Concept Learning
General to Specific ordering

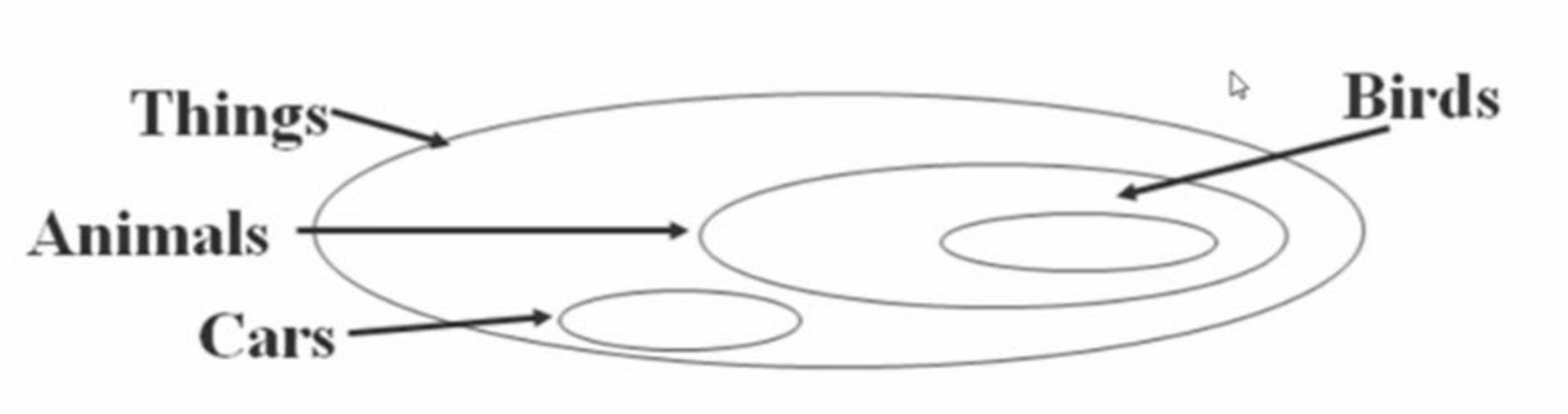
Lecture 2

Outline

- Concept Learning
- Consistent Hypothesis
- Version Space
- List-Then-Eliminate Algorithm
- Find-S Algorithm
- Candidate Elimination Algorithm

Concept

- A concept is a subset of objects or events defined over a larger set e.g we refer to a set of everything as a set of things. Animals are a subset of things, and birds are a subset of animals
- In more technical terms, a concept is a Boolean valued function defined over the larger set



Concept Learning

- Given a set of examples labeled as members or non-members of a concept, concept learning consists of automatically inferring the general definition of this concept.

- In other words, concept learning consists of approximating a Boolean-valued function from training examples of input and output

Example of concept Learning task

⇒ **Concept:** Good Days for Watersports (values: Yes, No)

⇒ **Attributes/Features:**

Sky (values: Sunny, Cloudy, Rainy)
AirTemp (values: Warm, Cold)
Humidity (values: Normal, High)
Wind (values: Strong, Weak)
Water (Warm, Cool)
Forecast (values: Same, Change)

⇒ **Example of a Training Point:**

<Sunny, Warm, High, Strong, Warm, Same, Yes>

Day	Sky	Airtemp	Humidity	Wind	Water	Forecast	WaterSport
1.	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2.	Sunny	Warm	High	Strong	Warm	Same	Yes
3.	Rainy	Cold	High	Strong	Warm	Change	No
4.	Sunny	Warm	High	Strong	Cool	Change	Yes

Chosen Hypothesis Representation:

Conjunction of constraints on each attribute where:

- "?" means "any value is acceptable"
- "0" means "no value is acceptable"

Example of a hypothesis: <?,Cold,High,?,?,?>
(If the air temperature is cold and the humidity high then it is a good day for water sports)

Concept Learning as Search

- Concept Learning can be viewed as the task of searching through a large space of hypothesis implicitly defined by hypothesis representation.

General and specific ordering

- In each specific concept learning tasks, there exists a hypothesis space H with a set of all possible hypothesis
- Each hypothesis is described by a tuple of constraints on the attributes of the concept. The constraints may be
  - ? Which is a general value
  - Φ which is a specific value
- Convert the collection of hypothesis into a form where they indicate the result of the task from a very specific task to a general task represented as:

<φ,φ,φ> to <?,?,?>

Consistent Hypothesis

An hypothesis h is **consistent** with a set of training examples D iff $h(x) = c(x)$ for each example in D

$$Consistent(h, D) \equiv (\forall \langle x, c(x)\rangle \in D)\ h(x) = c(x))$$

Consistent Hypothesis

An hypothesis h is **consistent** with a set of training examples D iff $h(x) = c(x)$ for each example in D

$$Consistent(h, D) \equiv (\forall \langle x, c(x) \rangle \in D)\ h(x) = c(x))$$

