Query Learning and Attribute Exploration

Supervised learning

Input: a training set divided into (for example) two classes w. r. t. a certain target property:

- positive examples;
- negative examples.

Build a classifier that determines whether a previously unseen object has the target property.

Learning with queries (Angluin 1988)

Input: an oracle capable of answering queries of certain predefined types concerning a target property.

Build a classifier that determines whether a previously unseen object has the target property.

Types of queries

Membership query: Does the object have the target property?

Types of queries

Membership query: Does the object have the target property?

Equivalence query: Does the hypothesis *H* accurately describe the set of objects with the target property? If not, the oracle must provide

- a positive counterexample that has the target property, but is not covered by the hypothesis

or

- a negative counterexample that doesn't have the property, but satisfies the hypothesis.

Types of queries

Subset query: Does the hypothesis *H* describe only objects with the target property?

- If not, provide a negative counterexample.

Superset query: Does the hypothesis *H* describe all the objects with the target property?

- If not, provide a positive counterexample.

Learning binary patterns

- A binary pattern is a nonempty string consisting of 0, 1, and variables from a countably infinite alphabet X.
- A pattern p defines the language $L(p)$ consisting of all words that can be obtained by substituting nonempty binary strings for variables.
- For example, the language of the pattern $x_1 0 x_2 0 x_1$ includes the strings 10101 and 010111001, but not 10100 or 1001.

Learning binary patterns

- A hint: I have guessed one of the 2^k variable-free patterns of length k.

Learning binary patterns

- A hint: I have guessed one of the 2^k variable-free patterns of length *k*.
- Membership query: Is word *w* in the language?
  - No.

Learning binary patterns

- A hint: I have guessed one of the 2^k variable-free patterns of length *k*.
- Membership query: Is word *w* in the language?
  - No.
- Equivalence query: Is *p* the right pattern?
  - No: give a negative counterexample.

Learning binary patterns

- A hint: I have guessed one of the 2^k variable-free patterns of length *k*.
- Membership query: Is word *w* in the language?
  - No.
- Equivalence query: Is *p* the right pattern?
  - No: give a negative counterexample.
- Subset query: Is *p* less general than the target pattern?
  - No: give a negative counterexample.

Learning binary patterns

- A hint: I have guessed one of the 2^k variable-free patterns of length *k*.
- Membership query: Is word *w* in the language?
  – No.
- Equivalence query: Is *p* the right pattern?
  – No: give a negative counterexample.
- Subset query: Is *p* less general than the target pattern?
  – No: give a negative counterexample.
- Each of the above queries excludes at most one of 2^k patterns.

No polynomial-time algorithm.

Learning binary patterns with superset queries

1. Determine the length *k* of the pattern.

2. Determine the constants in the pattern.

3. Determine identical variables in the pattern.

Learning binary patterns with superset queries

1. Determine the length k of the pattern.
   – Query about patterns x_1x_2, $x_1x_2x_3$, etc., until the answer is "No".
2. Determine the constants in the pattern.
3. Determine identical variables in the pattern.

Learning binary patterns with superset queries

1. Determine the length k of the pattern.
   - Query about patterns x_1x_2, $x_1x_2x_3$, etc., until the answer is "No".
2. Determine the constants in the pattern.
   - For $i = 1, 2, \ldots, k$, query about the patterns $x_1\ldots x_{i-1}0x_{i+1}\ldots x_k$ and $x_1\ldots x_{i-1}1x_{i+1}\ldots x_k$.
3. Determine identical variables in the pattern.

Learning binary patterns with superset queries

1. Determine the length k of the pattern.
   - Query about patterns x_1x_2, $x_1x_2x_3$, etc., until the answer is "No".
2. Determine the constants in the pattern.
   - For $i = 1, 2, \ldots, k$, query about the patterns $x_1 \ldots x_{i-1} 0 x_{i+1} \ldots x_k$ and $x_1 \ldots x_{i-1} 1 x_{i+1} \ldots x_k$.
3. Determine identical variables in the pattern.
   - For each pair of positions $i < j$ of variables in the pattern, query about the pattern $x_1 \ldots x_{i-1} y x_{i+1} \ldots x_{j-1} y x_{j+1} \ldots x_k$.

