Max Algorithms in Crowdsourcing Environments

Marco Amoruso   Daniele Anello

University of Salerno, department of computer science

June 13, 2015

Outline

Authors

Petros Venetis, Hector Garcia-Molina, Kerui Huang,
Neoklis Polyzotis.

Tag Cloud

pages
IMPACT
considered
Computing
close
bubble
WORK
tournament
algorithms
USED
APPROPRIATE
present
size
model
ask
RETRIEVED
rule
based
set
OPTIMAL
CORRECT
task
crowdsourcing
error
value
PARAMETERS
constraints
MAXIMIZE
ITEM
LINEAR
OPERATES
RETURNED
PROPOSE
RESILIENT
TYPES
MINIMUM
DEVELOP
evaluate
METRICS
BOUND
DETERMINE
time
Result
ANSWERS
INCREASES
study
EXECUTION
PROBLEM
COMP
run
ERENT
DECREASING
Quality
STEP
SORTING
AGGREGATION
ENVIRONMENTS
small
BETTER
TESTED
STRATEGIES
COMPARISONS
FOLLOWING
COMPARED
PERFORM
RESPONSES
Varying
best
PRACTICE
PROVIDED
Figure
PLURALITY
SECTION
FAMILIES
PARAMETERIZED
MAKE
applied
max
PROBABILITY
search
MAXIMUM
possible
observe
OBTAIN
SELECTED
multiple
EXAMINED
FUNCTION
ANALYSIS
human
MARKETPLACE
GENERAL
LARGE
EXAMPLE
COST
COMPENSATED
ACCURACY
LINE

Motivations

Humans are more effective than computers for many tasks

Motivations

Humans are more effective than computers for many tasks

- Identifying concepts in images

Motivations

Humans are more effective than computers for many tasks

- Identifying concepts in images
- Translating natural language

Motivations

Humans are more effective than computers for many tasks

- Identifying concepts in images
- Translating natural language
- Evaluating the usefulness of products

Definition

Crowdsourcing

Process of obtaining needed services, ideas, or content by soliciting contributions from a large group of people rather than from traditional employees or suppliers

Algorithm

In crowdsourcing algorithms **comparisons are done by humans** as opposed to traditional algorithms

Algorithm

In crowdsourcing algorithms **comparisons are done by humans** as opposed to traditional algorithms

The main challenge is in handling user **mistakes** or **variability**

Algorithm

In crowdsourcing algorithms **comparisons are done by humans** as opposed to traditional algorithms

The main challenge is in handling user **mistakes** or **variability**

Humans may give different answers in a comparison task

- Pick the wrong item
- Subjectivity

Definition

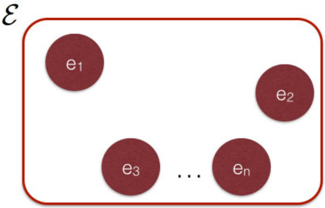
Max Algorithm

Important crowdsourcing algorithm that finds the best or maximum item in a set

Definition

Max Algorithm

Important crowdsourcing algorithm that finds the best or maximum item in a set



Max item $e^* \in \mathcal{E}$:
$e \leq e^* \, \forall e \in \mathcal{E} \setminus \{e^*\}$

Why?

- Most relevant URL for a given user query

Why?

- Most relevant URL for a given user query
- Find the best Facebook profile that matches a target person

Why?

- Most relevant URL for a given user query
- Find the best Facebook profile that matches a target person
- Pick the best photo that describes a restaurant
- ...

Comparisons

The maximum item is determined by a comparison operator

- **Comp**(S, r, R), asks r humans to compare the items in set $S \subseteq \mathcal{E}$ and combines the responses using aggregation rule R
- Probabilistic model to describe a worker
  - $\vec{p} = [p_1, p_2, ..., p_{|S|}]$
  - p_1 is the probability that the worker returns the maximum item, p_2 is the probability he returns the second best, and so on

Steps

Crowdsourcing algorithms are executed in **steps**

- A batch of comparisons C_1 is submitted during the first step to the marketplace and depending on the human answers, an appropriate set C_2 is selected for the second step

