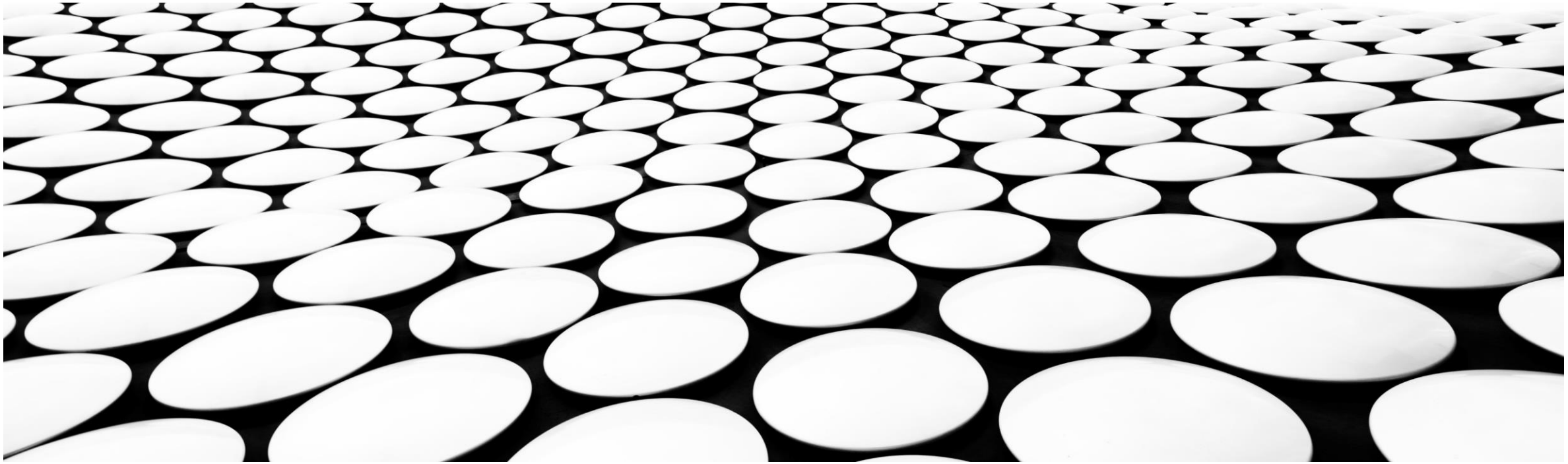
MACHINE LEARNING WITH ONTOLOGIES

LUKÁŠ KOREL



CONTENT

1. Introduction to ontology
2. Ontology components
3. Ontology languages
4. Ontology & ML

ONTOLOGY

- Attempt to represent entities, ideas and events, with all their interdependent properties and relations, according to a system of categories
- Explicit specification of a conceptualization of a domain

ONTOLOGY - INFORMATION SCIENCE VS. PHILOSOPHY

- **Ontology in philosophy:**
  - The branch of philosophy that studies concepts such as existence, being, becoming, and reality. It includes the questions of how entities are grouped into basic categories and which of these entities exist on the most fundamental level
- **Ontology in IT:**
  - Ontology encompasses a representation, formal naming and definition of the categories, properties and relations between the concepts, data and entities that substantiate one, many, or all domains of discourse
  - A way of showing the properties of a subject area and how they are related, by defining a set of concepts and categories that represent the subject

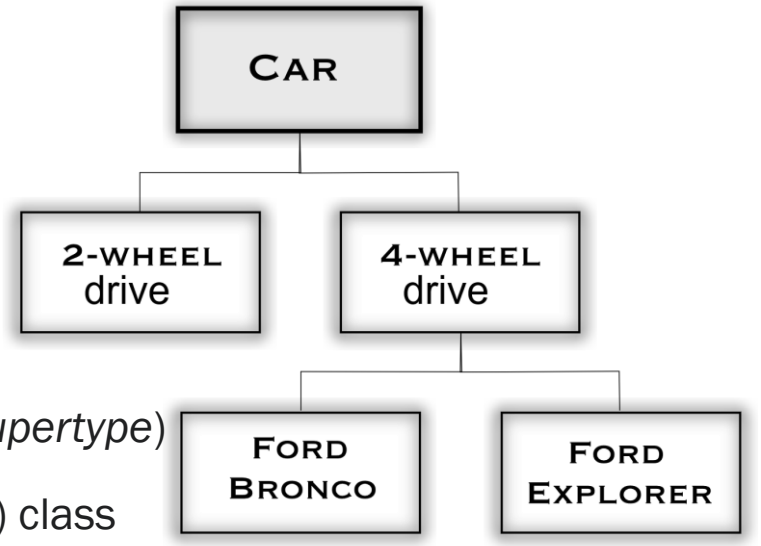
ONTOLOGY IN SCIENCE

- Every academic discipline or field creates ontology
- Limits complexity and organize data into information and knowledge

