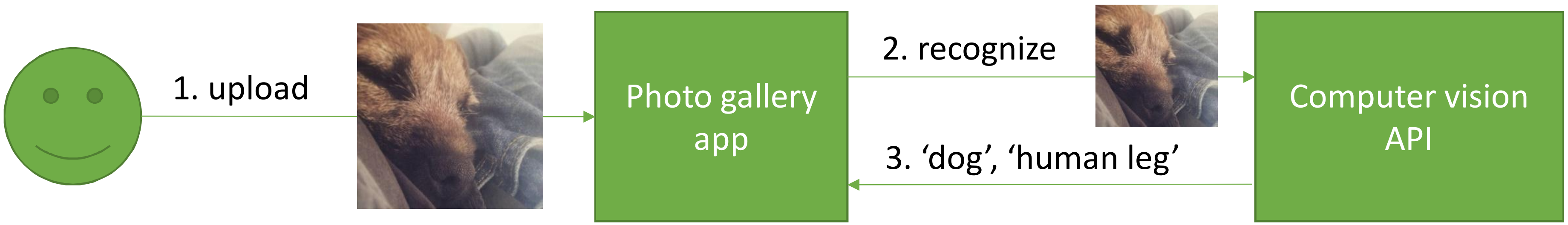
Merging Intelligent API Responses using a Proportional Representation Approach

Tomohiro Ohtake[1], Alex Cummaudo[1], Mohamed Abdelrazek[1],
Rajesh Vasa[1], and John Grundy[2]
1: Deakin University, Australia, 2: Monash University, Australia

2019-06-13 at ICWE 2019

A use case of intelligent APIs

- A developer wants to add an auto-tagging feature to his photo gallery application
- He starts to use intelligent web APIs instead of building intelligent engine from scratch
- He expected that APIs are reliable and deterministic



Issues on intelligent APIs

- Low reliability of results
- Results differs between similar endpoints

Input

Amazon

Label	Score
Animal	0.764
Cat	**0.764**
Mammal	0.764
Pet	0.764
Siamese	**0.764**

Google

Label	Score
dog breed group	**0.876**
nose	0.870
snout	0.836
dog like mammal	**0.753**
whiskers	0.745
fur	0.743
dog breed	**0.728**
puppy	**0.595**
paw	0.550

Microsoft

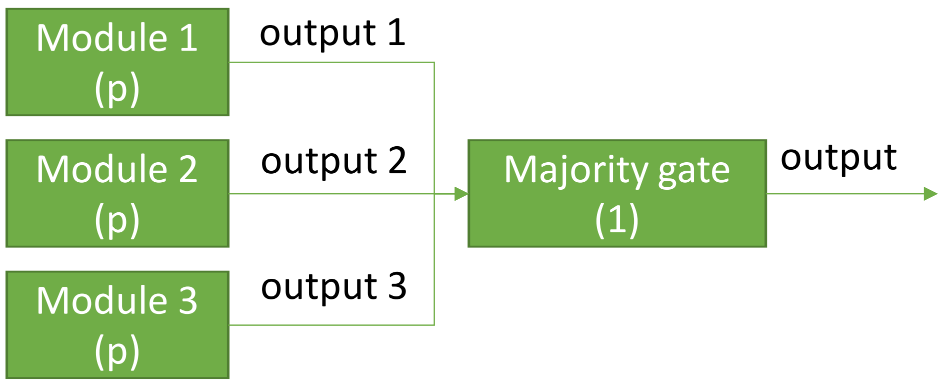
Label	Score
dog	**0.969**
indoor	0.954
animal	0.885
mammal	0.719
close	0.317

