Pattern Recognition Lab

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Non-deterministic Behavior of Ranking-based Metrics when Evaluating Embeddings

Can we hack mAP?

Anguelos Nicolaou, Sounak Dey, Vincent Christlein, Andreas Maier, and Dimosthenis Karatzas
Pattern Recognition Lab, FAU / Computer Vision Center, UAB
August 20, 2018

Performance Evaluation

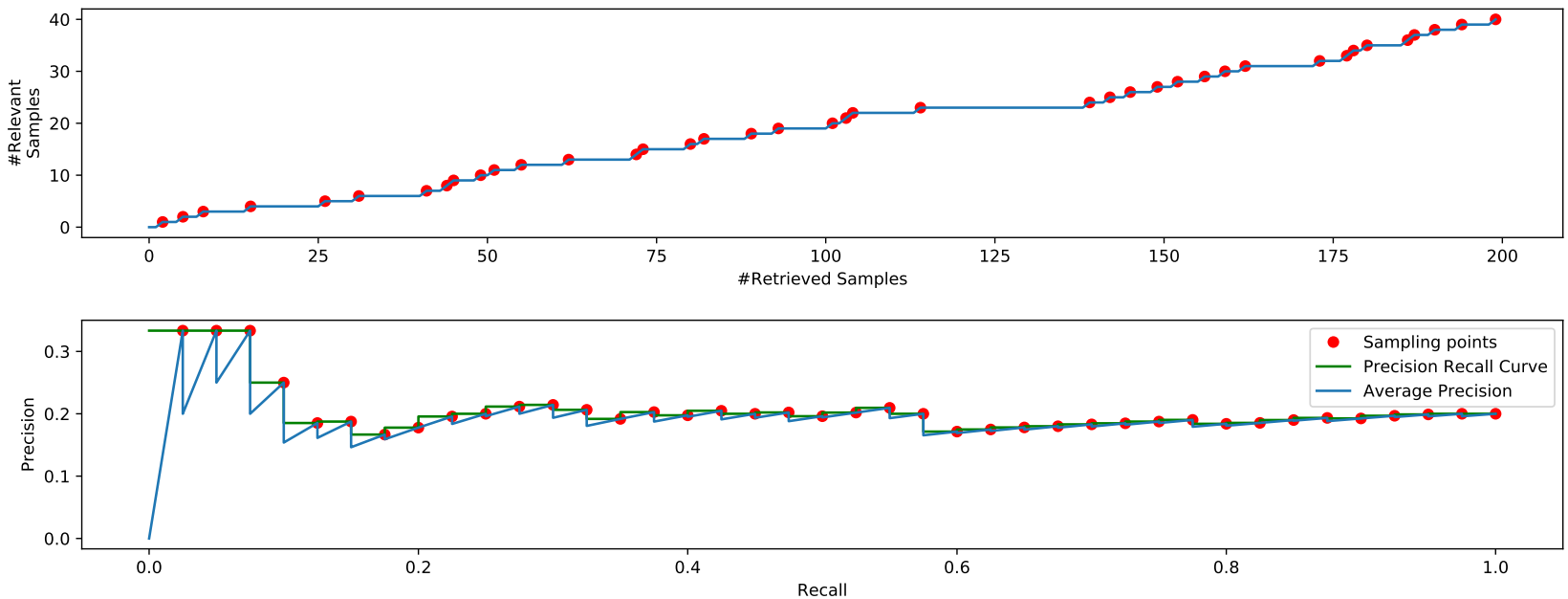
- Steers research in the long run.
  - Fitness function of an evolutionary process.
- Sometimes opaque.
- Our primary mean of analyzing systems scientifically.
- Someone has to do it!
  - Better not the guy who cares about the outcome the most.

Metrics

- Metric spaces:
  - Non negativity
  - Identity
  - Symmetry
  - Triangle Inequality
- Metric between two annotations of the same dataset (one being the ground-truth).
- Desired Properties in Performance Metrics:
  - Metric space of performance of systems.
  - Simple, intuitive.
  - Proportional to perceived differences (unsaturated).
  - Universal.