Steps

Crowdsourcing algorithms are executed in **steps**

- A batch of comparisons C_1 is submitted during the first step to the marketplace and depending on the human answers, an appropriate set C_2 is selected for the second step

For the selection of the C_i comparisons all answers from previous steps (1, 2, ..., i-1) can be considered

Problem Definition

Execution time

$Time(A, \mathcal{E})$, number of steps required for a max algorithm A to complete for input $\mathcal{E}$

Problem Definition

Execution time

$Time(A, \mathcal{E})$, number of steps required for a max algorithm A to complete for input $\mathcal{E}$

Cost

$Cost(A, \mathcal{E})$, total amount of money required for A to complete:

$\sum_{i=1}^{Time(A,\mathcal{E})} \sum_{Comp(S,r,R) \in C_i} \left[r \cdot Cost(|S|) \right]$

Problem Definition

Execution time

$Time(A, \mathcal{E})$, number of steps required for a max algorithm A to complete for input $\mathcal{E}$

Cost

$Cost(A, \mathcal{E})$, total amount of money required for A to complete:

$\sum_{i=1}^{Time(A,\mathcal{E})} \sum_{Comp(S,r,R) \in C_i} \left[r \cdot Cost(|S|) \right]$

Quality of the Results

$Quality(A, \mathcal{E})$, probability that max algorithm A returns the maximum item from input $\mathcal{E}$

Problem Definition

The focus is on maximizing the quality of the result, for a given cost budget B and a given time bound T

Formulation of the Problem

$$\begin{aligned} \text{maximize} \quad & Quality(A, \mathcal{E}) \\ \text{subject to} \quad & Cost(A, \mathcal{E}) \leq B \\ & Time(A, \mathcal{E}) \leq T \end{aligned}$$

Families of Max Algorithms

Two families

- Bubble
- Tournament

Families of Max Algorithms

Two families

- Bubble
- Tournament

They operate in steps and their parameters are

- r_i, the number of human responses seeked at step i
- s_i, the size of the sets compared by $Comp()$ at step i

Aggregation Rule

A max algorithm needs a rule to aggregate the responses

Plurality Rule

When comparison $Comp(S, r, R)$ is performed

- One of the items in S with the most votes is selected
- If there are items with same number of responses that is also the maximum, then one of this set is selected at random

Plurality Rule

Aggregation rules are important because they impact $Quality(A, \mathcal{E})$

$AggrQuality(s, r, \vec{p}, R)$

Probability that R returns the maximum of s items, assuming we collect r human responses, and each response follows a probabilistic distribution $\vec{p}$.

Plurality Rule

To calculate $AggrQuality(s, r, \vec{p}, R)$

- $S = \{e_1, e_2, \cdots, e_{|S|}\}$, $|S| = s$ and $e_s < e_{s-1} < \cdots < e_1$
- $\vec{p} = \left[p_1, p_2, \cdots, p_s\right]$, then p_i is the probability that a human response selected item e_i as the maximum in S

The number of received human responses for items $e_1, e_2, \cdots, e_s$ follows a multinomial distribution with parameter $\vec{p}$

Plurality Rule

$$AggrQuality(s, r, \vec{p}, R) = Pr\left[e_1 \text{ is returned}\right] =$$
$$= \sum_{l=1}^{s} Pr\left[e_1 \text{ is returned} \mid e_1 \in l \text{ winners}\right] \cdot Pr\left[e_1 \in l \text{ winners}\right]$$

Plurality Rule

$$AggrQuality(s, r, \vec{p}, R) = Pr\left[e_1 \text{ is returned}\right] =$$
$$= \sum_{l=1}^{s} Pr\left[e_1 \text{ is returned} \mid e_1 \in l \text{ winners}\right] \cdot Pr\left[e_1 \in l \text{ winners}\right]$$

$$Pr\left[e_1 \text{ is returned} \mid e_1 \in l \text{ winners}\right] = \frac{1}{l}$$
$$Pr\left[e_1 \in l \text{ winners}\right] = \sum_{n=1}^{r} Pr\left[e_1 \in l \text{ winners, with n responses each}\right]$$

