More about Similarity

Teeradaj Racharak (เอ็กซ์)
r.teeradaj@gmail.com

Introduction

Now, we have seen how SVMs make use of a similarity distance (kernels) for classification.

หรือ similarity function

Ones may question if there are other methods for defining such distances !

Looking at the literature, it turns out that many do study about this topic, ranging from the field of Mathematics to Philosophy.

In this lecture, we'd like to discuss some methodologies on similarity distances for natural language processing (NLP) tasks *e.g.*

- Tree kernel
- Cosine similarity
- Semantic similarity

Tree Kernel

Tree Kernel

Exploiting tree structures for classifying programs by functionalities

3 Author(s) Viet Anh Phan ; Ngoc Phuong Chau ; Minh Le Nguyen View All Authors

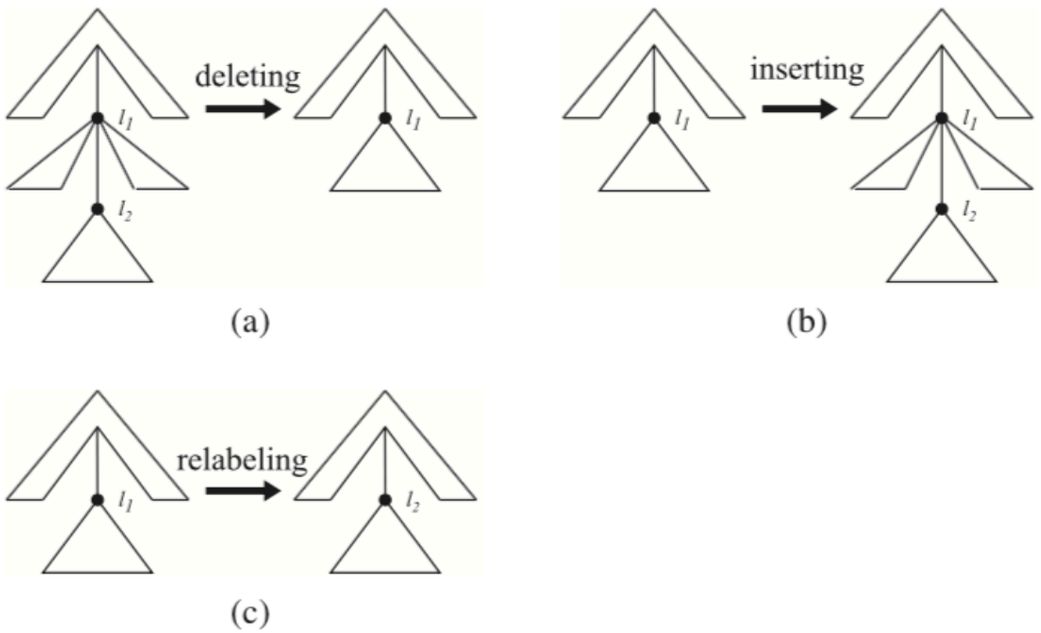


Figure 2 Basic tree edit operations

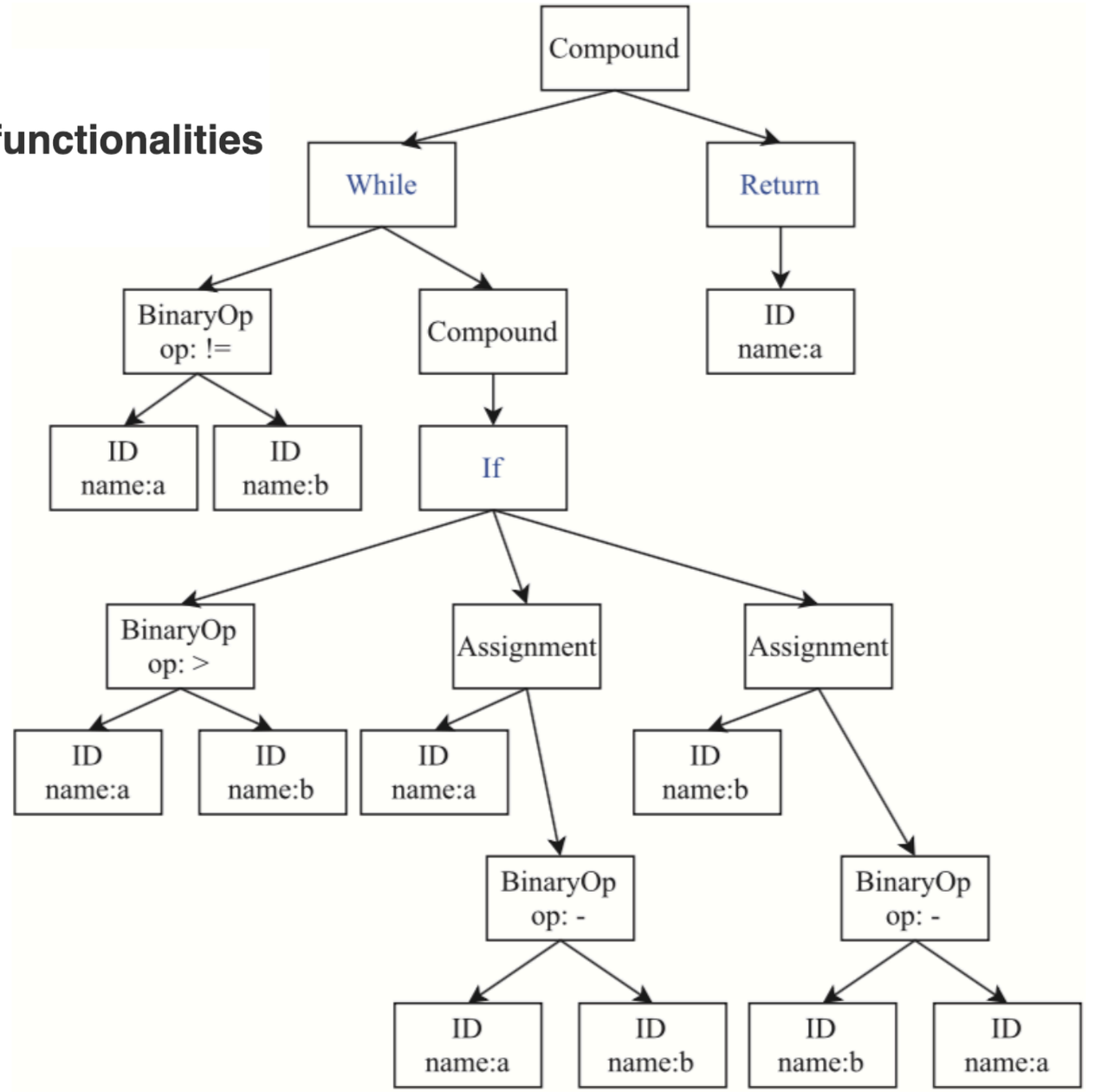
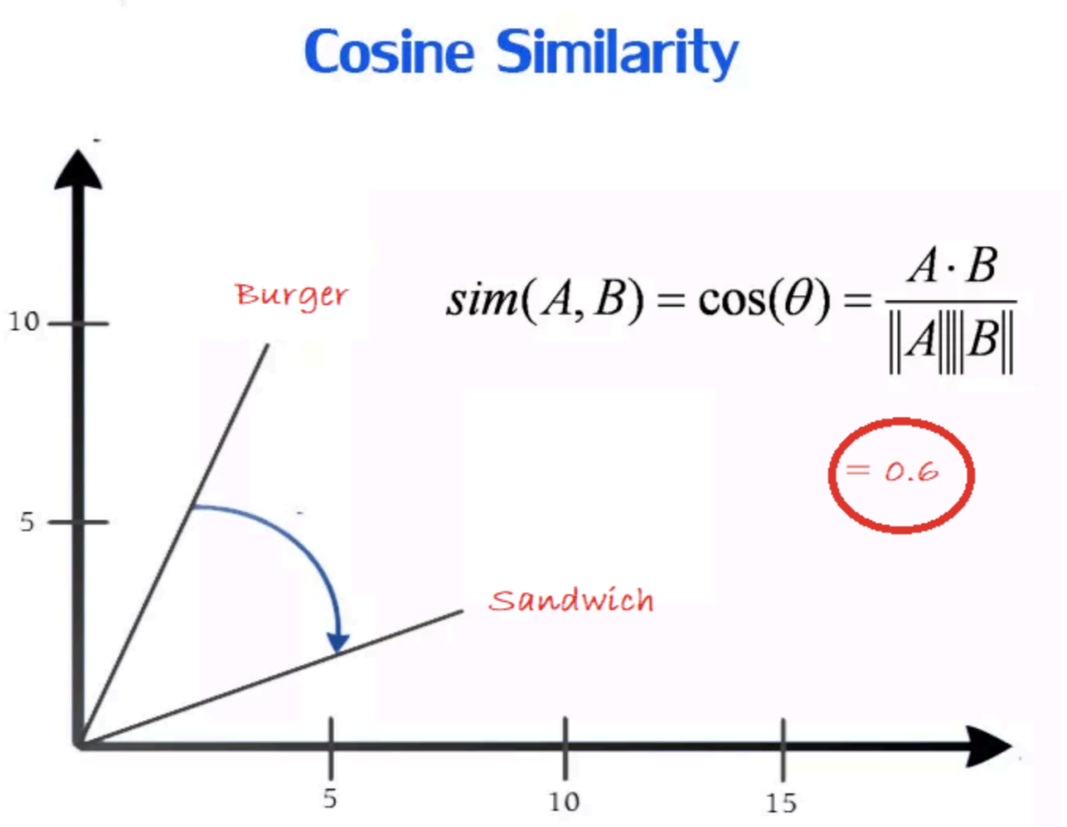