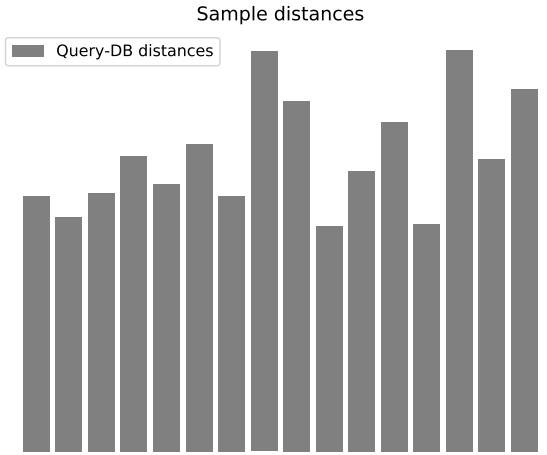
mean Average Precision

- Evaluates rankings of relevant and irrelevant samples.
- Approximation of the area of the Precision-Recall curve.



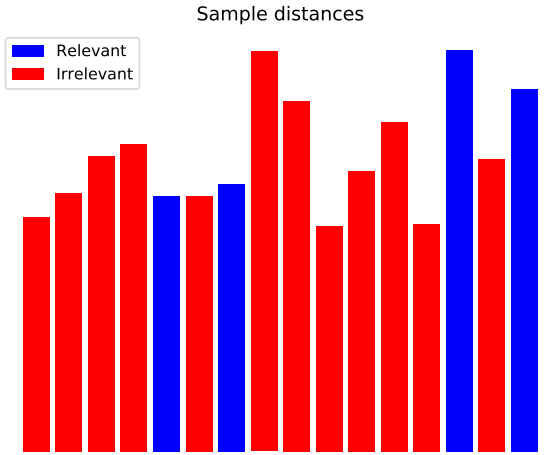
mAP Algorithm

- Get query-database distances.



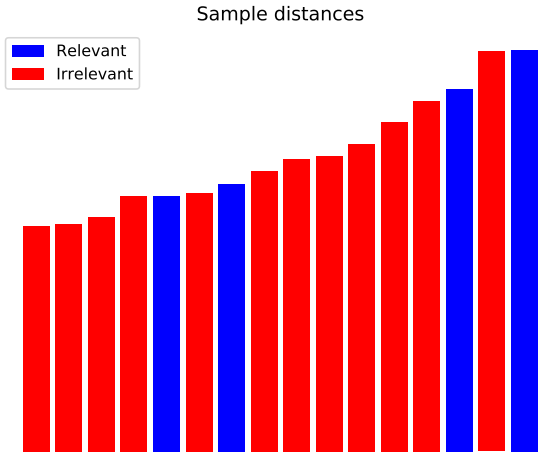
mAP Algorithm

- Get query-database distances.
- Separate the the database in relevant/non-relevant



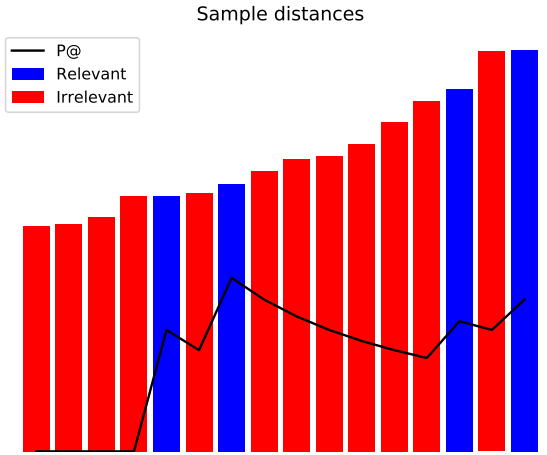
mAP Algorithm

- Get query-database distances.
- Separate the the database in relevant/non-relevant
- Sort by distance



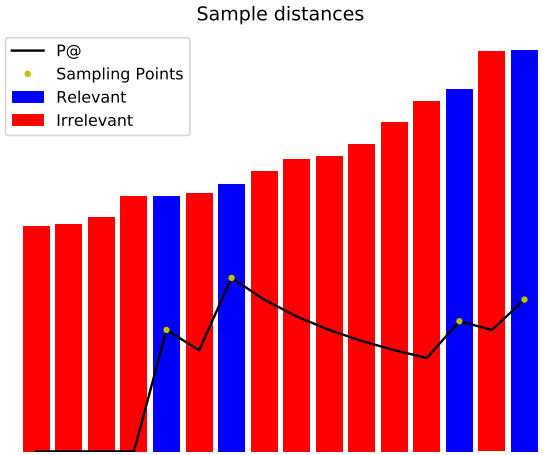
mAP Algorithm

- Get query-database distances.
- Separate the the database in relevant/non-relevant
- Sort by distance
- Estimate Average Precision