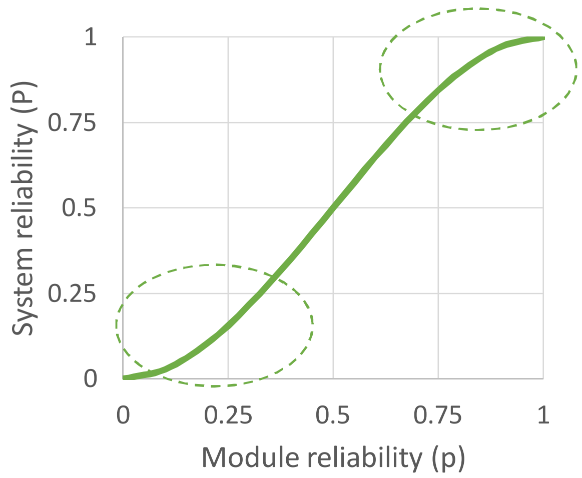
Human-verified

Label	Score
Animal	1
Carnivore	1
Close-up	1
Dog	**1**
Human leg	1
Mammal	1
Pet	1
Human arm	0
Human ear	0
Human eye	0
Human hand	0
Human head	0
Human mouth	0
Pig	0
Rabbit	0
Suidae	0
Wombat	0

Previous attempt for reliabilities

- Triple Modular Redundancy
  - It emits the majority module output as a system output
  - Reliability of the system: $P = p^3 + 3p^2(1-p) = 3p^2 - 2p^3$
  - Useful when improving reliabilities using highly-reliable modules



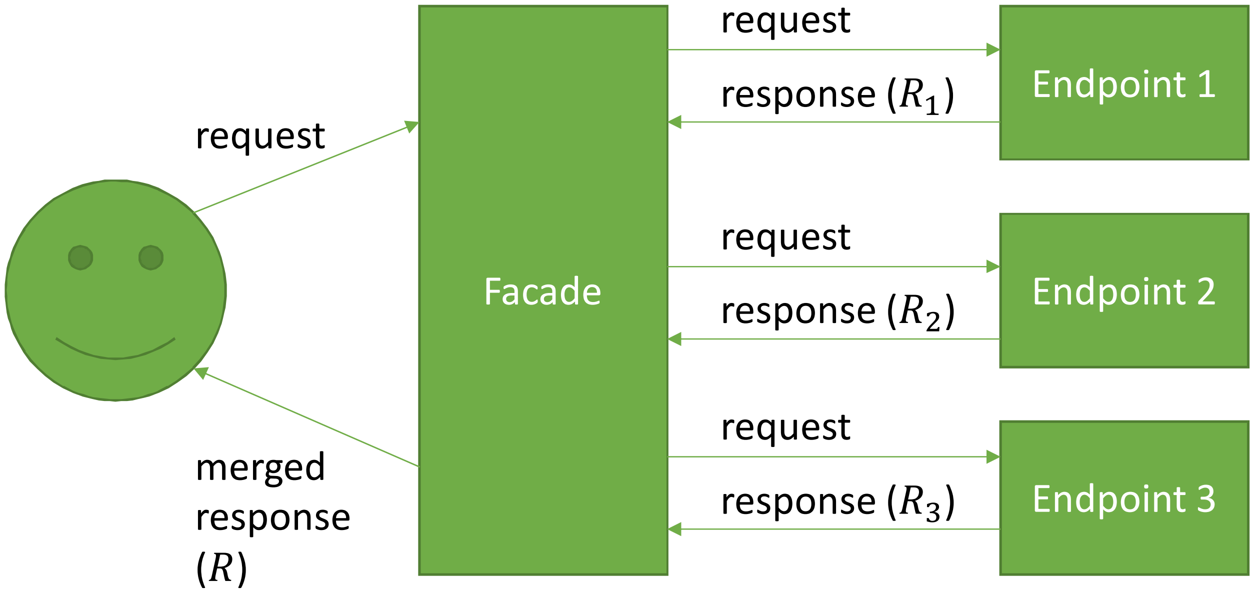


Research questions

1. Is it possible to improve reliability by merging multiple intelligent API results?
2. Are there better algorithms for merging these results than currently in use?