Figure 1 An abstract syntax tree for the code for the Euclidean algorithm

Cosine Similarity

Cosine Similarity

Cosine similarity is a measure of similarity between 2 non-zero vectors of an inner product space that measures the cosine of the angle between them



This technique is extensively used in the fields of information retrieval and data mining *e.g.*

- measure ‘cohesion’ within clusters
- recommendation query

Use Cases of Cosine Similarity

We can use cosine similarity to indicate the degree of similarity between two things *e.g.* as part of a recommendation query

- get movie recommendations based on the preferences of users who have given similar ratings to other movie that you've seen
- find similar documents

Example in Text Similarity

Suppose we want to calculate cosine similarity between two texts:

1. Julie loves me more than Linda loves me
2. Jane likes me more than Julie loves me

Steps of the calculation:

1. Identify all distinct words in both texts
2. Identify the frequency of occurrences of words in both texts and treat them as vectors
3. Apply cosine similarity function

Example in Text Similarity

1. Julie loves me more than Linda loves me
2. Jane likes me more than Julie loves me

Distinct words from both texts	Frequency in Text1	Frequency in Text2
Julie	1	1
loves	2	1
me	2	2
more	1	1
than	1	1
Linda	1	0
Jane	0	1
likes	0	1

Example in Text Similarity

1. Julie loves me more than Linda loves me
2. Jane likes me more than Julie loves me

Distinct words from both texts	Frequency in Text1	Frequency in Text2
Julie	1	1
loves	2	1
me	2	2
more	1	1
than	1	1
Linda	1	0
Jane	0	1
likes	0	1
	a	***b***

Example in Text Similarity

1. Julie loves me more than Linda loves me
2. Jane likes me more than Julie loves me

$$\cos(\boldsymbol{a},\boldsymbol{b}) = \frac{1\times1+2\times1+2\times2+1\times1+1\times1+1\times0+0\times1+0\times1}{\sqrt{1^2+2^2+2^2+1^2+1^2+1^2+0^2+0^2}\,\|\boldsymbol{b}\|}$$

$$= 0.8215838362577491$$

$$\boldsymbol{a} = \begin{bmatrix}1\\2\\2\\1\\1\\1\\0\\0\end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix}1\\1\\2\\1\\1\\0\\1\\1\end{bmatrix}$$

Value ranges between -1 and 1, where
-1 indicates perfectly dissimilar and
1 indicates perfectly similar

Semantic Similarity

Semantic Similarity

The similarity distance between documents or words can be calculated based on their meaning or semantic content.