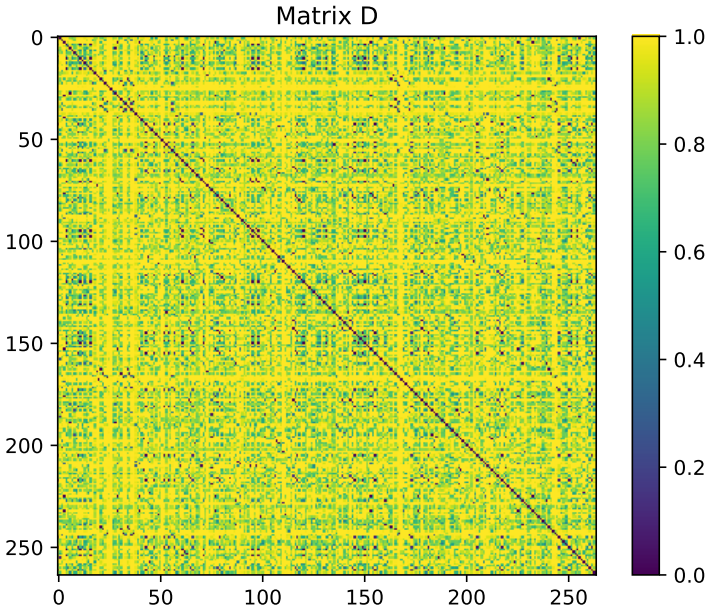
mAP Algorithm

- Get query-database distances.
- Separate the the database in relevant/non-relevant
- Sort by distance
- Estimate Average Precision
- Sample Average Precision



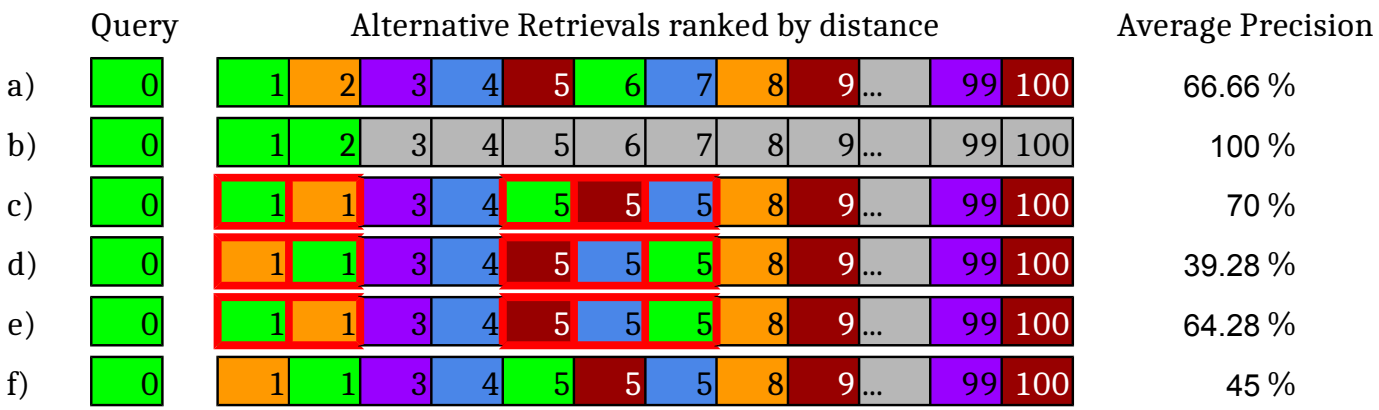
mAP Algorithm

- Get query-database distances.
- Separate the the database in relevant/non-relevant
- Sort by distance
- Estimate Average Precision
- Sample Average Precision
- Leave-one-out: Query in DB
- Efficiency: Computation in Distance Matrix