API Facade

- $R = \{< l_1, s_1 >, < l_2, s_2 >, \dots\}$
- merge: $R^n \to R$



Four properties of merging operators

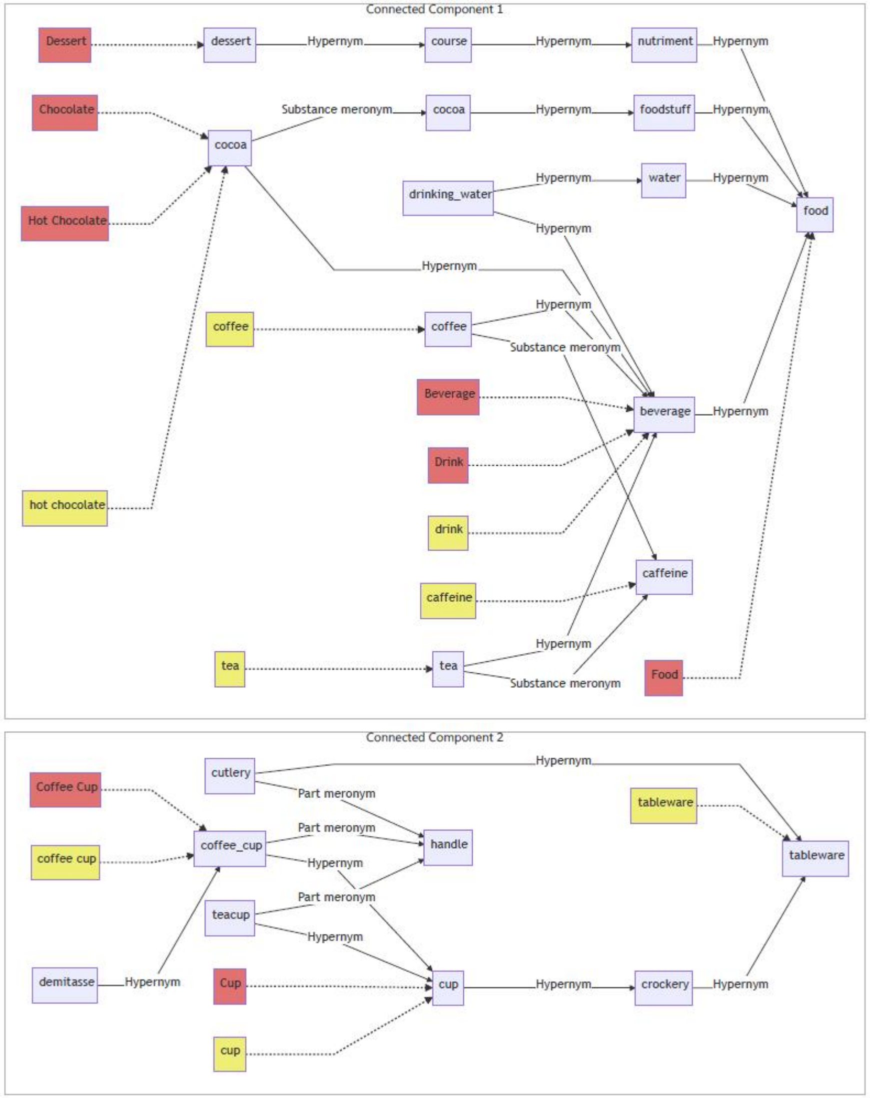
1. Identity
   - $R = \mathrm{merge}(R)$
2. Commutativity
   - $\mathrm{merge}(R_1, R_2) = \mathrm{merge}(R_2, R_1)$
3. Reflexivity
   - $R = \mathrm{merge}(R, R)$
4. Additivity
   - Let $R = \mathrm{merge}(R_1, R_2)$, $R' = \mathrm{merge}(R_1{}', R_2)$ be merged responses.
   - R_1 and $R_1{}'$ are same, except $R_1{}'$has a higher score for label l_x than R_1.
   - Then, R' score for l_x should be greater than or equal to R score for l_x.

Steps of merging

1. Groups labels into connected components (CCs)
2. Decides total number of labels
3. Allocates number of labels to CCs
4. Selecting labels from CCs

S1. Grouping into CCs

- Groups labels into connected components of WordNet synsets
- Nodes
  - Red: Labels from endpoint 1
  - Yellow: Labels from endpoint 2
  - Purple: Meaning (WordNet synset)



S2. Total number of labels

- $\min_i(|R_i|) \leq \frac{\sum_i |R_i|}{n} \leq \max_i(|R_i|) \leq \sum_i |R_i|$
- The proposal uses $\left\lfloor \frac{\sum_i |R_i|}{n} \right\rfloor$ to conform the four properties

S3. Similarity to proportional representation

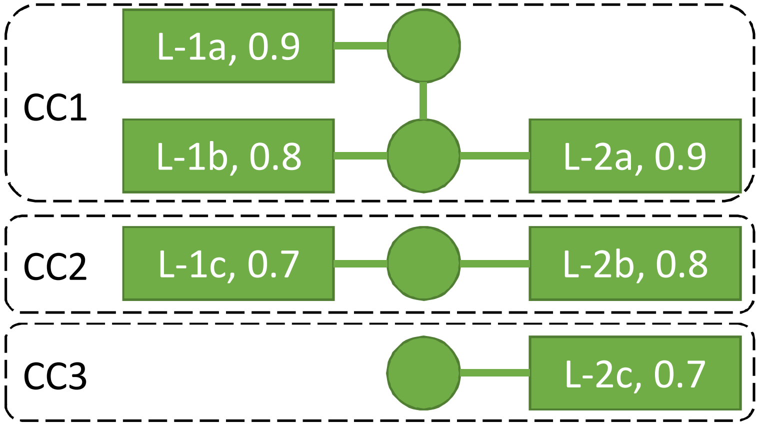
- Allocating number of labels is similar to proportional representation

Proportional representation	Allocation to CCs
Party	CC
Number of votes to a party	Number of labels in a CC
Number of seats	Number of labels to emit

- Differences and assumption
  - A CC which is supported by more endpoints should be more reliable
  - In context of voting, a party which is supported by wide-range of people should have more seats

S3. Allocation to CCs

Allocating 3 labels to 3 CCs



- A CC with the highest product of highest scores receives one allocation
  - Remove highest scores from the allocated CC
- If all CCs have an empty array, remove them

CC	Score	Max	Prod	Alloc
1	[0.9, 0.8], [0.9]	[0.9, 0.9]	0.81	0+1
2	[0.7], [0.8]	[0.7, 0.8]	0.56	0
3	[], [0.7]	[NA, 0.7]	NA	0

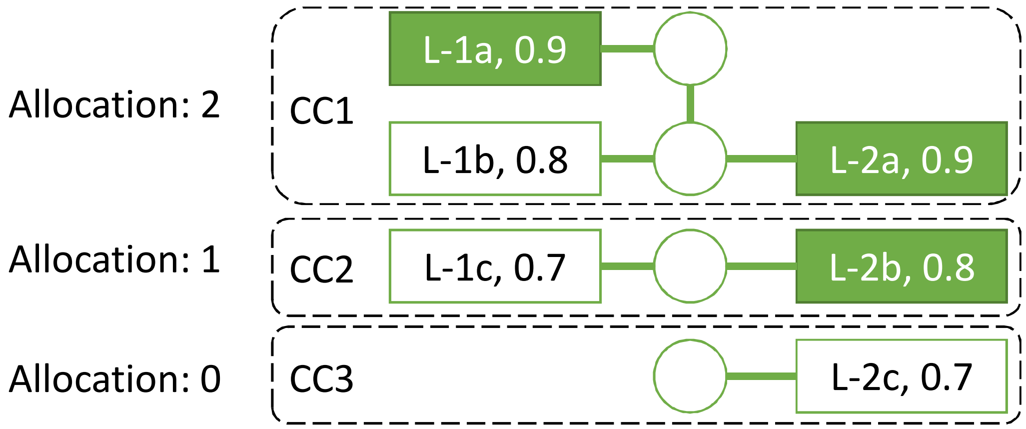
CC	Score	Max	Prod	Alloc
1	[0.8], []	[0.8, NA]	NA	1
2	[0.7], [0.8]	[0.7, 0.8]	0.56	0+1
3	[], [0.7]	[NA, 0.7]	NA	0

CC	Score	Max	Prod	Alloc
1	[0.8], []			1
2	[], []			1
3	[], [0.7]			0

CC	Score	Max	Prod	Alloc
1	[0.8]	[0.8]	0.8	1+1
2	[]	[]	NA	1
3	[0.7]	[0.7]	0.7	0

S4. Selecting labels

- Selects labels with n-highest scores up to number of allocation

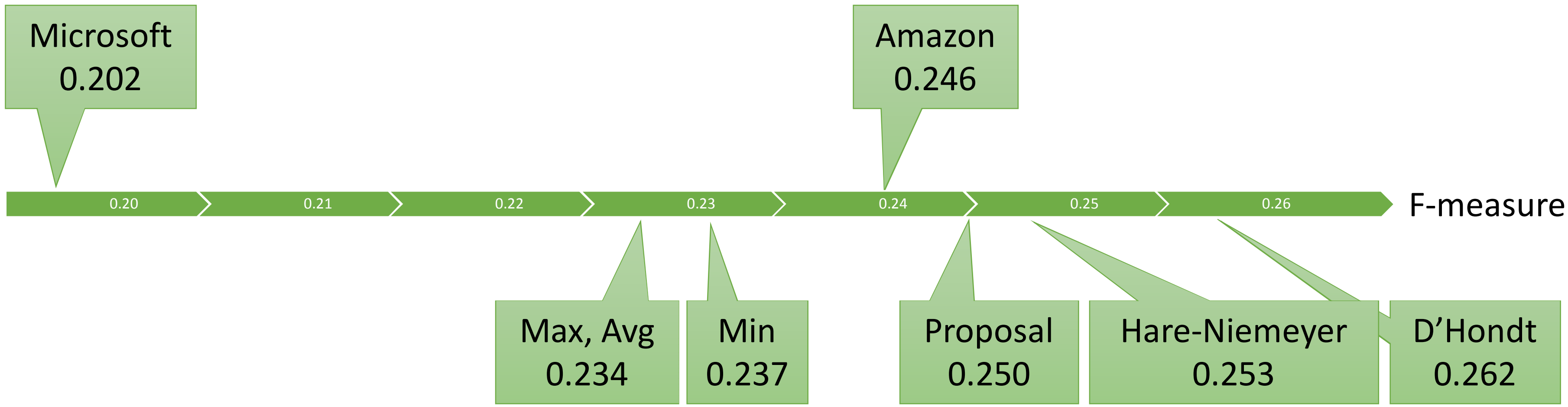


Evaluation

- Input
  - 1000 images from Open Images dataset V4
- API endpoints
  - Amazon
  - Google
  - Microsoft
- Merge operators
  - Naive
    - Min
    - Max
    - Average
  - Traditional proportional representation
    - D’Hondt
    - Hare-Niemeyer
  - Proposed

Evaluation result 1

- Merging Amazon results and Microsoft results
- All three PP-based methods performs better than Amazon
  - RQ1 is true



Evaluation result 2

- Average of 4 combinations: {A, G}, {G, M}, {M, A}, {A, G, M}
- The proposal performs the best in F-measure
  - RQ2 is true

	Precision	Recall	F-measure
Min	0.780	0.151	**0.252**
Max	0.266	0.500	**0.344**
Average	0.266	0.500	**0.344**
D'Hondt	0.361	0.335	**0.346**
Hare-Niemeyer	0.361	0.336	**0.347**
Proposal	**0.358**	**0.362**	**0.360**

Correction on the paper:
Precision and recall values in Table 8 are wrong.
F-measure values in Table 8 and all values in Table 7 are correct.

Conclusion and future works

- Conclusion
  - The proposed method merges API responses better than naive operators and other proportional representation methods
  - The proposed method can be applied to other intelligent APIs
    - If response type is a list of entity and score, and if there is a way to group entities
- Future works
  - Use graph structure to improve reliability
    - Selecting synsets instead of labels
    - Propagating scores to synsets