This means a suitable semantic resource (usually represented by a topological data structure) is required. Fortunately, there are many of them nowadays for many areas *e.g.*

- WordNet is a lexical database of English words
- Gene Ontology contains descriptions about genes and proteins
- SnomedCT contains medical terminologies
- OpenStreetMaps contains map information across the world

⋮

Semantic Similarity in NLP

Semantic similarity is being used in several areas of NLP:

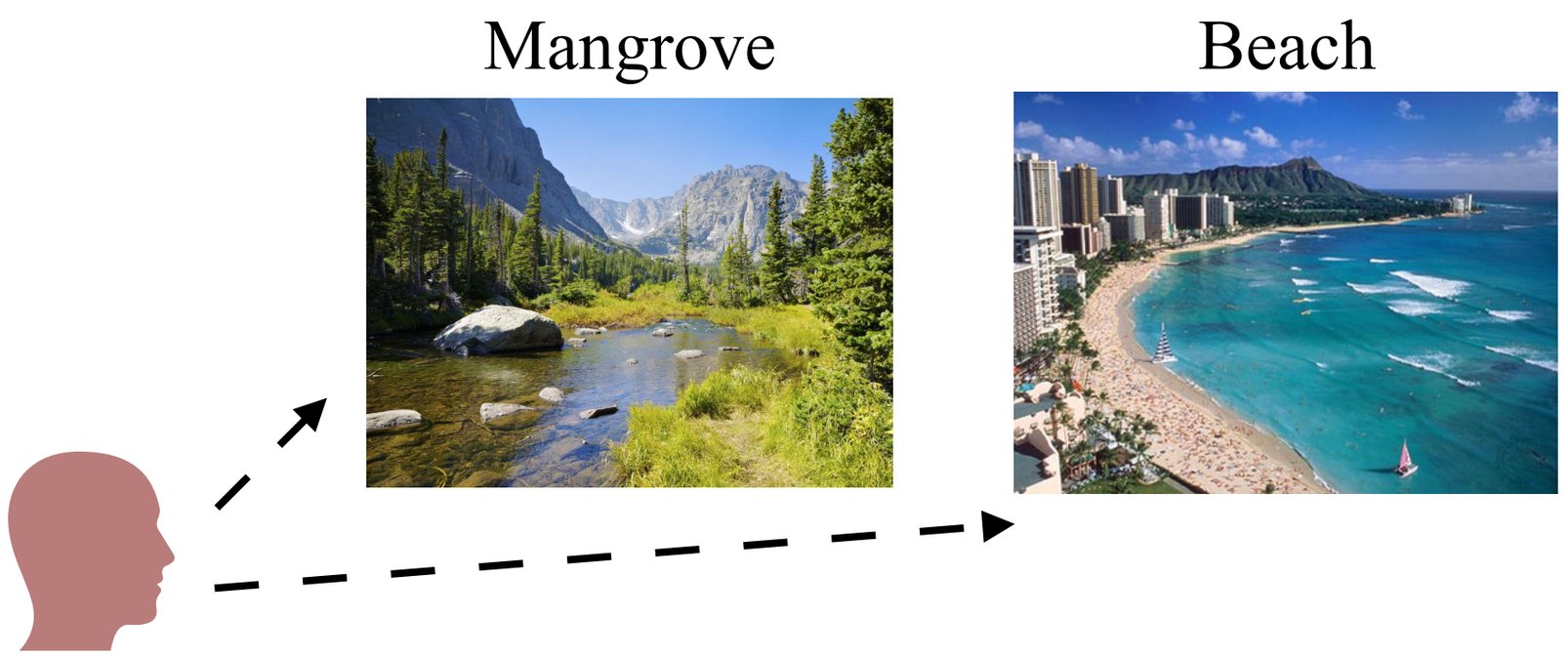
- sentiment analysis (sometimes called ‘opinion mining’)
- natural language understanding
- machine translation (automatically translate texts one human language to another)

Similarity as a Mathematical Operator

Can we define cognitive similarity as a mathematical operator?
(just like we do addition / subtraction / multiplication / *etc.*)

If we can define such operators, then we will understand the behaviors of semantic similarity operators.