ONTOLOGY COMPONENTS

- **Individuals** Instances or objects (the basic or "ground level" objects)
- **Classes** Sets, collections, concepts, classes in programming, types of objects or kinds of things
- **Attributes** Aspects, properties, features, characteristics or parameters that objects (and classes) can have
- **Relations** Ways in which classes and individuals can be related to one another
- **Function terms** Complex structures formed from certain relations that can be used in place of an individual term in a statement
- **Restrictions** Formally stated descriptions of what must be true in order for some assertion to be accepted as input
- **Rules** Statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form
- **Axioms** Assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application
- **Events** The changing of attributes or relations

HIARARCHY
CLASS ← SUBCLASSES



- Class can subsume or be subsumed by other classes
- Class subsumed by another is = *subclass* (or *subtype*) of the subsuming class (or *supertype*)
- Inheritance of properties from the parent (subsuming) class to the child (subsumed) class

RELATIONS - SUBSUMPTION

- **Subsumption** relation <*is-a-superclass-of*> (the converse of <*is-a*>, <*is-a-subtype-of*> or <*is-a-subclass-of*>) defines which objects are classified by which class
  - Example: Class Woman *is-a-subclass-of* Human, which in turn *is-a-subclass-of* Mammal
- The is-a-subclass-of relationships creates a tree-like structure (or, more generally, a partially ordered set)

RELATIONS - MEREOLOGY

- Mereology relation <*part-of*>
  - Represents how objects combine to form composite objects
  - For example, if we extended our example ontology to include concepts like Steering Wheel, we would say that a "Steering Wheel **is-by-definition-a-part-of-a** Ford Explorer" since a steering wheel is always one of the components of a Ford Explorer
- With meronymy relationships the hierarchy is no longer able to be held in a simple tree-like structure
- Members can appear under more than one parent or branch
- This new structure is known as a **directed acyclic graph**

RELATIONS APPEARS BETWEEN DIFFERENT CATEGORIES

- classes
- individuals
- an individual and a class
- a single object and a collection
- collections

DOMAIN-SPECIFIC RELATIONS

- Relation types are sometimes domain-specific
- Used to store specific kinds of facts or to answer particular types of questions
- If the definitions of the relation types are included in an ontology, then the ontology defines its own ontology definition language (example: Gellish ontology)
- Example (Domain of automobiles):
  - We might need a <*made-in*> type relationship which tells us where each car is built
  - **Ford Explorer** is <*made-in*> **Louisville**
  - Ontology may also know that **Louisville** <is-located-in> **Kentucky** and **Kentucky** <is-classified-as-a> **state** and <is-a-part-of> the **U.S.**
  - Software using this ontology could now answer a question like „Which cars are made in the U.S.?“

ONTOLOGY LANGUAGES

- In computer science and artificial intelligence, ontology languages are formal languages used to construct ontologies
- Allow encoding of knowledge about specific domains and often include reasoning rules that support the processing of that knowledge
- Usually declarative languages
- Almost always generalizations of frame languages
- Commonly based on either first-order logic or on description logic

ONTOLOGY LANGUAGES – CLASSIFICATION BY SYNTAX

- Traditional syntax ontology languages
  - Common Logic - and its dialects, CycL, DOGMA (Developing Ontology-Grounded Methods and Applications), F-Logic (Frame Logic), KIF (Knowledge Interchange Format), KL-ONE, KM programming language, LOOM, OCML (Operational Conceptual Modelling Language), OKBC (Open Knowledge Base Connectivity), PLIB (Parts LIBrary)
- Markup ontology languages
  - Use a markup scheme to encode knowledge, most commonly with XML.
  - DAML+OIL, Ontology Inference Layer (OIL), Web Ontology Language (OWL), Resource Description Framework (RDF), RDF Schema (RDFS), SHOE
- Controlled natural languages
  - Attempto Controlled English
- Open vocabulary natural languages
  - Executable English

ONTOLOGY LANGUAGES – CLASSIFICATION BY STRUCTURE (LOGIC TYPE)

- Frame-based
  - Three languages are completely or partially frame-based languages (F-Logic, OKBC, KM)
- First-order logic-based (also kwnown as **predicate logic** or **quantificational logic**)
  - Several ontology languages support expressions in first-order logic and allow general predicates (Common Logic, CycL, KIF)

$$\forall x \forall y (P(f(x)) \rightarrow \neg (P(x) \rightarrow Q(f(y), x, z)))$$

- Description logic-based
  - Decidable fragment of first order predicate logic
  - Description logic provides an extension of frame languages, without going so far as to take the leap to first-order logic and support for arbitrary predicates (KL-ONE, RACER, OWL – most often used in ontologies)

F-LOGIC

Classes

```
man::person.
woman::person.
brad:man.
angelina:woman.
```

Statements

```
person[hasSon=>man].
brad[hasSon->{maddox,pax}].
married(brad,angelina).
```

Axioms

```
man(X) <- person(X) AND NOT woman(X).
X:person[hasFather->Y] <- Y:man[hasSon -> X].
```

DESCRIPTION LOGIC

Symbol	Description	Example	Read
⊤	⊤ is a special concept with every individual as an instance	⊤	top
⊥	empty concept	⊥	bottom
⊓	intersection or conjunction of concepts	C ⊓ D	C and D
⊔	union or disjunction of concepts	C ⊔ D	C or D
¬	negation or complement of concepts	¬ C	not C
∀	universal restriction	∀ R . C	all R-successors are in C
∃	existential restriction	∃ R . C	an R-successor exists in C
⊑	Concept inclusion	C ⊑ D	all C are D
≡	Concept equivalence	C ≡ D	C is equivalent to D
≐	Concept definition	C ≐ D	C is defined to be equal to D
:	Concept assertion	a : C	a is a C
:	Role assertion	(a , b) : R	a is R-related to b

WHY ONTOLOGIES + MACHONE LEARGNING?

"IN THE KNOWLEDGE LIES THE POWER"

ONTOLOGIES USAGE

- Formal representation and reason over domain knowledge
- Provide background knowledge in similarity-based analysis and machine learning models
- Provide methods that use ontologies to compute similarity and incorporate them in machine learning methods
- Semantic similarity measures and ontology embeddings can exploit the background knowledge in ontologies
- As constraints that improve machine learning models

ONTOLOGIES & ML

- Machine learning methods are applied widely to develop predictive models
- Domain-specific knowledge can be used to
  - Constrain search
  - Find near-optimal solutions faster
  - Find better solutions
- Power of AI systems lies in the domain-specific knowledge they encode
- AI systems are able to exploit klowledge
- Domain-specific knowledge is encoded in ontologies and database
- Ontologies provide controlled vocabularies for characterizing domain

ONTOLOGIES & ML

- We want to identify general ways in which ontologies (and their underlying formalisms based on OWL) can be combined with the modern machine learning and optimization methods
- Use traditional semantic similarity measures applied to ontologies
- Semantic similarity measures determines the similarity between two/more entities using the formalized background knowledge in ontologies

ENTAILMENTS FROM LOGICAL EXPRESSIONS

- The semantics of logical languages gives rise to entailment
  - Statement φ is logically entailed by a set of statements O if all the structures in which all statements in O are true also make φ true
- Ontologies in OWL are primarily sets of axioms
- Axioms can be used to generate a graph structures
- Conversion of axioms into nodes and edges
  - Possibility to generate using entailments
  - Example: If ***X*** $\exists$***part-of.Y*** and ***Y*** $\exists$***part-of.Z*** are a part of an ontology, and the relation `part-of` is transitive (part-of $\circ$ part-of $\subseteq$ part-of), then ***X*** $\exists$***part-of.Z*** would be entailed and consequently a `part-of` edge between *X* and *Z* created (*X* -***part-of-->*** *Y*)
- Types and complexity of axiom patterns giving rise to edges

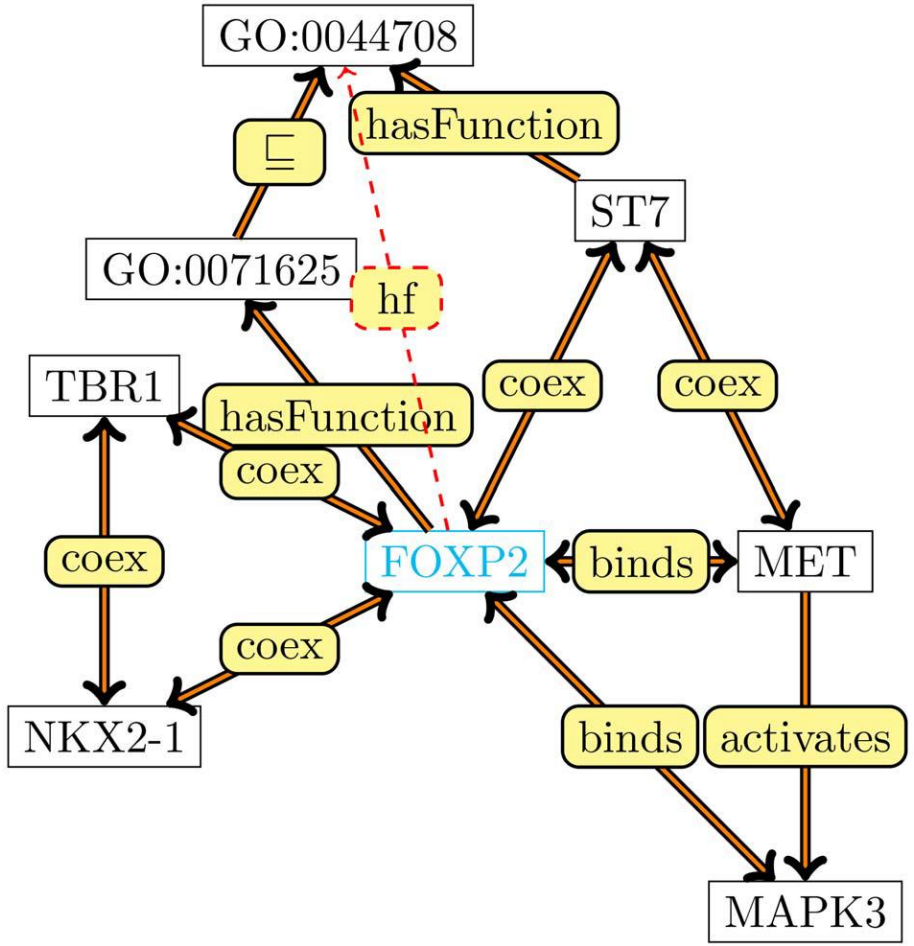
GRAPHS FROM ONTOLOGIES

- Graphs generated from ontologies interact with graph-based representations of data
  - In particular using the Resource Description Framework (RDF)
- Nodes represent entities within a domain
- Edges represent the relations between the nodes
- Called *knowledge graphs*
  - Correspond to a subset of the formalism underlying OWL
- Ontologies as graphs leads to some loss of the information encoded in axioms which cannot be naturally represented in a graph

KLOWLEDGE GRAPH – BIOINFORMATIC EXAMPLE

A KNOWLEDGE GRAPH CENTERED AROUND PROTEIN–PROTEIN INTERACTIONS AND FUNCTIONS OF FOXP2

- Graph with interactions between proteins
- Associations between
  - Proteins
  - Protein's functions
  - Axioms from the Gene Ontology
- Several ways in which such a graph could have been represented in OWL and then converted into such a graph representation using axiom patterns
- The edge between *FOXP2* and *GO:0071625* from the axiom *FOXP2* ∃ hasFunction .GO:0071625
- The dashed edge between *FOXP2* and GO:0044708 is an edge that would be generated through entailment based on these axioms



SIMILARITY MEASURES IN ONTOLOGIES

NEED MEASURE SIMILARITY BETWEEN REPRESENTED CLASSES / ENTITIES / ETC.

SEMANTIC SIMILARITY MEASURES

- In computer science applications it is useful to determine how similar two concepts are
- Measures that compute similarity between concepts
- Used to compare
  - Words / Terms in natural language texts / Entities / Ontology classes
- Used as unsupervised methods for association prediction
  - as features in supervised learning models or in clustering algorithms
- In ontology similarity compute between
  - Classes / Individuals / Annotated entities

SIMILARITY FUNCTION

- A function $sim : D \times D$ is a similarity function on a domain D if it is non-negative ($sim(x, y) \leq 0$), symmetric ($sim(x, y) = sim(y, x)$) and if self similarity yields the highest similarity values within the domain ($sim(x, x) = \max D$), or—as a weaker version—if self-similarity is higher than similarity to any other domain entity ($sim(x, x) > sim(x, y)$).

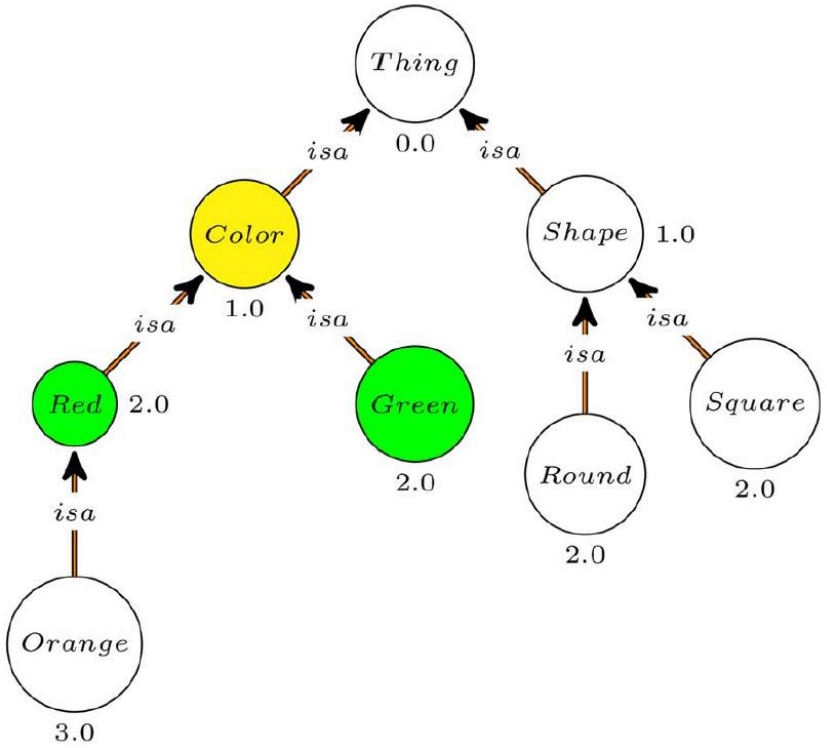
SIMILARITY RADA

$$sim_{Rada}(x, y) = \frac{1}{dist_{SP}(x, y) + 1}$$

- *simRada*
- A simple similarity measure
- Based on the shortest path between two nodes in the graph
- Useful only if every edge has constant distance

SIMILARITY RADA - PROBLEM EXAMPLE

- `is-a` edges order classes from general to more specific
- ***simRada*(*Color*, *Shape*)** and ***simRada*(*Red*, *Green*)** have same value
- In many applications *Red* and *Green* should be more similar than *Color* and *Shape*



CLASS SPECIFICITY

- Many ways to compute class specificity
- For instance compute specificity as a function of the depth
  - # children or information content of a class

- Class specificity is a function $\sigma : C \rightarrow \mathrm{R}$ which meets the condition that for all $x, y \in C$, if $x \sqsubseteq y$ then $\sigma(x) \geq \sigma(y)$. The specificity measure can be defined using only the classes within an ontology (such as measures that consider the number of super-classes a class has, or the distance of a class to the root), or using information such as the number of instances of a class, or the number of annotations of a class within a database.

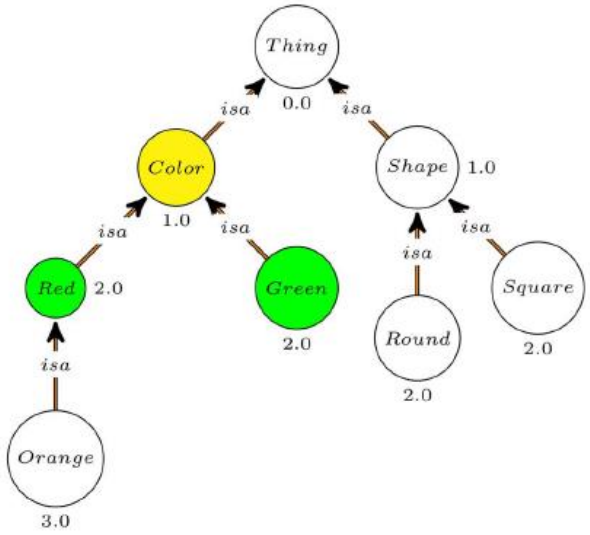
SIMILARITY RESNIK

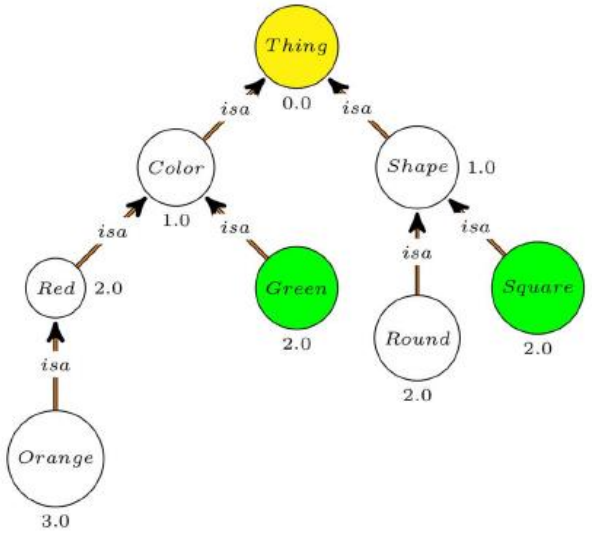
- Resnik measure defines the similarity between classes *x* and *y* as information content of their *most informative common ancestor (MICA)*

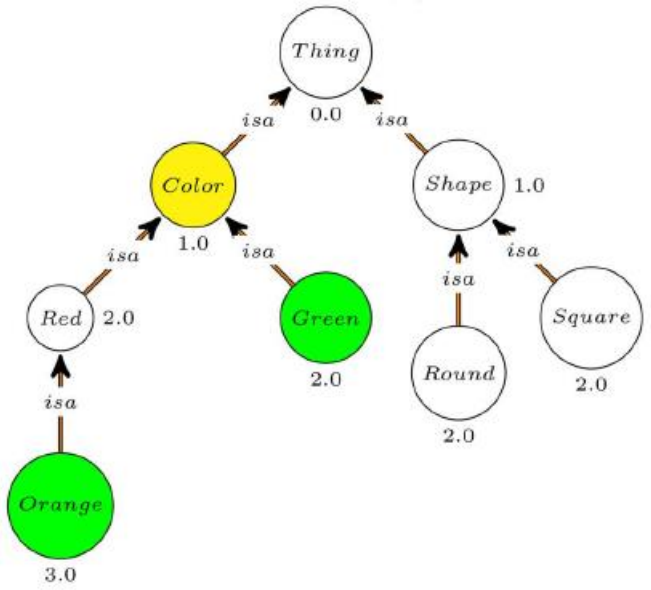
$$Sim_{Resnik}(x, y) = IC(MICA(x, y))$$

SIMILARITY RESNIK – PROBLEM EXAMPLE

- *SimResnik*(*Red*, *Green*) == 1.0 and *SimResnik*(*Color*, *Shape*) == 0.0 although they have the same distance
- The downside of this similarity measure is that it does not take into account the specificity of the compared classes and all classes under the same MICA will have the same similarity value
- *SimResnik*(*Green*, *Square*) == 0.0 == *SimResnik*(*Color*, *Shape*)
- *SimResnik*(*Red*, *Green*) == *SimResnik*(*Orange*, *Green*) == 1.0







LIN'S MEASURE

- Considers information content of the compared classes
- *SimLin*(*Red*, *Green*) == 0.5 whereas *SimLin*(*Orange*, *Green*) == 0.4

$$Sim_{Lin}(x, y) = \frac{2 \cdot IC(MICA(x, y))}{IC(x) + IC(y)}$$

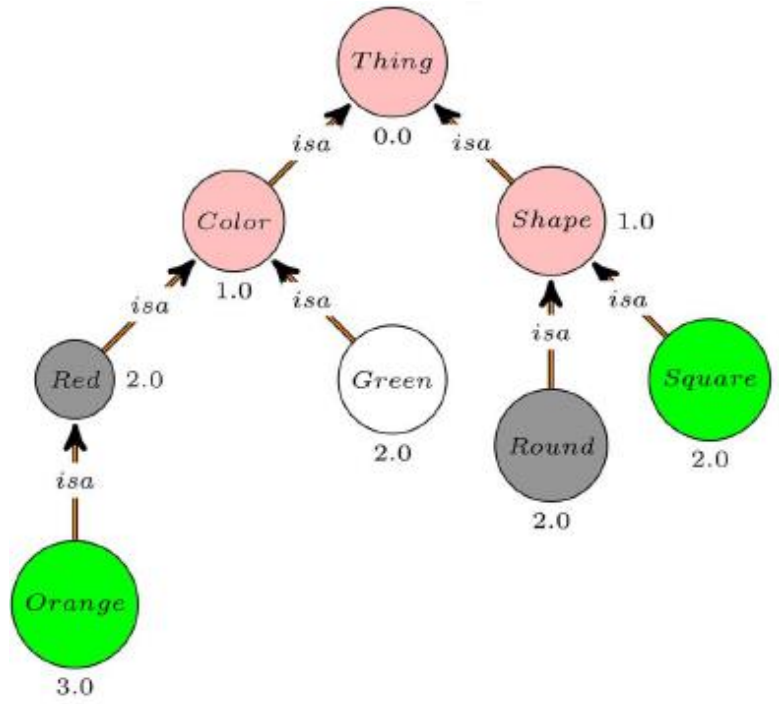
SETS OF CLASSES SIMILARITY

- In two instances or entities in ontology, we need to compare sets of classes
- Problem example: Compute the similarity of the set of all Gene Ontology annotations of one protein with the set of all Gene Ontology annotations of a second protein. Determine the similarity between two sets of classes *A* and *B*.
  - We can compute the pairwise similarities between all pairs of classes (*a*, *b*) such that $a \in A$ and $b \in B$, and then combine similarity values according to some combination strategy (such as computing the average)
  - We can directly define a similarity measure between the two sets *A* and *B* using a set similarity measure

JACCARD SIMILARITY

$$Sim_{Jaccard}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

- For instance - use the Jaccard index between the two sets
- Semantic similarity with respect to superclass axioms
- If $C \sqsubseteq D$ and $C \in X$ then $D \in X$
- Propagation of ontology classes for computing the similarity between a square-and-orange thing and a round-and-red thing
- Set similarity can also incorporate class specificity, such as the weighted Jaccard index in the SimGIC measure



SEMANTIC SIMILARITY MEASURES - SUMMARY

- Difficult to choose the right measure for a particular application
- Similarity measures behave differently depending on their applications
- Using different similarity measures will result in different performance
- Not immune to biases in data and different similarities
  - May react to the biases differently
- Cannot easily be extended to new languages since they require specific loss functions to be designed which may prove challenging for some languages

EMBEDDING

- Structure-preserving map from one mathematical structure to another
- Second structure may enable different or additional operations which are not possible in the first structure
- Take ontologies or graphs that are discrete entities and map them into a continuous space
- Operate on continuous data => Machine learning or continuous optimization algorithms may be applicable
- Natural similarity measures between real-valued vectors such as the cosine similarity or other distance measures and metrics
- Many structures in which ontologies may be embedded
  - axioms within the natural numbers
  - real-valued vector spaces R^n (can be applyed optimization and machine learning algorithms)
- Embedding ontologies approaches classification based on what aspects of the ontologies are preserved the space
  - graph-based
  - syntactic
  - model-theoretic

GRAPH-BASED ONTOLOGY EMBEDDINGS

- Preserve a graph structure within R^*n*
- Form of graph embeddings based on random walks
  - Graphs generated from ontologies
  - Random walks used to explore the neighborhood of each node in the graph
  - The set of walks is used as the basis of the embeddings
- DeepWalk
  - Method for learning graph embeddings through random walks
  - Generates a corpus of sentences (i.e. sequences of nodes in the graph)
  - Applies Word2Vec on the resulting corpus => obtain embedding vectors
  - Include labeled edges extends algorithm for knowledge graphs

RANDOM WALKS EXAMPLE

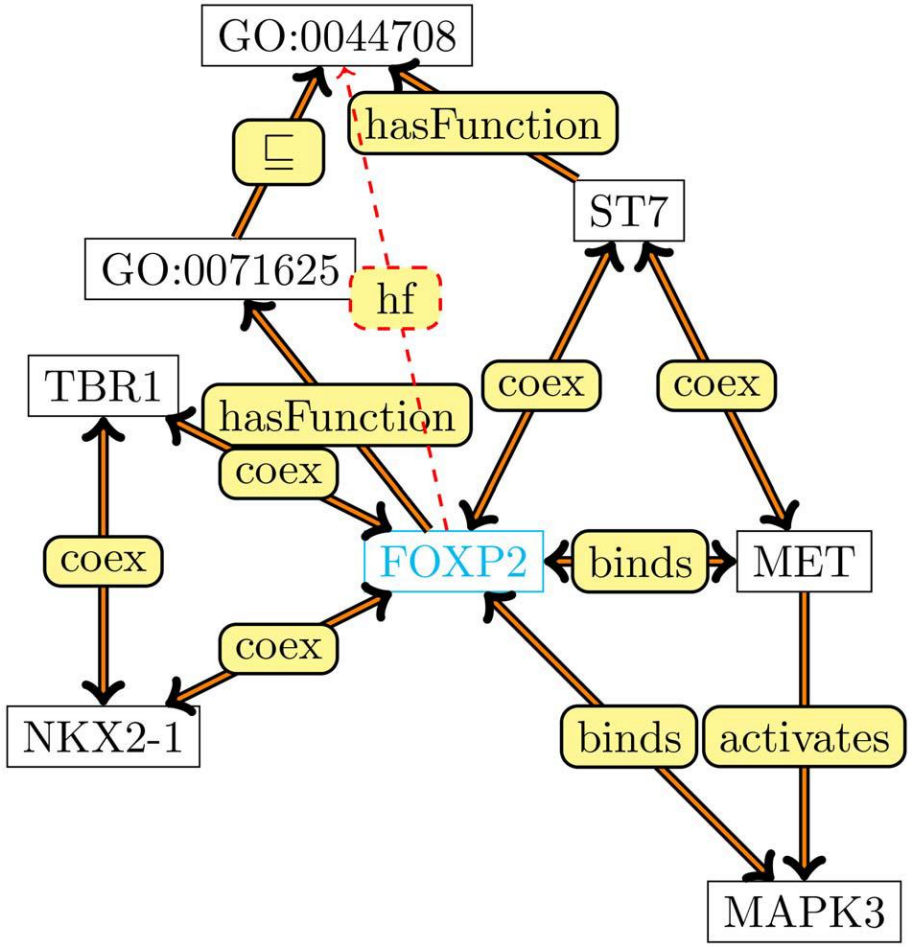
- Generate sentences such as

  FOXP2 cooex ST7 hasFunction GO:0044708...

  FOXP2 hasFunction GO:0071625 is-a GO:0044708...

  and Word2Vec will then embed each node and edge label while preserving co-occurrence relations within this corpus

- Node2Vec
  - modified model - explore the original graph through biased random walks
  - can force walks to remain within a certain distance of the origin node



SYNTACTIC ONTOLOGY EMBEDDINGS

- Syntactic embeddings embed ontologies in R^n
- Considering only the set of axioms without creating an intermediate graph-based representation
- Advantage
  - Able to use all axioms in the ontology
  - Not limited to particular axiom types or expressivity of the formal language

SYNTACTIC ONTOLOGY EMBEDDINGS – ONTO2VEC

- Generates embeddings for ontology classes and instances based on logical axioms that define the semantics
- Treats each axiom as a sentence + embeds the set of axioms using the Word2Vec language model
- Allows embed ontologies directly based on their axioms while considering all axiom types
- Opa2Vec - Extends Onto2Vec
  - Include OWL annotation properties
  - Combines the corpus generated from the asserted and entailed logical axioms in Onto2Vec with a corpus generated from selected annotation properties