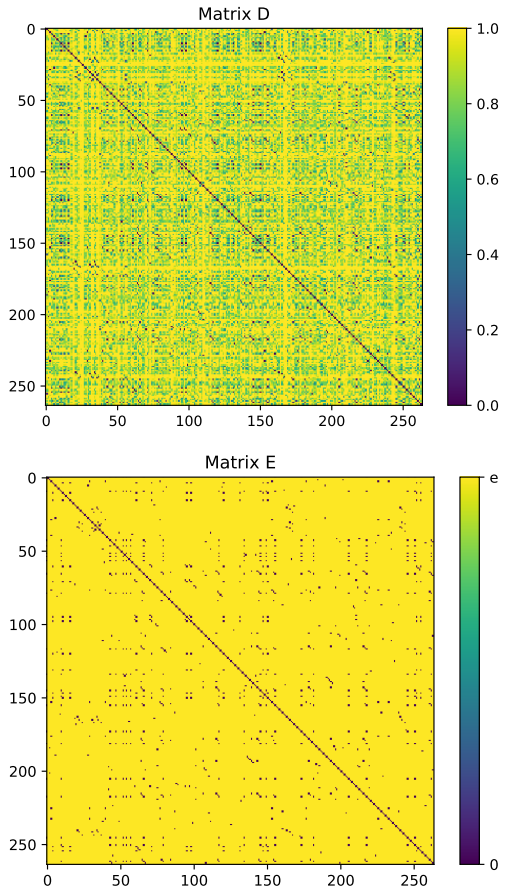
Ranking Ambiguity

- When evaluating a retrieval system: no-problem.
- When sorting is part of the performance evaluation, sorting must be unambiguous.
  - Sometimes it is not.
- Example retrieval in 100 samples with 2 relevant:



mAP bounds:

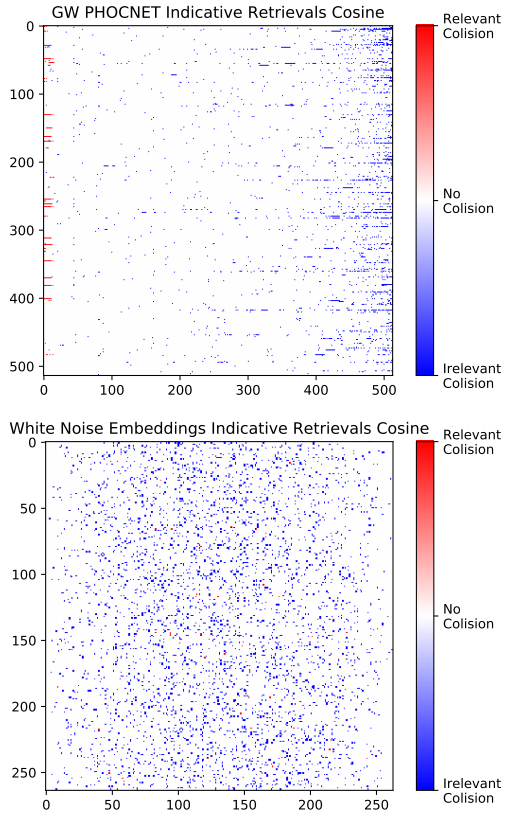
- Unpredictability:
  - Not random!
  - Follows no statistic!
  - Tends to be systemic.
  - Cannot be (easily) detected.
- Bounds are deterministic, predictable, and fast:
- $mAP^{-}(D,R) = mAP(D + e * R, R)$
- $mAP^{+}(D,R) = mAP(D - e * R, R)$



Any empirical evidence? Does it ever happen in the real world?

- They are rare.
- We only found one PHOCNet on GW!
  - State-of-the-art word spotter.
  - 899 samples, 166 classes.
  - A single collision with an measurable impact on mAP $\sim 0.05\%$
- What about edit-distances? HOC embeddings?

Amplified Visualization:



What is the worst case scenario? Who cares about .05%?

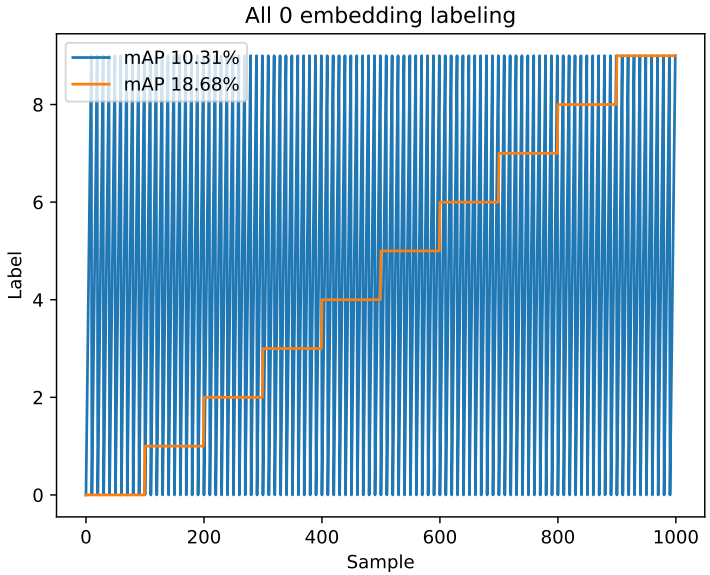
- Example 1000 samples of 10 classes:
- Zero distance matrix: $mAP^{+} = 100\%$
- Zero distance matrix: $mAP^{-} = 5.18\%$
- Zero distance matrix + e * white-noise = Random distance matrix
- Random distance matrix: $E[mAP] = 10.4\%$
- Random distance matrix: $\sigma_{mAP} = 0.053\%$