Example	Citations	Size	InLibrary	Price	Editions	Buy
1	Some	Small	No	Affordable	One	No
2	Many	Big	No	Expensive	Many	Yes

h1 = (?, ?, No, ?, Many)

Consistent Hypothesis

An hypothesis h is **consistent** with a set of training examples D iff $h(x) = c(x)$ for each example in D

$$Consistent(h, D) \equiv (\forall \langle x, c(x)\rangle \in D)\ h(x) = c(x))$$

Example	Citations	Size	InLibrary	Price	Editions	Buy
1	Some	Small	No	Affordable	One	No
2	Many	Big	No	Expensive	Many	Yes

h1 = (?, ?, No, ?, Many) – Consistent

Consistent Hypothesis

An hypothesis h is **consistent** with a set of training examples D iff $h(x) = c(x)$ for each example in D

$$Consistent(h, D) \equiv (\forall \langle x, c(x) \rangle \in D)\ h(x) = c(x))$$

Example	Citations	Size	InLibrary	Price	Editions	Buy
1	Some	Small	No	Affordable	One	No
2	Many	Big	No	Expensive	Many	Yes

h1 = (?, ?, No, ?, Many) **– Consistent**

h2 = (?, ?, No, ?, ?)

Consistent Hypothesis

An hypothesis h is **consistent** with a set of training examples D iff $h(x) = c(x)$ for each example in D

$$Consistent(h, D) \equiv (\forall \langle x, c(x) \rangle \in D)\ h(x) = c(x))$$

Example	Citations	Size	InLibrary	Price	Editions	Buy
1	Some	Small	No	Affordable	One	No
2	Many	Big	No	Expensive	Many	Yes

h1 = (?, ?, No, ?, Many) **– Consistent**

h2 = (?, ?, No, ?, ?) **– Not Consistent**

Version Space

- The version space $VS_{H,D}$ is the subset of the hypothesis from H *consistent* with the training example in D

$$VS_{H,D} \equiv \{h \in H \mid Consistent(h, D)\}$$

List-Then-Eliminate Algorithm

Version space as list of hypotheses

1. *VersionSpace* $\leftarrow$ a list containing every hypothesis in *H*
2. For each training example, $\langle x, c(x)\rangle$ Remove from *VersionSpace* any hypothesis *h* for which $h(x) \neq c(x)$
3. Output the list of hypotheses in *VersionSpace*

List-Then-Eliminate

- F1 - > A, B
- F2 - > X, Y
- **Instance Space:** (A, X), (A, Y), (B, X), (B, Y) – **4 Examples**
- **Hypothesis Space:** (A, X), (A, Y), (A, **ø**), (A, ?), (B, X), (B, Y), (B, **ø**), (B, ?), (**ø**, X), (**ø**, Y), (**ø**, **ø**), (**ø**, ?), (**?**, X), (**?**, Y), (**?**, **ø**), (**?**, ?) - **16 Hypothesis**
- **Semantically Distinct Hypothesis :** (A, X), (A, Y), (A, ?), (B, X), (B, Y), (B, ?), (**?**, X), (**?**, Y (**?**, ?), (**ø**, **ø**) – **10**

List-Then-Eliminate Algorithm

- Version Space: (A, X), (A, Y), (A, ?), (B, X), (B, Y), (B, ?), (?, X), (?, Y) (?, ?), (ø, ø),
- Training Instances

F1	F2	Target
A	X	Yes
A	Y	Yes

List-Then-Eliminate Algorithm

- Version Space: (A, X), (A, Y), (A, ?), (B, X), (B, Y), (B, ?), (?, X), (?, Y) (?, ?), (ø, ø),
- Training Instances

F1	F2	Target
A	X	Yes
A	Y	Yes

- Consistent Hypothesis are: (A, ?), (?, ?)

Limitations

- The hypothesis space must be finite

Ordering with Example

Let’s consider the example as seen in the previous concept learning. Where the target concept to be learnt is the “Days on which the person enjoys WaterSport”

Day	Sky	Airtemp	Humidity	Wind	Water	Forecast	WaterSport
1.	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2.	Sunny	Warm	High	Strong	Warm	Same	Yes
3.	Rainy	Cold	High	Strong	Warm	Change	No
4.	Sunny	Warm	High	Strong	Cool	Change	Yes

Ordering with Example

The possible values of each of these attributes are:

Sky	Sunny, Rainy, Cloudy
Airtemp	Warm, Cold
Humidity	Normal, High
Wind	Strong, Weak
Water	Warm, Cool
Forecast	Same, Change

The function EnjoySport() which is the concept to be learnt can be denoted as a boolean function as:

C(x) => { Yes, No }

Where C(x) is the EnjoySport values

Applying Find-S Algorithm

The FIND-S Algorithm is used to find the maximally specific hypothesis i.e. the form in which it satisfies all the given hypothesis for the required result.