EMBEDDINGS AS SEMANTIC SIMILARITY MEASURES IN ML

- Embeddings - representations of the symbols in ontologies —vectors in R^n
- Visualisation using dimensionality reduction techniques
  - Principal component analysis
- Use any kind of similarity or distance measure applicable to real-valued vectors
  - Cosine similarity
  - Euclidean distance
- Example from Gene Ontology: Determine whether two proteins are (functionally) similar
  - Similarity can then be used to predict interactions between proteins based on the hypothesis that similar proteins are more likely to interact

ML TO APROXIMATE FUNCTIONS

- Take more than one embedding as input
- Use these functions to predict relations between the entities that were embedded

- Learn similarity measures between ontology entities

- One of the used NN architecture for similarity learning is Siamese neural network architecture
- Can be used to identifying drug–target interactions

WHY ONTOLOGIES + MACHINE LEARNING?

ONTOLOGIES HELPS IN DESIGN AND LEARNING PROCESS

ONTOLOGIES AS CONSTRAINTS

- Ontologies embeddings
  - Useful technique to make information in ontologies available as background knowledge
  - Define similarity measures
  - To learn features for machine learning models
- Axioms in the ontology used to constrain the output of a function
- Ontologies are used in task to predict whether some entity has a relation with one or more ontology classes
  - Predicting gene–disease or drug–disease associations (using disease ontologies as output)

HIERARCHICAL RELATIONSHOP AS A CONSTRAINT

- Satisfy the hierarchical constraints imposed by the ontologies in the output space
  - If an entity e is predicted to be associated with a class C, and that class C is a subclass of D, then e must also be associated with D
  - Similar constraints arise from other axioms in the ontology

APPROACHES AS A CONSTRAINT IN HIERARCHICAL MODELS

- Flat classification
  - This approach employs the constraints imposed by the ontology independently from the training or prediction process
- Local classifications
  - Predictions are made starting from the most general classes first and then moving to more specific ones
  - Stopping the prediction process once classes are predicted as negative
- Local classifiers - disadvantages
  - All classification models are trained independently from each other
  - During the prediction process errors will propagate from general classes to more specific ones

HIERARCHICAL RELATIONS AS A CONSTRAINT: GLOBAL HIERARCHICAL CLASSIFIERS

- Include the hierarchical constraints during training of a machine learning model and also during prediction
- Output labels are forced to be consistent with the ontology axioms
- Advantages of these models
  - Take the semantics into account during training and therefore potentially reduce the search space
  - Can exploit dependencies between classes during training and prediction
- Disadvantage
  - Increased complexity of these classifiers

ANOTHER FUNCTION PREDICTION METHODS

- Dedicated machine learning methods that rely on the ontology structure during training and testing models
  - Structured Support Vector Machine (SSVM) generalizes a SVM -> allowed different structures as output (sets, trees, graphs, sequences, ...)
  - SSVMs utilize similarity measures between the structured objects (tree similarity or graph similarity) in loss functions
  - SSVMs use a cutting plane method to significantly reduce the number of constraints that need to be examined

FUNCTION PREDICTION USING NEURAL NETWORKS

- Using different neural network architectures that use ontologies as part of their structure
- Several methods developed that specifically incorporate the ontology structure as part of neural networks
- DeepGO – based on Gene Ontology for its predictions
  - Use a convolutional NN for feature learning
  - Use sigmoid functions as classifiers
  - The output of the sigmoid classifier for a class C would be less than the sigmoid output for any of its superclasses
  - If $C \sqsubseteq D$ then $\sigma(C) \leq \sigma(D)$
  - Hierarchical classifier significantly improved prediction performance compared to classifiers that do not consider ontology structure
- There are several further methods that incorporate hierarchical constraints in artificial NN (mostly variations of the ontology-based methods)

NN ARCHITECTURE BASED ON ONTOLOGY STRUCTURE

- Small linear layer of neurons for each class
- Connections according to the ontology's subclass hierarchy

- Ontologies can be used to make the inner workings of NN "visible"
- Ontologies can be used to turn blackbox prediction models into interpretable models

ENHANCEMENT ML WITH ONTOLOGY AND TEXT

- An advantage of embeddings that rely on language models such as Word2Vec is that they can easily be combined with information in natural language
- Natural language texts can have information that is complementary to the structured information in ontologies
- Combining structured information with text improve predictive model performance
- Most syntactic and graph-based approaches do not interpret negation as constraints => can not use negation to restrict search in models based on graph approaches only

PUBLICATIONS RELATED TO MACHINE LEARNING & ONTOLOGY

- https://academic.oup.com/bib/article/22/4/bbaa199/5922325
- https://link.springer.com/chapter/10.1007%2F978-3-030-58861-8_10
- https://dl.acm.org/doi/10.1145/3308558.3313636

QUESTIONS? AND ANSWERS (MAYBE)

THANK YOU FOR YOUR ATTENTION