Learning binary patterns with superset queries

1. Determine the length k of the pattern.
   - Query about patterns x_1x_2, $x_1x_2x_3$, etc., until the answer is "No".
2. Determine the constants in the pattern.
   - For $i = 1, 2, \ldots, k$, query about the patterns $x_1\ldots x_{i-1}0x_{i+1}\ldots x_k$ and $x_1\ldots x_{i-1}1x_{i+1}\ldots x_k$.
3. Determine identical variables in the pattern.
   - For each pair of positions $i < j$ of variables in the pattern, query about the pattern $x_1\ldots x_{i-1}yx_{i+1}\ldots x_{j-1}yx_{j+1}\ldots x_k$.

Polynomial-time algorithm!

Formal Concept Analysis

Formal context (G, M, I)

- a set G of objects
- a set M of attributes
- objects are described with attributes via a binary relation $I \subseteq G \times M$

A formal context

Europe	EU	Euro	Schengen	NATO	Monarchy
Italy	×	×	×	×	
United Kingdom	×			×	×
Poland	×		×	×	
Denmark	×		×	×	×
Norway			×	×	×
Russia					
Spain	×	×	×	×	×
Turkey				×	

Formal Concept Analysis

Derivation operators

For $A \subseteq G$ and $B \subseteq M$:

- $A' = \{m \in M \mid \forall g \in A: gIm\}$
- $B' = \{g \in G \mid \forall m \in B: gIm\}$

For $g \in G$ and $m \in M$, the set $\{g\}'$ is called an object intent and the set $\{m\}'$ is called an attribute extent.

$(\cdot)'' : 2^M \to 2^M$ is a closure operator.

Derivation operators

Europe	EU	Euro	Schengen	NATO	Monarchy
Italy	×	×	×	×	
United Kingdom	×			×	×
Poland	×		×	×	
Denmark	×		×	×	×
Norway			×	×	×
Russia					
Spain	×	×	×	×	×
Turkey				×	

$$\{\text{EU}, \text{Euro}, \text{Schengen}\}' = \{\text{Italy}, \text{Spain}\}$$

Derivation operators

Europe	EU	Euro	Schengen	NATO	Monarchy
Italy	×	×	×	×	
United Kingdom	×			×	×
Poland	×		×	×	
Denmark	×		×	×	×
Norway			×	×	×
Russia					
Spain	×	×	×	×	×
Turkey				×	

$$\{\text{Italy, Spain}\}' = \{\text{EU, Euro, Schengen, NATO}\}$$

Formal Concept Analysis

Implication $A \to B$ $(A, B \subseteq M)$

- An attribute subset $X \subseteq M$ is a model of $A \to B$ if A is not a subset of X or B is a subset of X.
- $A \to B$ holds in the context if $A' \subseteq B'$.

- X is a model of an implication set L if it is a model of every implication from L.
- Two implication sets are equivalent if they have the same models.
- Among equivalent implication sets, there is a particular one called the canonical (or Duquenne–Guigues) basis. It is minimal in the number of implications among all equivalent implication sets.

Canonical basis

Europe	EU	Euro	Schengen	NATO	Monarchy
Italy	×	×	×	×	
United Kingdom	×			×	×
Poland	×		×	×	
Denmark	×		×	×	×
Norway			×	×	×
Russia					
Spain	×	×	×	×	×
Turkey				×	

EU → NATO
Monarchy → NATO

Schengen → NATO
Euro → EU, Schengen, NATO

Computing implications

- If *M* is finite, the canonical basis is finite, too.

- If *G* is also finite, we can compute the canonical basis, e.g., using the NextClosure algorithm (Ganter 1984).

- However, if *G* is infinite, it is not possible to work with the entire context directly.

Learning implications with queries

- Forget about the context—talk to an oracle to compute the implication set L^*.
- A set of implications describes a set of *models*— attribute combinations that satisfy these implications.

Membership queries: Is $A \subseteq M$ a model of L^*?

Equivalence queries: Is an implication set L equivalent to L^*?

- If not, the oracle must provide a counterexample: a set that is a model of L^*, but not of L (positive counterexample), or vice versa (negative counterexample).

Polynomial-time algorithm (Angluin *et al.* 1992)

Equivalence queries: Is an implication set L equivalent to L^*?

- If not, the oracle must provide a counterexample: a set that is a model of L^*, but not of L (positive counterexample), or vice versa (negative counterexample).

How to handle a counterexample X?

If X is positive and it doesn't satisfy some $A \rightarrow B$ from L, weaken $A \rightarrow B$ by replacing it with $A \rightarrow B \cap X$.