Note: FIND-S Algorithm only considers positive results / results in favour and ignores negative results.

In the given example, the 4 given hypothesis **h1**, **h2**, **h3**, **h4** are denoted by the results on Day 1,2,3 and 4 respectively. For the FIND-S Algorithm only **h1**,**h2** and **h4** will be considered because **c(x)=>{Yes}** for these hypothesis.

Find-S Algorithm

The **FIND-S** Algorithm:

1. Initialize *h* to the most specific hypothesis in H (hypothesis space / set of hypothesis given)

   $h \leftarrow \langle \varnothing, \varnothing, \varnothing, ..., \varnothing, \varnothing, \varnothing \rangle$ is the most specific hypothesis

2. For each positive training instance x
   a. For each attribute constraint of x in h
      i. If the attribute has the same value as the hypothesis
         1. Do nothing
      ii. Else replace the x by the next general constraint satisfied by x

3. Output hypothesis h

Find-S Algorithm

Lets run through the example and see how the algorithm works

h ← <∅,∅,∅,...,∅,∅,∅> is the most specific hypothesis.

h1 ← <sunny, warm, normal, strong, warm, same>

Results in:

h ← <sunny, warm, normal, strong, warm, same>

Find-S Algorithm

```
h ← <sunny, warm, normal, strong, warm, same>
```

```
h2 ← <sunny, warm, high, strong, warm, same>
```

Results in:

```
h ← <sunny, warm, ?, strong, warm, same>
```

Because the concepts learns that humidity value <normal/high> doesn’t affect the result of C(x) => {Yes} for EnjoySport()

h ← <sunny, warm, ?, strong, warm, same>

h3 ← <rainy, cold, high, strong, warm, same>

Since **h3** is a negative result it isn't considered in FIND-S resulting in the **h** being the same

h ← <sunny, warm, ?, strong, warm, same>

Find-S Algorithm

h ← <sunny, warm, ?, strong, warm, same>

h4 ← <sunny, warm, high, strong, cool, change>

h4 makes the **h** value change because water temp and forecast are different (<cool/warm> and <same/change>) but do not affect the concept learnt

h ← <sunny, warm, ?, strong, ?, ?>

The training is now complete and the hypothesis on completion of FIND-S Algorithm is

h ← <sunny, warm, ?, strong, ?, ?>

Limitations

- There is no way to determine if the hypothesis is consistent throughout the data.
- Inconsistent training sets can actually mislead the Find-S algorithm, since it ignores the negative examples.
- Find-S algorithm does not provide a backtracking technique to determine the best possible changes that could be done to improve the resulting hypothesis.

Candidate Elimination Algorithm

The candidate elimination algorithm uses version spaces and also considers the negative results unlike the FIND-S Algorithm

- Initialize G to be most general hypothesis in H
- Initialize S to be most specific hypothesis in H
- For each training example d do:
    - If d is a positive example
        - Remove from G any hypothesis inconsistent with d.
        - For each hypothesis s in S that is not consistent with d
            - Remove s from S
            - Add to S all minimal generalizations h of s such that
                - h is consistent with d
                - Some member of G is more general than h
            - Remove from S any hypothesis that is more general than another hypothesis in S

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

S_0: $\boxed{\langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing. \varnothing \rangle}$

G_0: $\boxed{\langle ?, ?, ?, ?, ?, ? \rangle}$

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

S_0: $\langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing. \varnothing \rangle$

S_1: $\langle Sunny, Warm, Normal, Strong, Warm, Same \rangle$

G_0: G_1: $\langle ?, ?, ?, ?, ?, ? \rangle$

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

S_0: $\boxed{\langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing. \varnothing \rangle}$

S_1: $\boxed{\langle Sunny, Warm, Normal, Strong, Warm, Same \rangle}$

S_2: $\boxed{\langle Sunny, Warm, ?, Strong, Warm, Same \rangle}$

G_0: G_1: G_2: $\boxed{\langle ?, ?, ?, ?, ?, ? \rangle}$

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

S_0: $\boxed{\langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing. \varnothing \rangle}$

S_1: $\boxed{\langle Sunny, Warm, Normal, Strong, Warm, Same \rangle}$

S_2: S_3: $\boxed{\langle Sunny, Warm, ?, Strong, Warm, Same \rangle}$

G_3: $\boxed{\langle Sunny,?,?,?,?,? \rangle}$ $\boxed{\langle ?,Warm,?,?,?,? \rangle}$ $\boxed{\langle ?,?,Normal,?,?,? \rangle}$ $\boxed{\langle ?, ?,?,?,Cool,? \rangle}$ $\boxed{\langle ?,?,?,?,?,Same \rangle}$