Plurality Rule

Because the number of human responses per item follows the multinomial distribution

Plurality Rule

Because the number of human responses per item follows the multinomial distribution

$AggrQuality(s, r, \vec{p}, R)$

$$\sum_{l=1}^{s} \frac{1}{l} \cdot \sum_{n=1}^{r} \sum_{L \in \Gamma} \sum_{\substack{0 \leq k_i \leq n-1, i \in \bar{L} \\ \sum_{i \in \bar{L}} k_i + l \cdot n = r}} \left[\frac{r!}{(n!)^l \cdot \prod_{j \in \bar{L}} k_j!} \cdot \prod_{z \in L} {p_z}^n \cdot \prod_{w \in \bar{L}} {p_w}^{k_w} \right]$$

Plurality Rule

Because the number of human responses per item follows the multinomial distribution

$AggrQuality(s, r, \vec{p}, R)$

$$\sum_{l=1}^{s} \frac{1}{l} \cdot \sum_{n=1}^{r} \sum_{L \in \Gamma} \sum_{\substack{0 \le k_i \le n-1, i \in \bar{L} \\ \sum_{i \in \bar{L}} k_i + l \cdot n = r}} \left[\frac{r!}{(n!)^l \cdot \prod_{j \in \bar{L}} k_j!} \cdot \prod_{z \in L} {p_z}^n \cdot \prod_{w \in \bar{L}} {p_w}^{k_w} \right]$$

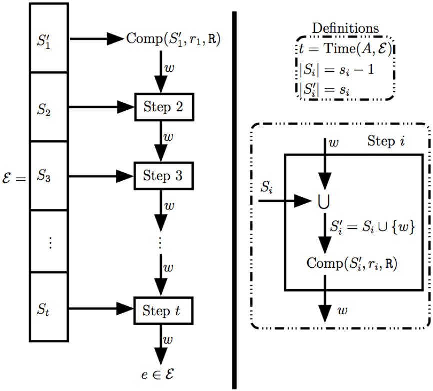
$AggrQuality(s, r, \vec{p}, R)$ is non-decreasing on r

Bubble Algorithm

Algorithm 1: $\mathcal{A}_{\mathrm{B}}(\{r_i\}, \{s_i\})$ operating on $\mathcal{E}$

if $\mathcal{E} == \{e\}$ **then**
    return e
$S'_1 \leftarrow$ random subset of $\mathcal{E}$ of size $\min(s_1, |\mathcal{E}|)$;
$\mathcal{E} \leftarrow \mathcal{E} \setminus S'_1$;
`//` w `is the winner of the last comparison performed`
$w \leftarrow \mathrm{Comp}(S'_1, r_1)$;
$i \leftarrow 2$;
while $\mathcal{E} \neq \emptyset$ **do**
    $S_i \leftarrow$ random subset of $\mathcal{E}$ of size $\min(s_i - 1, |\mathcal{E}|)$;
    $\mathcal{E} \leftarrow \mathcal{E} \setminus S_i$;
    $S'_i \leftarrow S_i \cup \{w\}$;
    $w \leftarrow \mathrm{Comp}(S'_i, r_i, \mathtt{R})$;
    $i \leftarrow i + 1$;
return w

Bubble Algorithm

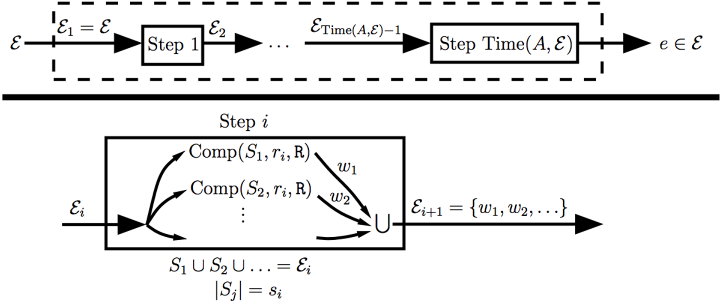


Tournament Algorithm

Algorithm 2: $\mathcal{A}_{\mathrm{T}}(\{r_i\}, \{s_i\})$ operating on $\mathcal{E}$