But, how human beings think about the notion of similarity?



Cognitive Similarity

Racharak *et al.* (2018) formalized properties of semantic similarity for cognitive use as follows:

Let Π be a set of a user's preferences. Then, a similarity is called:

1. symmetric iff $C \overset{\pi}{\sim} D = D \overset{\pi}{\sim} C$ for any $\pi \in \Pi$
2. equivalent invariant iff C and D are identical implies $\forall \pi \in \Pi : (C \overset{\pi}{\sim} E = D \overset{\pi}{\sim} E)$, and vice versa
3. structurally dependent iff increasing common sub-structure between C and D w.r.t. any $\pi \in \Pi$ implies increasing the degree of similarity between them
4. preference invariant w.r.t. equivalence iff C and D are identical implies $\forall \pi \in \Pi : (C \overset{\pi}{\sim} D = 1)$, and vice versa

---

Teeradaj Racharak, Boontawee Suntisrivaraporn, Satoshi Tojo: Personalizing a Concept Similarity Measure in the Description Logic ELH with Preference Profile. Computing and Informatics 37(3): 581-613 (2018)

Cognitive Similarity

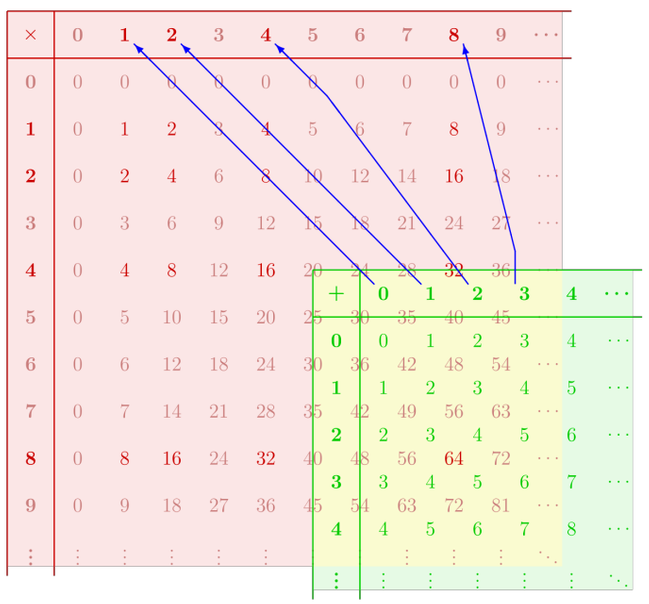
Now, we defined the behaviors of similarity. The next question is that: can we find a concrete operator that satisfies such properties?

If we can find such operators, then we can say

A is similar to *B* (subject to certain preferences)

in a human sense.

One approach is to ‘fit’ the meaning of terms into some structure and employs a structure-preserving mapping technique in algebra (a so-called ‘homomorphism’) for similarity calculation.



(Image from https://en.wikipedia.org/wiki/Homomorphism)

Similarity between Trees

$$\mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C) = \mu^{\pi}(\mathcal{P}_D, \mathcal{E}_D) \cdot \mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_D, \mathcal{P}_C) + (1 - \mu^{\pi}(\mathcal{P}_D, \mathcal{E}_D)) \cdot \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_D, \mathcal{E}_C)$$