Deceptive/Adversarial Solutions

- The set-up:
  - Can we control the order in whitch samples are evaluated?
  - Or are they simply ordered by class?
  - Self-classification of 1000 samples with 10 classes

Deceptive/Adversarial Solutions

- The set-up:
  - Can we control the order in whitch samples are evaluated?
  - Or are they simply ordered by class?
  - Self-classification of 1000 samples with 10 classes
- The all-zero cheat:
  - From 10.4% to 18.68%
  - $> 155 * \sigma_{mAP}$

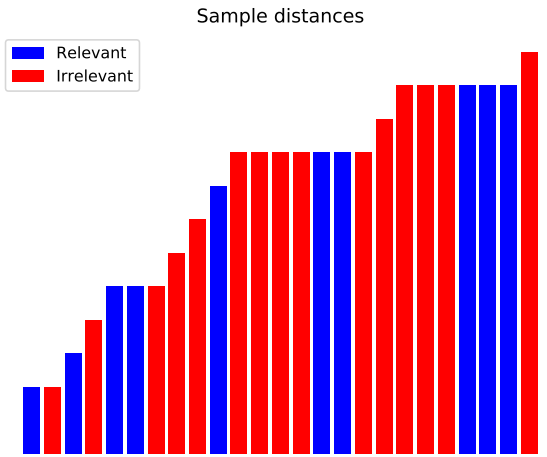


mAP expectation

- The true AP: $E[AP]$ over all permutations of equidistant samples.
- Dynamic programming
- $O(n) -> O(n^2)$?
  $O(r+l) -> O(r+i)$
- Algorithm:
  - Map every permutation to a path from top-left to bottom-right
    Relevant: move down
    Irrelevant: move right
  - Compute the probability of every cell $P_{cell}(n,k)$
  - Compute the probability a cell is used $P_{parent}(n,k)$
  - Compute $P@(n,k)$
  - $AP = \sum_{n=1}^{|R|} \sum_{k=1}^{|I|} P_{cell}(n,k) P_{parent(n,k)}$

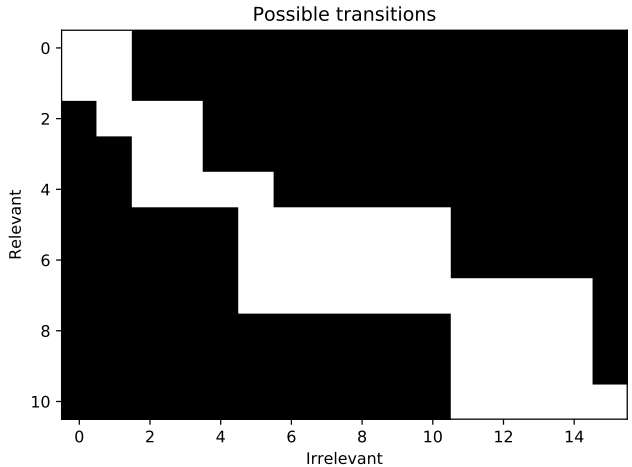
Dynamic AP

- Input: Ambiguous retrievals



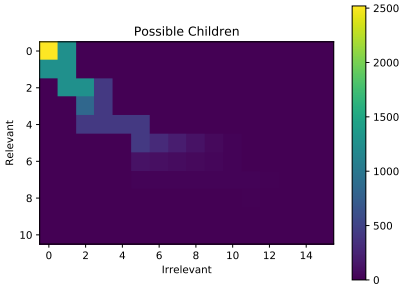
Dynamic AP

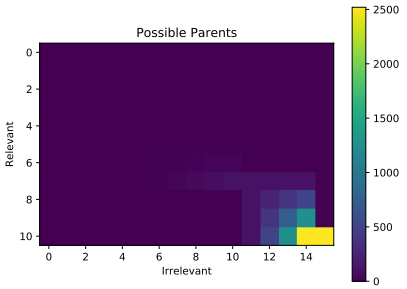
- Input: Ambiguous retrievals
- Compute possible paths