$i \leftarrow 1$;
$\mathcal{E}_i \leftarrow \mathcal{E}$;
while $|\mathcal{E}_i| \neq 1$ **do**
    partition $\mathcal{E}_i$ in non-overlapping sets S_j, with $|S_j| = s_i$ (the last set can have fewer items);
    $i \leftarrow i + 1$;
    $\mathcal{E}_i \leftarrow \bigcup_j \{C(S_j, r_i, \mathtt{R})\}$;
return $e \in \mathcal{E}_i$

Tournament Algorithm



Strategies for Max Algorithm

Several strategies can be considered in order to improve performance

- Determining heuristically paramenter $\{r_i\}$ and $\{s_i\}$
- Satisfying the budget and time constraints
- Based on *constant sequences* or *random hill climbing*

Constant Sequences

Idea

The practitioner decides how big the sets of items are and how many human responses he seeks per set of items

If $AggrQuality(s, r, \vec{p}, R)$ is non-decreasing on r, this strategy returns the optimal selection of r and s

Constant Sequences

Algorithm 3: ConstantSequences(x_0, $\vec{p}$, B, T, $\mathcal{A}$)

// $\hat{p}$, $\hat{r}$, and $\hat{s}$ is the best quality of result, best r, and best s seen so far for constant sequences

$(\hat{p}, \hat{r}, \hat{s}) \leftarrow (0.0, NULL, NULL)$;
for $s \leftarrow 2$ **to** m **do**
    $s_i \leftarrow s, \forall i$;
    $r \leftarrow$ maximum possible repetitions for the selected s and B;
    $r_i \leftarrow r, \forall i$;
    if Time$(\mathcal{A}(\{r_i\}, \{s_i\}), \mathcal{E}) > T$ **then**
        continue;
    $p \leftarrow$ Quality $(\mathcal{A}(\{r_i\}, \{s_i\}), \mathcal{E})$;
    if $p > \hat{p}$ **then**
        $(\hat{p}, \hat{r}, \hat{s}) \leftarrow (p, r, s)$;
return $(\hat{p}, \hat{r}, \hat{s})$;

Random Hill Climbing

Algorithm 4: RandomHillclimb(x_0, $\vec{p}$, B, T, $\mathcal{A}$)

```
(p̂, r̂, ŝ) ← ConstantSequences(x_0, p⃗, B, T, 𝒜);
// p̂ is the highest probability after a random source step is examined
r̂_i ← r̂, ∀i;
ŝ_i ← ŝ, ∀i;
repeat
    // c is a random source step
    c ← random step from tournament;
    // p̄ is the highest probability seen until after each target node is examined
    p̄ ← 0.0;
    // t is a random target step
    for t ← 1 to Time(𝒜({r̂_i}, {s_i}), ℰ) do
        {r_i} ← {r̂_i};
        // reducing the repetitions for the source step
        r_c ← r_c − 1;
        // increasing repetitions for the target step
        r_t ← maximum value that does not violate budget B;
        p ← Quality(𝒜({r_i}, {s_i}), ℰ);
        if p > p̄ then
            p̄ ← p;
            {r̄_i} ← {r_i};
    if p̄ > p̂ then
        p̂ ← p̄;
        {r̂_i} ← {r̄_i};
until p̄ < p̂ ;
return (p̂, {r̂_i}, {ŝ_i});
```

Other Strategies

AllPairsHillclimb

Extension of RandomHillclimb that considers all possible steps as sources c *(not just a random one)*

AllSAllPairsHillclimb

Generalization of the AllPairsHillclimb that considers all possible s's

VaryingS

Algorithm 5: $\texttt{VaryingS}(x_0,\ \vec{p},\ B,\ T,\ \mathcal{A})$

// global optimal values

$(\hat{p}, \{\hat{r}_i\}, \{\hat{s}_i\}) \leftarrow (0.0, NULL, NULL)$;

$(p, \{r_i\}, \{s_i\}) \leftarrow \texttt{AllSAllPairsHillclimb}(x_0, \vec{p}, B, T, \mathcal{A})$;

$u \leftarrow 1$;

$\hat{r}_u \leftarrow r_1$;

$\hat{s}_u \leftarrow s_1$;

$b_1 \leftarrow$ budget consumed from the selection of $\hat{r}_1$ and $\hat{s}_1$ on an input of size x_0 ;

while $x_u > 1$ **do**

  $u \leftarrow u + 1$;

  $(p, \{r_i\}, \{s_i\}) \leftarrow \texttt{AllSAllPairsHillclimb}(x_{u-1}, \vec{p}, B - b_{u-1}, T - (u-1), \mathcal{A})$;

  $\hat{r}_u \leftarrow r_1$;

  $\hat{s}_u \leftarrow s_1$;

  $b_u \leftarrow$ budget consumed from the selection of $\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_u$ and $\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_u$ on an input of size x_0 ;

$\hat{p} \leftarrow \text{Quality}(\mathcal{A}(\{\hat{r}_i\}, \{\hat{s}_i\}), \mathcal{E})$;

return $(\hat{p}, \{\hat{r}_i\}, \{\hat{s}_i\})$;

Human Error Models

Given a set of items $S = \{e_1, e_2, \cdots, e_{|S|}\}$ to humans, the *error model* assigns probabilities to each (*possible*) response of a human

- e_i represents i^{th} best item in $\mathcal{E}$
- A human response has probability p_i of returning item e_i

Proximity/Order-Based Error Model

- Parameter $p \in \left[\frac{1}{|S|}, 1\right]$

Proximity/Order-Based Error Model

- Parameter $p \in \left[\frac{1}{|S|}, 1\right]$
- Distance function $d(\cdot,\cdot) \in (0,1)$ that compares how different two items are
- In Order-Based Model $d(e_i, e_j) = \frac{\left|rank(e_i,S) - rank(e_j,S)\right|}{|S|}$
  - $rank(e_i, S)$, is defined as the number of items in S that are better than e_i plus 1

Proximity/Order-Based Error Model

- Parameter $p \in \left[\frac{1}{|S|}, 1\right]$
- Distance function $d(\cdot, \cdot) \in (0, 1)$ that compares how different two items are
- In Order-Based Model $d(e_i, e_j) = \frac{\left|rank(e_i,S) - rank(e_j,S)\right|}{|S|}$
  - $rank(e_i, S)$, is defined as the number of items in S that are better than e_i plus 1
- A worker returns e_i with probability

$$\begin{cases} p_1 = p \\ p_i = (1-p) \cdot \frac{1-d(e_i,e_1)}{\sum_{j=2}^{|S|}\left[1-d(e_j,e_1)\right]}, i \in \{2, 3, \cdots, |S|\} \end{cases}$$

Linear Error Model

- Probability that a worker selects the maximum item
  - $p_1 = 1 - p_e - s_e \cdot (|S| - 2)$

Linear Error Model

- Probability that a worker selects the maximum item
  - $p_1 = 1 - p_e - s_e \cdot (|S| - 2)$
- When the worker fails to return the maximum item, he returns a random item from S
  - Each item in $\{e_2, \cdots, e_{|S|}\}$ is selected with probability

    $$\frac{1 - p_e - s_e \cdot (|S| - 2)}{|S| - 1}$$

Constant Error Model

- This model assumes that the human worker is able to determine the maximum item from S with probability $p \in \left[\frac{1}{|S|}, 1\right]$, for any S

Constant Error Model

- This model assumes that the human worker is able to determine the maximum item from S with probability $p \in \left[\frac{1}{|S|}, 1\right]$, for any S
- In the event that the worker is not able to determine the maximum item from S, he returns one random non-maximum item from S
  - Each item in $\{e_2, \cdots, e_{|S|}\}$ has probability $\frac{1-p}{|S|-1}$

Human Cost Models

Constant Model

$Cost(|S|) = c$ for some constant c
(each task has a fixed price)

Human Cost Models

Constant Model

$Cost(|S|) = c$ for some constant c
(each task has a fixed price)