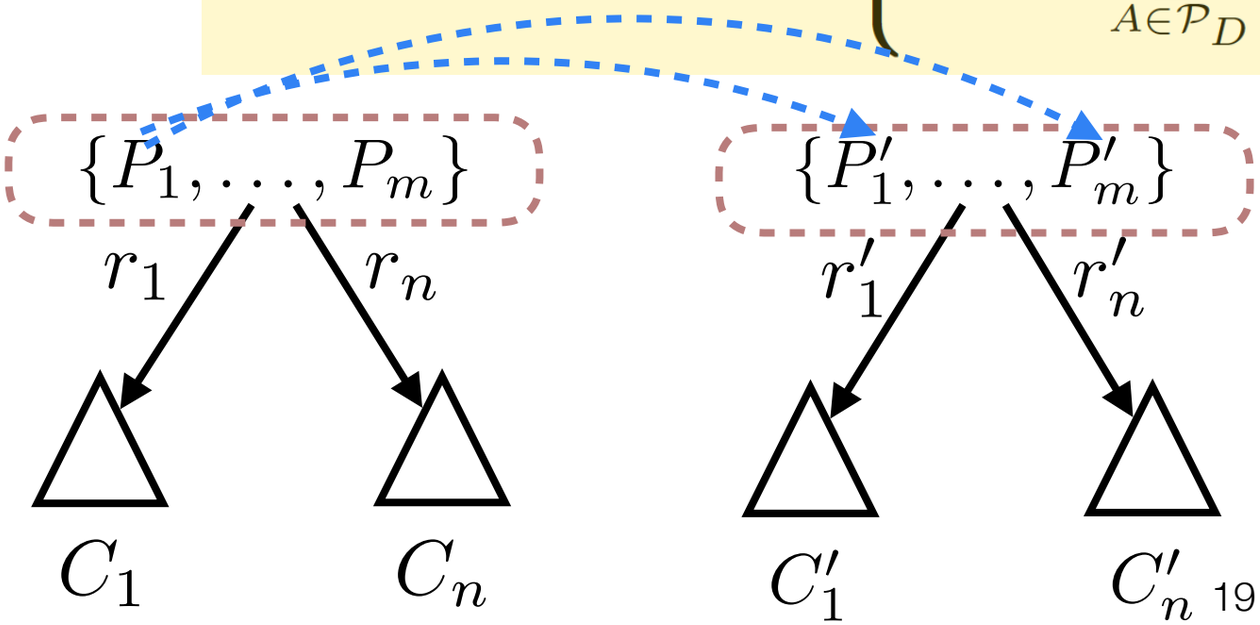
$$\text{where } \mu^{\pi}(\mathcal{P}_D, \mathcal{E}_D) = \begin{cases} 1 & \text{if } \sum_{A \in \mathcal{P}_D} \hat{\mathrm{i}}(A) + \sum_{\exists r. X \in \mathcal{E}_D} \hat{\mathrm{i}}(r) = 0 \\ \dfrac{\sum_{A \in \mathcal{P}_D} \hat{\mathrm{i}}(A)}{\sum_{A \in \mathcal{P}_D} \hat{\mathrm{i}}(A) + \sum_{\exists r. X \in \mathcal{E}_D} \hat{\mathrm{i}}(r)} & \text{otherwise;} \end{cases}$$

Next, we consider how to compute a subsumption degree between 2 nodes

Similarity between Trees

$$\mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C) = \mu^{\pi}(\mathcal{P}_D, \mathcal{E}_D) \cdot \mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_D, \mathcal{P}_C) + (1 - \mu^{\pi}(\mathcal{P}_D, \mathcal{E}_D)) \cdot \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_D, \mathcal{E}_C)$$

$$\mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_D, \mathcal{P}_C) = \begin{cases} 1 & \text{if } \sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) = 0 \\ 0 & \text{if } \sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) \neq 0 \\ & \text{and } \sum_{B \in \mathcal{P}_C} \hat{\mathfrak{i}}(B) = 0 \\ \dfrac{\sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) \cdot \max_{B \in \mathcal{P}_C} \{\hat{\mathfrak{s}}(A,B)\}}{\sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A)} & \text{otherwise;} \end{cases}$$



Idea.

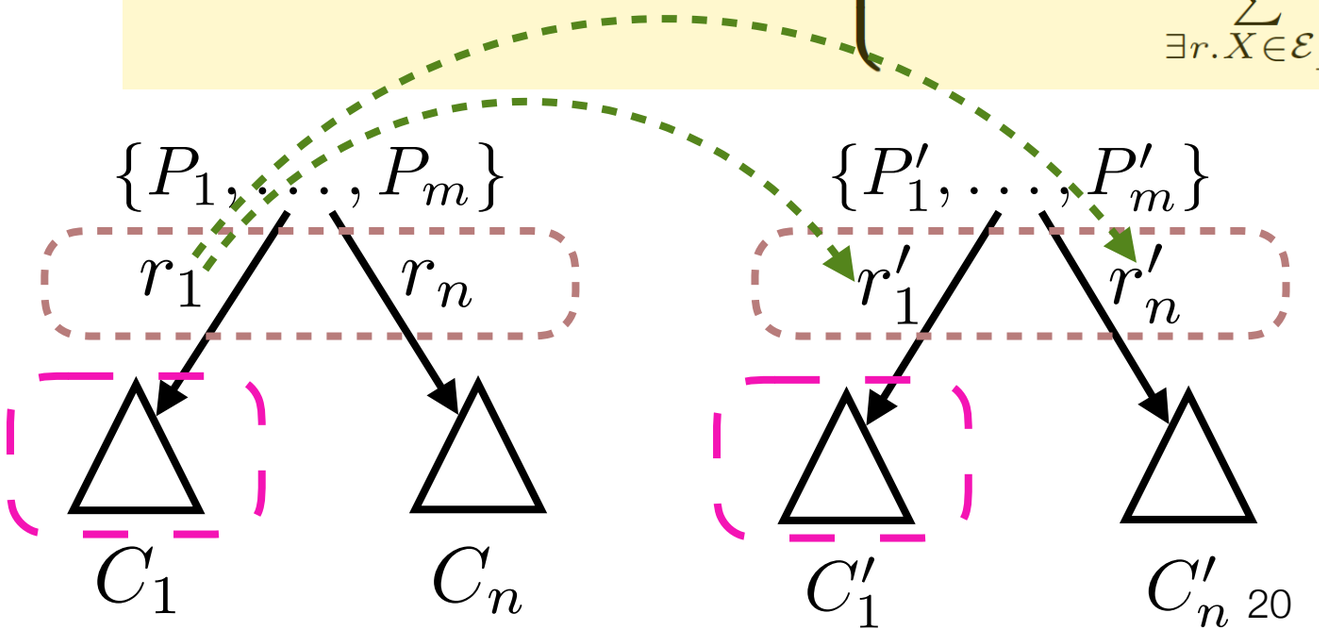
Consider each importance of each best matching partner

Next, we consider how to compute a subsumption degree between 2 edges

Similarity between Trees

$$\mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C) = \mu^{\pi}(\mathcal{P}_D, \mathcal{E}_D) \cdot \mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_D, \mathcal{P}_C) + (1 - \mu^{\pi}(\mathcal{P}_D, \mathcal{E}_D)) \cdot \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_D, \mathcal{E}_C)$$

$$\mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_D, \mathcal{E}_C) = \begin{cases} 1 & \text{if } \sum_{\exists r. X \in \mathcal{E}_D} \hat{\mathbf{i}}(r) = 0 \\ 0 & \text{if } \sum_{\exists r. X \in \mathcal{E}_D} \hat{\mathbf{i}}(r) \neq 0 \\ & \text{and} \\ & \sum_{\exists s. Y \in \mathcal{E}_C} \hat{\mathbf{i}}(s) = 0 \\ \dfrac{\sum_{\exists r. X \in \mathcal{E}_D} \hat{\mathbf{i}}(r) \cdot \max_{\epsilon_j \in \mathcal{E}_C} \{\mathsf{e\text{-}hd}^{\pi}(\exists r. X, \epsilon_j)\}}{\sum_{\exists r. X \in \mathcal{E}_D} \hat{\mathbf{i}}(r)} & \text{otherwise;} \end{cases}$$



Idea.

Consider each importance of each best matching partner

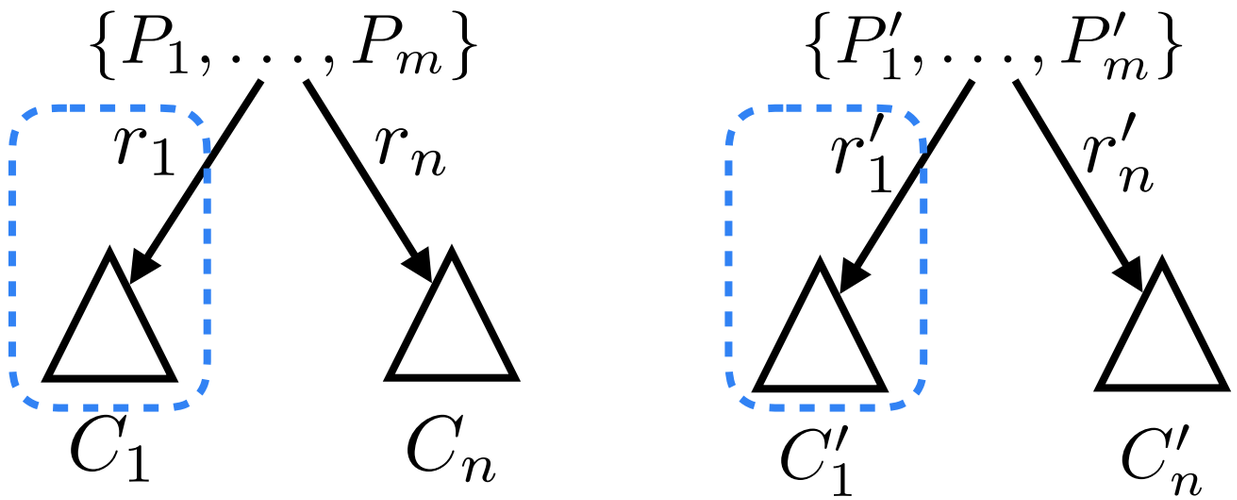
Next, we consider a subsumption degree between its subtrees.

Similarity between Trees

$$\mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C) = \mu^{\pi}(\mathcal{P}_D, \mathcal{E}_D) \cdot \mathsf{p\text{-}hd}^{\pi}(\mathcal{P}_D, \mathcal{P}_C) + (1 - \mu^{\pi}(\mathcal{P}_D, \mathcal{E}_D)) \cdot \mathsf{e\text{-}set\text{-}hd}^{\pi}(\mathcal{E}_D, \mathcal{E}_C)$$

$$\mathsf{e\text{-}hd}^{\pi}(\exists r.X, \exists s.Y) = \gamma^{\pi}(r, s) \cdot (\hat{\mathfrak{d}}(r) + (1 - \hat{\mathfrak{d}}(r)) \cdot \mathsf{hd}^{\pi}(\mathcal{T}_X, \mathcal{T}_Y))$$

$$\gamma^{\pi}(r, s) = \begin{cases} 1 & \text{if } \sum_{r' \in \mathcal{R}_r} \hat{\mathfrak{i}}(r') = 0 \\ \dfrac{\sum_{r' \in \mathcal{R}_r} \hat{\mathfrak{i}}(r') \cdot \max_{s' \in \mathcal{R}_s} \{\hat{\mathfrak{s}}(r', s')\}}{\sum_{r' \in \mathcal{R}_r} \hat{\mathfrak{i}}(r')} & \text{otherwise.} \end{cases}$$



Convention.
$\mathcal{T}_X$ denotes a concept tree of X.

Idea.
The degree of subsumption under preference profile between its subtrees can be recursively called hd^{π}.

Similarity between Trees

Once we calculate the degree (subjective to a user's preferences) of homomorphism mapping between two tree structures, then the degree of similarity between these two trees can be also calculated as follows:

$$\frac{\mathsf{hd}^{\pi}(C,D)+\mathsf{hd}^{\pi}(D,C)}{2}$$

where C and D are two different terms / words / texts from a corpus.

In Racharak *et al.* (2018), the mathematical proofs of its properties and computational complexity were clearly shown *i.e.* the computation:

- satisfies desirable properties, and
- can be computed in polynomial time (worst-case complexity).

Teeradaj Racharak, Boontawee Suntisrivaraporn, Satoshi Tojo: Personalizing a Concept Similarity Measure in the Description Logic ELH with Preference Profile. Computing and Informatics 37(3): 581-613 (2018)