If X is negative, strengthen L by replacing some $A \rightarrow B$ from L with $A \cap X \rightarrow B$ or by adding a new implication $X \rightarrow M$, so as to exclude X from the set of models of L.

Polynomial-time algorithm (Angluin *et al.* 1992)

An example for $M = \{a, b, c, d\}$:

- Initial hypothesis: the empty implication set—everything is possible!
- Equivalence query returns a negative counterexample $\{a, b, c\}$.
- New hypothesis: $\{a, b, c\} \to M$.

Polynomial-time algorithm (Angluin *et al.* 1992)

An example for $M = \{a, b, c, d\}$:

- Current hypothesis: $\{a, b, c\} \rightarrow M$.
- Equivalence query returns a negative counterexample $\{a\}$.
- Strengthen the current hypothesis to exclude $\{a\}$.
- The new hypothesis: $\{a\} \rightarrow M$.

Polynomial-time algorithm (Angluin *et al.* 1992)

An example for $M = \{a, b, c, d\}$:

- Current hypothesis: $\{a\} \rightarrow M$.
- Equivalence query returns a negative counterexample $\{c, d\}$.
- Can we strengthen $\{a\} \rightarrow M$?

Polynomial-time algorithm (Angluin *et al.* 1992)

An example for $M = \{a, b, c, d\}$:

- Current hypothesis: $\{a\} \rightarrow M$.
- Equivalence query returns a negative counterexample $\{c, d\}$.
- Can we strengthen $\{a\} \rightarrow M$?
- Membership query w. r. t. $\{a\} \cap \{c, d\} = \varnothing$.
- Answer: yes!
    - The empty set is a model of L^*.

Polynomial-time algorithm (Angluin *et al.* 1992)

An example for $M = \{a, b, c, d\}$:

- Current hypothesis: $\{a\} \rightarrow M$.
- Equivalence query returns a negative counterexample $\{c, d\}$.
- Can we strengthen $\{a\} \rightarrow M$?
- Membership query w. r. t. $\{a\} \cap \{c, d\} = \varnothing$.
- Answer: yes!
  - The empty set is a model of L^*.
- New hypothesis: $\{a\} \rightarrow M$, $\{c, d\} \rightarrow M$.

Polynomial-time algorithm (Angluin *et al.* 1992)

An example for $M = \{a, b, c, d\}$:

- Current hypothesis: $\{a\} \rightarrow M$, $\{c, d\} \rightarrow M$.
- Equivalence query returns a positive counterexample $\{a, c, d\}$.
- Weaken $\{a\} \rightarrow M$ and $\{c, d\} \rightarrow M$.
- New hypothesis: $\{a\} \rightarrow \{a, c, d\}$, $\{c, d\} \rightarrow \{a, c, d\}$.

Polynomial-time algorithm (Angluin *et al.* 1992)

An example for $M = \{a, b, c, d\}$:

- Current hypothesis: $\{a\} \rightarrow \{a, c, d\}$, $\{c, d\} \rightarrow \{a, c, d\}$.
- Equivalence query returns a positive counterexample $\{a, c\}$.
- Weaken $\{a\} \rightarrow \{a, c, d\}$.
- New hypothesis: $\{a\} \rightarrow \{a, c\}$, $\{c, d\} \rightarrow \{a, c, d\}$.

Polynomial-time algorithm (Angluin *et al.* 1992)

An example for $M = \{a, b, c, d\}$:

- Current hypothesis: $\{a\} \rightarrow \{a, c\}$, $\{c, d\} \rightarrow \{a, c, d\}$.
- Equivalence query returns a negative counterexample $\{a, b, c\}$.
- Membership query w. r. t. $\{c, d\} \cap \{a, b, c\} = \{c\}$.
- Answer: yes!
- New hypothesis: $\{a\} \rightarrow \{a, c\}$, $\{c, d\} \rightarrow \{a, c, d\}$, $\{a, b, c\} \rightarrow M$.

Polynomial-time algorithm (Angluin *et al.* 1992)

An example for $M = \{a, b, c, d\}$:

- Current hypothesis: $\{a\} \rightarrow \{a, c\}$, $\{c, d\} \rightarrow \{a, c, d\}$, $\{a, b, c\} \rightarrow M$.
- Equivalence query returns no counterexamples.
- Success!

Polynomial-time algorithm (Angluin *et al.* 1992)

$\mathcal{L} := \varnothing$
while there is a counterexample X to $\mathcal{L}$ **do** {Equivalence oracle}
  if $\mathcal{L}(X) = X$ **then** {negative counterexample}
    $found :=$ **false**
    for all $A \to B \in \mathcal{L}$ **do**
      $C := A \cap X$
      if $A \neq C$ **and** $C \neq \mathcal{L}^*(X)$ **then** {Membership oracle}
        $\mathcal{L} := \mathcal{L} \setminus \{A \to B\}$
        $\mathcal{L} := \mathcal{L} \cup \{C \to B\}$
        $found :=$ **true**
        exit for
    if not $found$ **then**
      $\mathcal{L} := \mathcal{L} \cup \{X \to M\}$
  else {positive counterexample}
    for all $A \to B \in \mathcal{L}$ such that $A \subseteq X$ and $B \nsubseteq X$ **do**
      $\mathcal{L} := \mathcal{L} \setminus \{A \to B\}$
      $\mathcal{L} := \mathcal{L} \cup \{A \to B \cap X\}$

Polynomial-time algorithm (Angluin *et al.* 1992)

- Computes the canonical basis (Arias and Balcázar 2011)
- Makes $O(m^2n)$ membership and $O(mn)$ equivalence queries
  - m is the size of the basis
  - n is the number of attributes

Attribute exploration

- An alternative technique from formal concept analysis.
- Start with any (possibly empty) set of objects.
- Generate an implication valid in the current subcontext.
- If the implication is not valid in the entire context, provide an object that violates it.
- Go to the next implication, etc.

Follow the canonical basis to ask only questions that are necessary.

Attribute exploration

- An alternative technique from formal concept analysis.
- Start with any (possibly empty) set of objects.

- Generate an implication valid in the current subcontext.
- If the implication is not valid in the entire context, provide an object that violates it.

- Go to the next implication, etc.

This is a superset query:

Do the models of the implication $A \rightarrow B$ include all the models of the target implication set L^*?

Attribute exploration in lexical typology

- Objects are words of different languages.
- Attributes are frames corresponding to individual meanings.
- We want to build a semantic map showing which meanings can be combined together within a single lexeme.

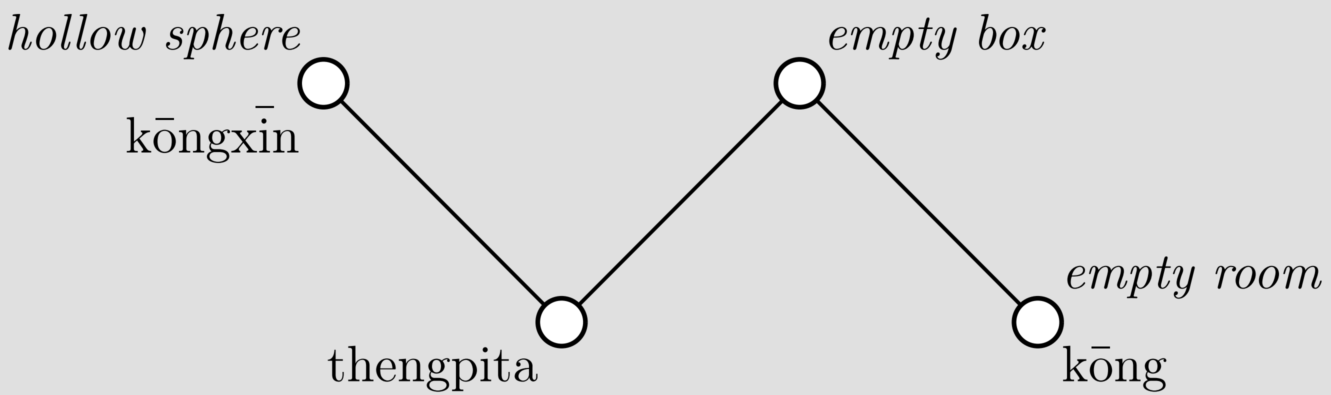
Attribute exploration of the semantic field `empty'

- Start with three words (two Chinese and one Korean) and three frames:

	hollow sphere	*empty box*	*empty room*
kōngxin	×		
kōng		×	×
thengpita	×	×	

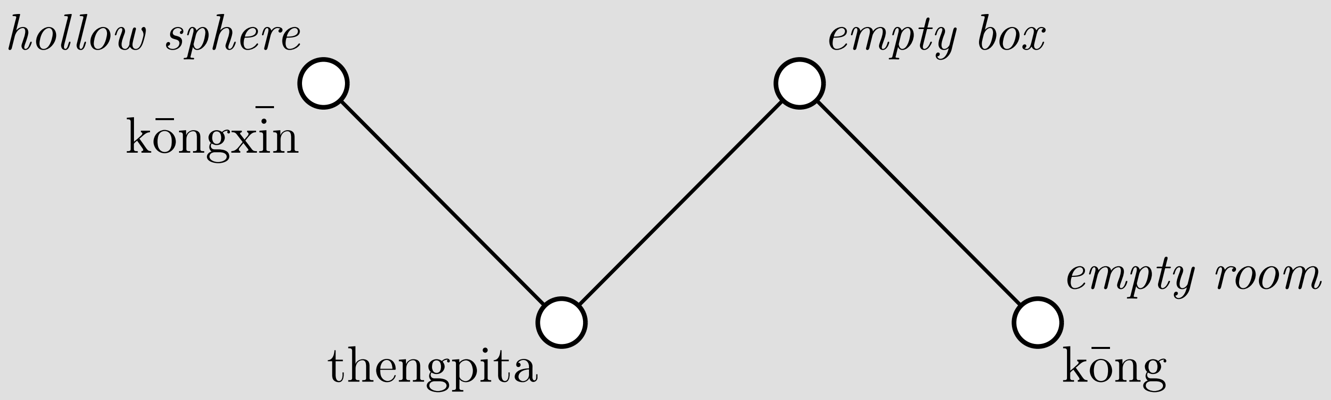
Attribute exploration of the semantic field `empty'

- Start with three words (two Chinese and one Korean) and three frames:



Attribute exploration of the semantic field `empty'

- Start with three words (two Chinese and one Korean) and three frames:



Does every word for *empty room* is also suitable for *empty box*?

Attribute exploration of the semantic field `empty’

Does every word for *empty room* is also suitable for *empty box*?

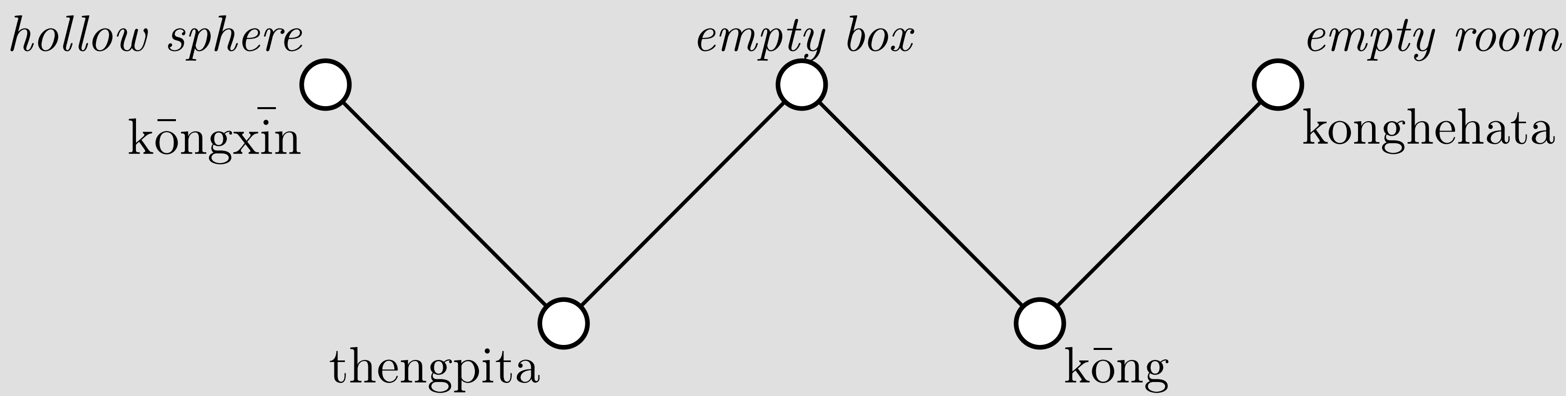
No: there is a counterexample in Korean.

	hollow sphere	*empty box*	*empty room*
kōngxin	×		
kōng		×	×
thengpita	×	×	
konghehata			×

Attribute exploration of the semantic field `empty'

Does every word for *empty room* is also suitable for *empty box*?

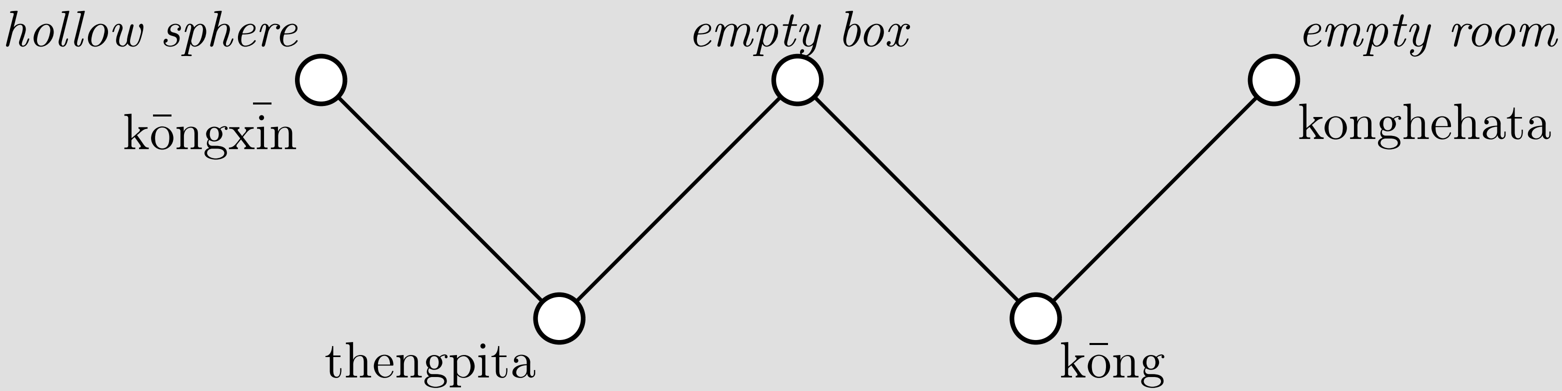
No: there is a counterexample in Korean.



Attribute exploration of the semantic field `empty’

Does every word for *empty room* is also suitable for *empty box*?

No: there is a counterexample in Korean.

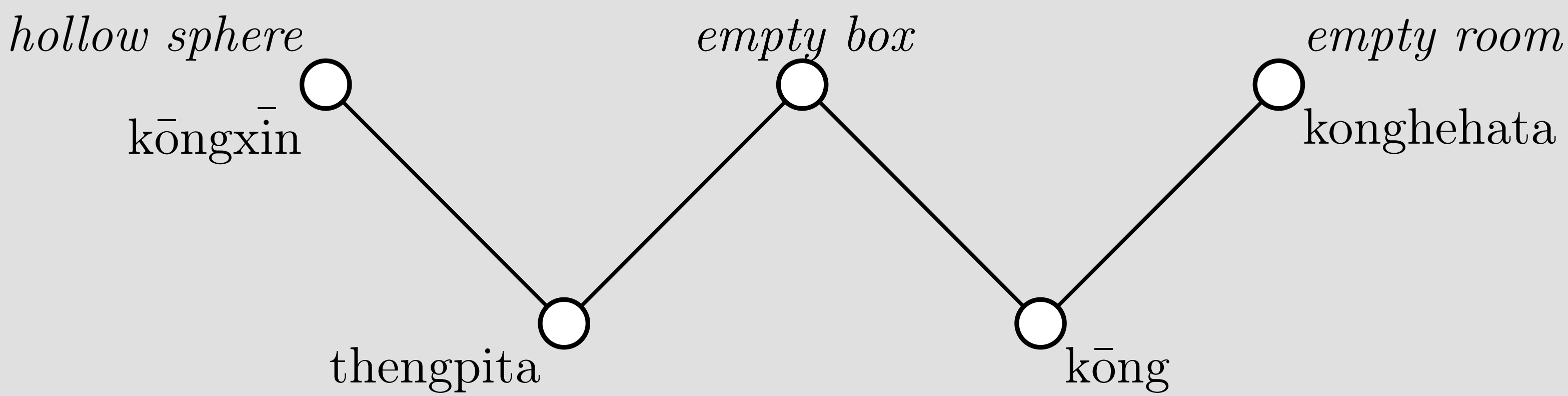


Is there a word used for both *hollow sphere* and *empty room*?

Attribute exploration of the semantic field `empty'

Does every word for *empty room* is also suitable for *empty box*?

No: there is a counterexample in Korean.

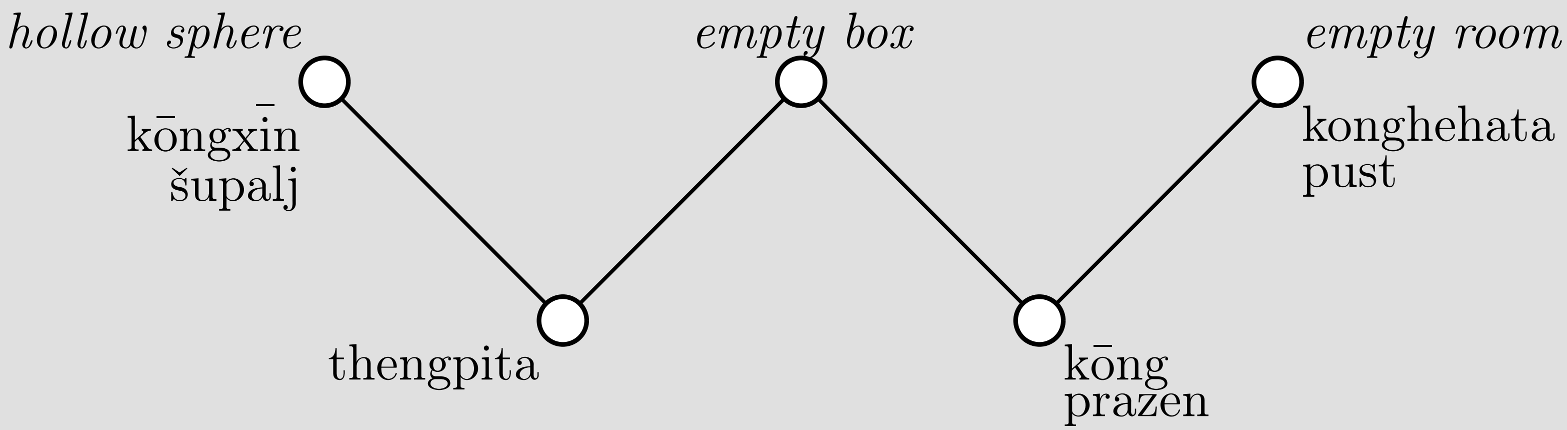


Is there a word used for both *hollow sphere* and *empty room*?

Probably, no.

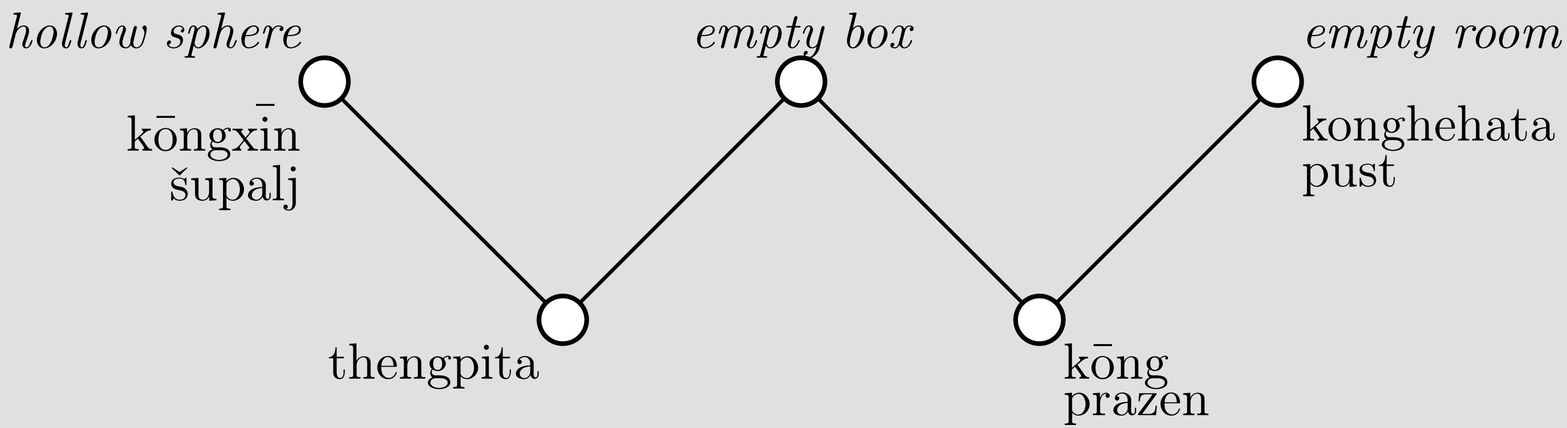
Attribute exploration of the semantic field `empty'

Let us add some Serbian words:



Object exploration of the semantic field `empty'

Let us add some Serbian words:



and run attribute exploration the other way round:

Are all the meanings of "pust" shared by "kōng", "konghehata", and "prazen"?

Object exploration of the semantic field `empty'

Are all the meanings of "pust" shared by "kōng", "konghehata", and "prazen"?

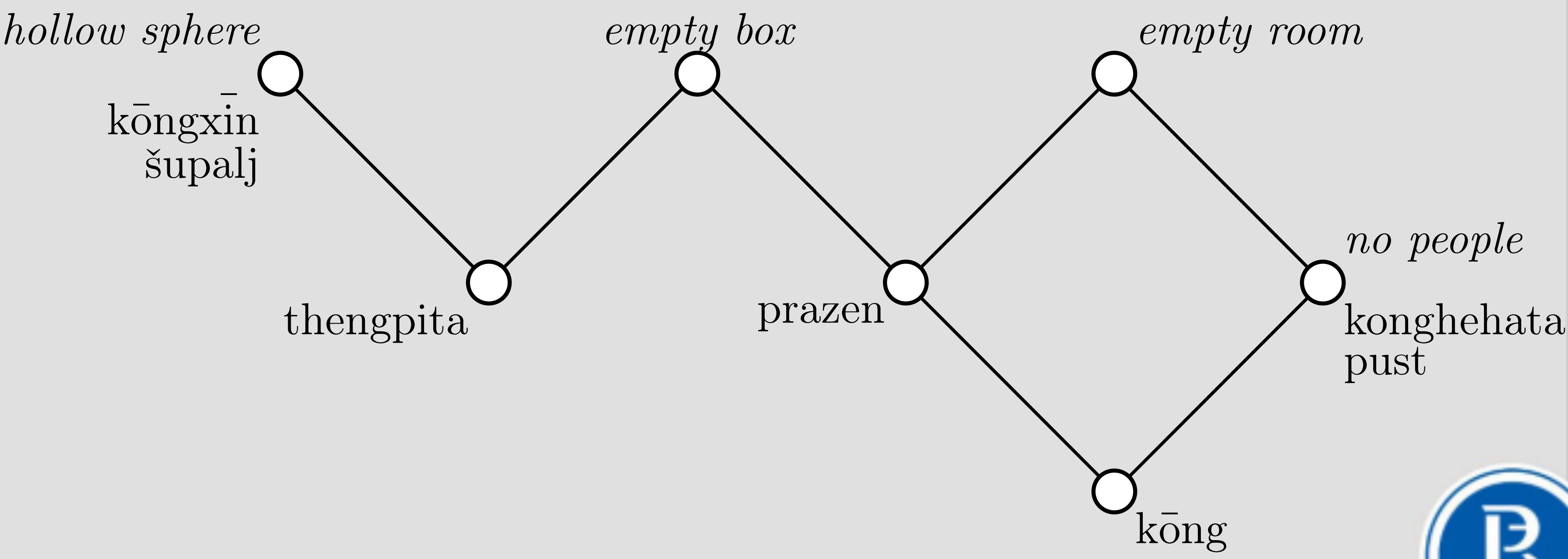
No: "prazen" is not used to denote local spaces without people (but only those without inanimate objects).

	no people
kōngxin	
kōng	×
thengpita	
konghehata	×
šupalj	
prazen	
pust	×

Object exploration of the semantic field `empty'

Are all the meanings of "pust" shared by "kōng", "konghehata", and "prazen"?

No: "prazen" is not used to denote local spaces without people (but only those without inanimate objects).



Many possible extensions

- Background knowledge
- Exceptions
- Symmetries
- Incompletely specified examples

- First-order rule exploration
- Exploration for description logics
- Concept exploration
- Exploration of noisy, fuzzy, triadic data, etc.

- Collaborative exploration