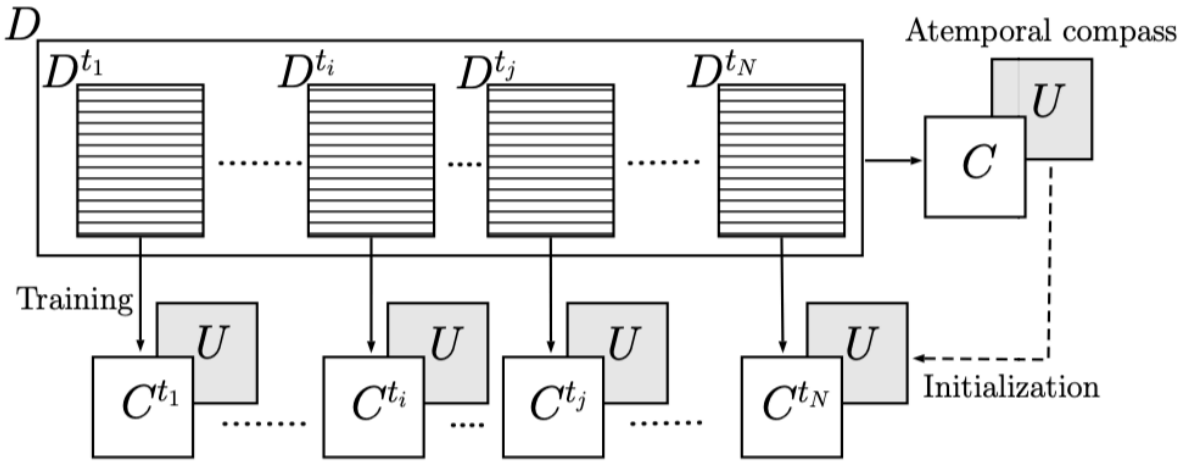


Figure 1: The TWEC model. The temporal context embeddings $\mathbf{C}^t$ are independently trained over each temporal slice, with frozen pre-trained atemporal target embeddings $\mathbf{U}$.

"Our approach is based on the same assumption used in previous work, that is, t**he majority of words do not change their meaning over time** (Kulkarni et al. 2015). From this assumption, we derive a second one: **we assume that a shifted word, i.e., a word whose meaning has shifted across time, appears in the contexts of words whose meaning changes slightly**. However, differently from the latter assumption, our assumption is particularly true for shifted words. For example, the word clinton appears during some temporal periods in the contexts of words that are related to his position as president of the USA (e.g., president, administration); conversely, the meanings of these words have not changed. **The above assumption allows us to heuristically consider the target embeddings as static,** i.e., **to freeze them during training, while allowing the context embeddings to change based on co-occurrence frequencies that are specific to a given temporal interval.** Thus, our training method returns the context embeddings as temporal word embeddings."

Details of our model using CBOW as underlying Word2vec model, since we empirically found that it produces temporal models that show better performance

Phases

1. First, we construct **two atemporal matrices C and U by applying the original CBOW model on the whole diachronic corpus D, ignoring the time slicing**; C and U represents the set of atemporal context embeddings and atemporal target embeddings, respectively.
2. Second, for each time slice Dti , we construct a temporal context embedding matrix Cti as follows. **We initialise the output weight matrix of the neural network with the previously trained target embeddings from the matrix U.** We run the **CBOW algorithm using the temporal slice Dti**. During this training process, **the target embeddings of the output matrix U are not modified, while we update the context embeddings in the input matrix Cti.** After applying this process on all the time slices Dti, **each input matrix Cti will represent our temporal word embeddings at the time ti. Intuitively, it moves the temporal context closer to the atemporal target embeddings. The resulting temporal context embeddings can be used as temporal word embeddings:** they will be already aligned, thanks to the shared atemporal target embeddings used as a compass during the independent trainings.

Complementing Logical Reasoning with Sub-Symbolic Common sense *(06/01)*

*Logic Tensor Networks (LTNs), a neuro-symbolic model. Using **LTNs** it is possible to integrate axioms and facts with common sense knowledge represented in a sub-symbolic form in one single model performing well in reasoning tasks.*

LTN

Logic Tensor Networks (LTNs) are an example of a **neuro-symbolic model that embeds first-order fuzzy logic in a vector space.** In LTNs logic **constants are represented as vectors and n-ary predicates are n-ary functions whose values are real numbers in the range [0, 1].** A neural network for each predicate **learns both the representation of logic constants and the weights that characterise the n-ary function.** Learning is based on a set of axioms.

Sub-symbolic common sense

Distributional semantics has been found to provide representations that are strongly correlated with associative learning; we thus refer to these **representations as sub-symbolic common sense.**

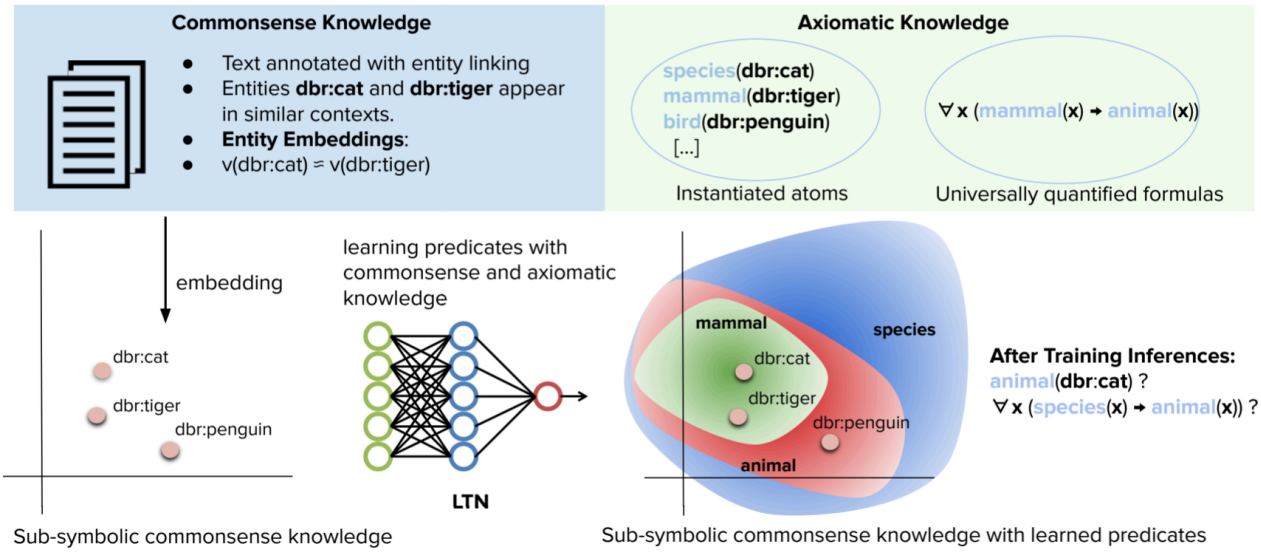


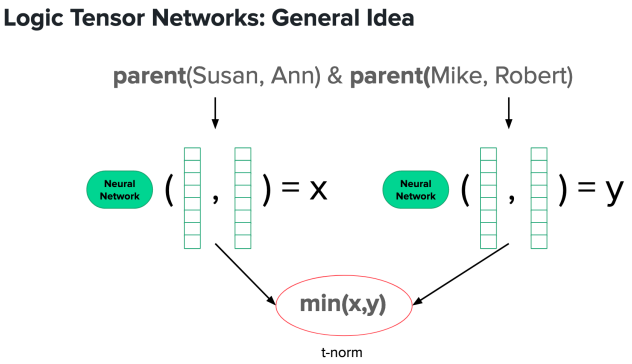
Fig. 1. We learn embeddings from text and we use LTNs to learn how to represent predicates with the network. "dbr:" stands for the DBpedia Knowledge Graph namespace.

While LTNs generally needs to learn the representation of vector from scratch **in our setting are already learned (sub-symbolic common sense) and do not need any more training.**

Ideas

Grounding: *The action of representing elements of the logic language as elements in the vector space is referred to as* ***grounding****.*

Ogni costante è un vettore R^n*, ogni atomo è costruito tramite un processo di* ***grounding.*** Ogni predicato è rappresentato da matrici di pesi di una rete neurale -> Each predicate in LTN is a NN (the training phase is on many networks as predicates). Le formule **quantificate** si ottengono usando operazioni di aggregazione. I connettivi sono funzioni statistiche (e.s. t-norm)



LTNs reduce the learning problem to a **maximum satisfiability problem**: the task is to find groundings for terms and predicates that **maximise the satisfiability of the formulas in the knowledge base.**

For a grounded formula like mammal(cat), **the network updates the representation of the predicate mammal (i.e., the parameters in the tensor layer) and the representation of cat (i.e., its vector)** in such a way that the degree of truth of an instantiated atom is closer to 1.

Semantic: distributional hypothesis *(16/12)*

Distributional Hypothesis
The degree of semantic similarity between two linguistic expressions A and B is a function of the similarity of the linguistic contexts in which A and B can appear.

"It is fair to say that the names that best represent this approach to meaning are **distributional** and **context-theoretic semantics.** In fact, the essence of such models resides in the idea that **word meaning depends on the contexts in which words are used, and that at least parts of a word content can be characterised by its contextual representation,** to be defined as an abstraction over the linguistic contexts in which a word is encountered" - *Distributional semantics in linguistic and cognitive research*

"We must not forget that language includes terms like walk or dog – whose **meaning is unlikely to be learnt without having some specific sensory-motor experience** with the objects and events to which they refer – side by side with terms like understand, society, idea, etc. whose **semantic content we most probably acquire via linguistic experiences**, i.e. by observing how they are used in language. As we will see in §3, applying purely corpus-based distributional methods allows us to tackle the crucial issue of *how and to what extent feature extracted from the linguistic input shape meaning*. Therefore, even the contingent restriction of distributional methods to language data may be turned into a *source for interesting research questions about meaning*" - *Distributional semantics in linguistic and cognitive research*

"Since n-dimensional vectors represent the coordinates of points in a n-dimensional space, **if we associate a word with a contextual representation and we formalise the latter as a vector, we can also conceive words as points in a "distributional space",** i.e. a **space whose dimensions are provided by the relevant linguistic contexts,** and in which the position of a word-vector is **determined by its statistical distribution in each context**. In turn, **if we adopt the DH and assume that semantic distance between words is a function of their distributional similarity, we can interpret this distributional vector space as a semantic space,** in which distances between points correspond to semantic distances between the corresponding words" - *Distributional semantics in linguistic and cognitive research*

"**Distributional models are indeed empiricists**, since they claim that **(at least parts of) word semantic properties depend on the way words are used,** and therefore can be inductively derived from the statistical analysis of language data." - *Distributional semantics in linguistic and cognitive research*

DH and biases

"We elaborate on further implications of our results. In psychology, our results add to the credence of the IAT by replicating its results in such a different setting. Further, our methods may yield an efficient way to explore previously unknown implicit associations. Researchers who conjecture implicit associations might first test them using the WEAT on a suitable corpus before testing human subjects. Similarly, our methods could be used to quickly find differences in bias between demo graphic groups, given large corpora authored by members of the respective groups. If substantiated through testing and replication, the WEAT may also give us access to implicit associations of groups not available for testing, such as historic populations. We have demonstrated that word embeddings encode not only stereotyped biases but also other knowledge, such as the visceral pleasantness of flowers or the gender distribution of occupations. **These results lend support to the distributional hypothesis in linguistics,**

namely that the statistical contexts of words capture much of what we mean by meaning". - *Semantics derived automatically from language corpora contain human-like biases*

Weak and strong DH

Weak

"The DH only assumes the **existence of a correlation between semantic content and linguistic distributions, and exploits such correlation to get at a better understanding of the semantic behaviour of lexical items**. Assuming this weak version of the DH does not entail assuming that word distributions are themselves constitutive of the semantic properties of lexical items at a cognitive level. It rather corresponds to **taking semantics as a kind of "latent variable" which is responsible for the linguistic distributions that we observe, and that we try to uncover by inspecting a significant number of such distributions.**" - *Distributional semantics in linguistic and cognitive research*

Strong

"In its strong version, **the DH is a cognitive hypothesis about the form and origin of semantic representations**, as countenanced by Miller and Charles. Repeated encounters with words in different linguistic contexts eventually lead to the formation of a contextual representation as an abstract characterisation of the most significant contexts with which the word is used. Crucially, assuming the strong DH entails assuming that word distributions in context have a specific causal role in the formation of the semantic representation for that word. Under this version, **the distributional behaviour of a word in contexts is not only taken as a way to get at its semantic behaviour, but indeed as a way to explain its semantic content at the cognitive level**. […] Although under different fashions, all these models adhere to the strong DH, and **assume that the encounters of a word in different linguistic environments have an effect on the semantic representations of these words**, e.g. on the similarity relationships that are established among them in the mental lexicon." - *Distributional semantics in linguistic and cognitive research*

Issues and solutions

Compositionally

"It is commonly stated that the DH mainly concerns lexical meaning. ***However, it is also true that part of the meaning of a lexical expression consists in its ability to semantically compose with other linguistic expressions to form the meaning of a complex linguistic structure.*** *[…]. A model for semantic representations, like the one proposed by the DH, should therefore be able to explain how t****he meaning of a complex expression can be built from the meanings of its components,*** *and at the same time should be able to model the semantic constraints governing the range of expressions with which a given lexeme can compose." - Distributional semantics in linguistic and cognitive research*

"Although many problems still remain to be solved, ***compositionally is indeed an aspect of meaning that can in principle be tackled with contextual representations,*** *and actually this is an important issue in the agenda of distributional semantics nowadays (cf. also Jones & Mewhort 2007)." - Distributional semantics in linguistic and cognitive research*