G_0: G_1: G_2: $\boxed{\langle ?, ?, ?, ?, ?, ? \rangle}$

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

S_0: $\langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing. \varnothing \rangle$

S_1: ⟨*Sunny, Warm, Normal, Strong, Warm, Same*⟩

S_2: S_3: ⟨*Sunny, Warm, ?, Strong, Warm, Same*⟩

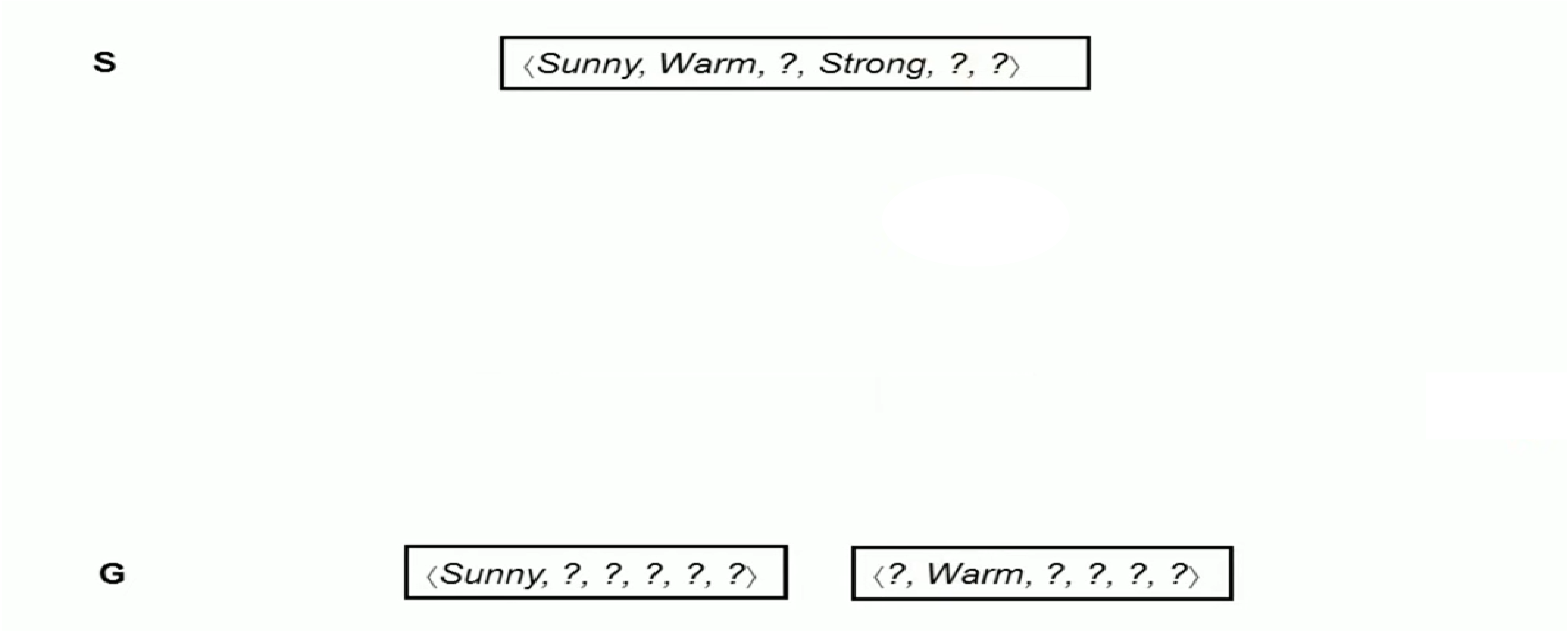
S_4 ⟨*Sunny, Warm, ?, Strong, ?, ?*⟩

G_4: ⟨*Sunny, ?, ?, ?, ?, ?*⟩ ⟨*?, Warm, ?, ?, ?, ?*⟩

G_3: ⟨*Sunny,?,?,?,?,?*⟩ ⟨*?,Warm,?,?,?,?*⟩ ⟨*?,?,Normal,?,?,?*⟩ ⟨*?, ?,?,?,Cool,?*⟩ ⟨*?,?,?,?,?,Same*⟩

G_0: G_1: G_2: ⟨?, ?, ?, ?, ?, ?⟩

Learned Version Space using Candidate Elimination Algorithm



Learned Version Space using Candidate Elimination Algorithm

S

⟨*Sunny, Warm, ?, Strong, ?, ?*⟩

⟨*Sunny, ?, ?, Strong, ?, ?*⟩ ⟨*Sunny, Warm, ?, ?, ?, ?*⟩ ⟨*?, Warm, ?, Strong, ?, ?*⟩

G

⟨*Sunny, ?, ?, ?, ?, ?*⟩ ⟨*?, Warm, ?, ?, ?, ?*⟩