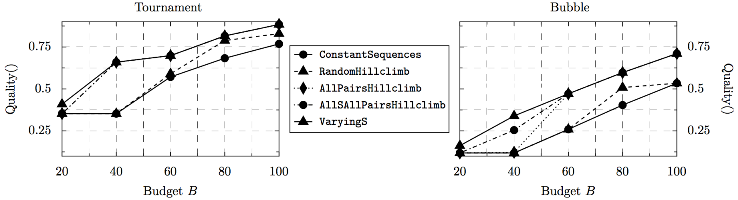
Linear Model

$Cost(|S|) = c + s_c \times (|S| - 2)$, for some constants c and s_c
(each task has a price that depends on $|S|$)

Max Algorithms and Strategies Performance

The Max Algorithms, *Bubble* and *Tournament*, are evaluated using strategies analyzed and human models with parameters

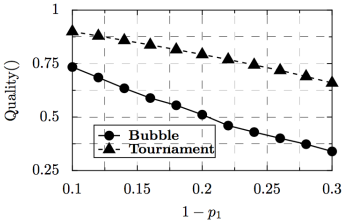
- $|\mathcal{E}| = 100$, time bound $T = \infty$ and the budget $20 \leq B \leq 100$
- Order-based error model with $p = 0.78$
- Linear cost model with $c = 1$ and $s_c = 0.1$
- $|S| \leq 10$



Error Model

The performance of the bubble and tournament max algorithms are evaluated for various values of $1 - p_1$, the probability of a human making an error, using VaryingS strategy with parameters

- $|\mathcal{E}| = 100$ and $B = 40$
- Order-based error model and $0.1 \leq 1 - p_1 \leq 0.3$
- Linear cost model with $c = 1$ and $s_c = 0.1$
- $|S| \leq 7$



Other Metrics

These values are obtained by simulating the application of $A(\mathcal{E})$ E times

Mean Reciprocal Rank

$\frac{1}{E}\sum_{i=1}^{E}\frac{1}{rank_i}$, where $rank_i$ is the rank of the returned item in the i^{th} simulation

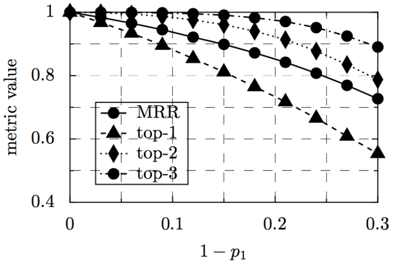
Top-k

The fraction of the E simulations for which $A(\mathcal{E})$ belonged in the top-k items of $\mathcal{E}$

Other Metrics

Quality() and *top* − 1 are the same: the probability that an algorithm returns the maximum/*top* − 1 item

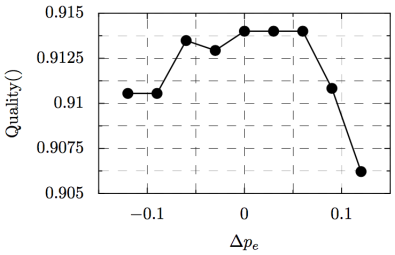
These metrics are used to evaluate VaryingS on tournament algorithm



Error Model Parameters Sensitivity

How sensitive algorithms are to the Error Model Parameters?

- Linear error model with parameters $p_e = 0.15$ and $s_e = 0.02$
- Crowdsourcing marketplace with parameters $p_e^{'} \in (0, 0.3)$ and $s_e^{'} = s_e = 0.02$



Conclusions

The type of worker errors impact results accuracy, but not what is the best algorithm/strategy

Conclusions

The type of worker errors impact results accuracy, but not what is the best algorithm/strategy

It pays off to vary the size of a task and to vary/optimize the number of repetitions

Conclusions

The type of worker errors impact results accuracy, but not what is the best algorithm/strategy

It pays off to vary the size of a task and to vary/optimize the number of repetitions

Finding the maximum item with high accuracy is expensive, unless workers are very reliable

Conclusions

The type of worker errors impact results accuracy, but not what is the best algorithm/strategy

It pays off to vary the size of a task and to vary/optimize the number of repetitions

Finding the maximum item with high accuracy is expensive, unless workers are very reliable

Future Works

Take into account the event of "no answers" in the max algorithms and/or retrieve the top-k items from a set and sort them

Thanks for your Attention