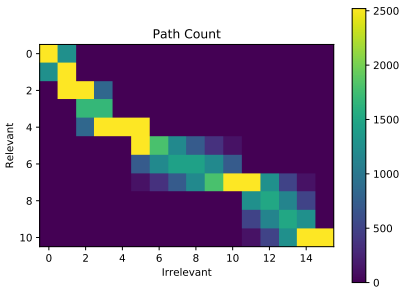


Dynamic AP

- Input: Ambiguous retrievals
- Compute possible paths
- Compute cell probability

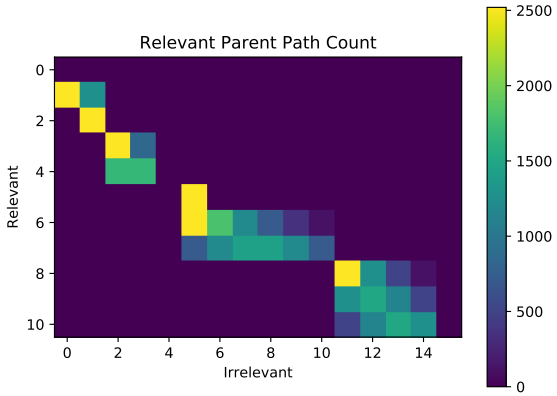


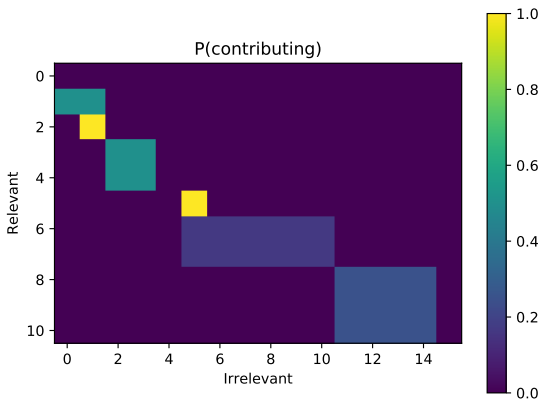




Dynamic AP

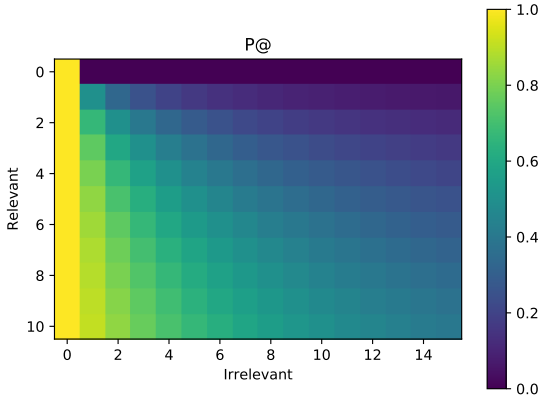
- Input: Ambiguous retrievals
- Compute possible paths
- Compute cell probability
- Compute contribution probability...





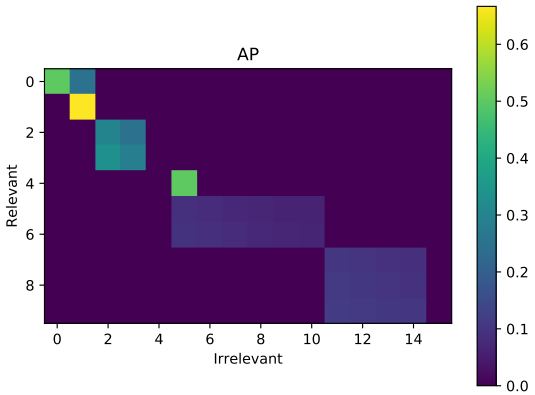
Dynamic AP

- Input: Ambiguous retrievals
- Compute possible paths
- Compute cell probability
- Compute contribution probability...
- Compute *P@*



Dynamic AP

- Input: Ambiguous retrievals
- Compute possible paths
- Compute cell probability
- Compute contribution probability...
- Compute *P@*
- Compute *AP*



Conclusions

- Performance Metrics should be held to a higher standard than methods.
- Equidistant samples can have a measurable impact in real world scenarios.
- They hard to detect!
- They could be exploited with adversarial solutions.
- They are easy to combat: mAP^{-}!
- True mAP of ambiguous sorting complicated.
- Don't use mAP on weak systems / hard benchmarks.
- How about architecture search? could AI learn to cheat with mAP?

Implementation available: