# Unsupervised Rank Aggregation with Distance-Based Models

Kevin Small
Tufts University

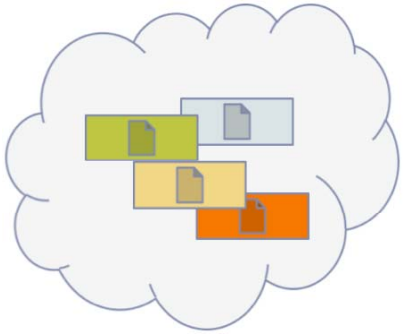
Collaborators: Alex Klementiev (Johns Hopkins University)
Ivan Titov (Saarland University)
Dan Roth (University of Illinois)

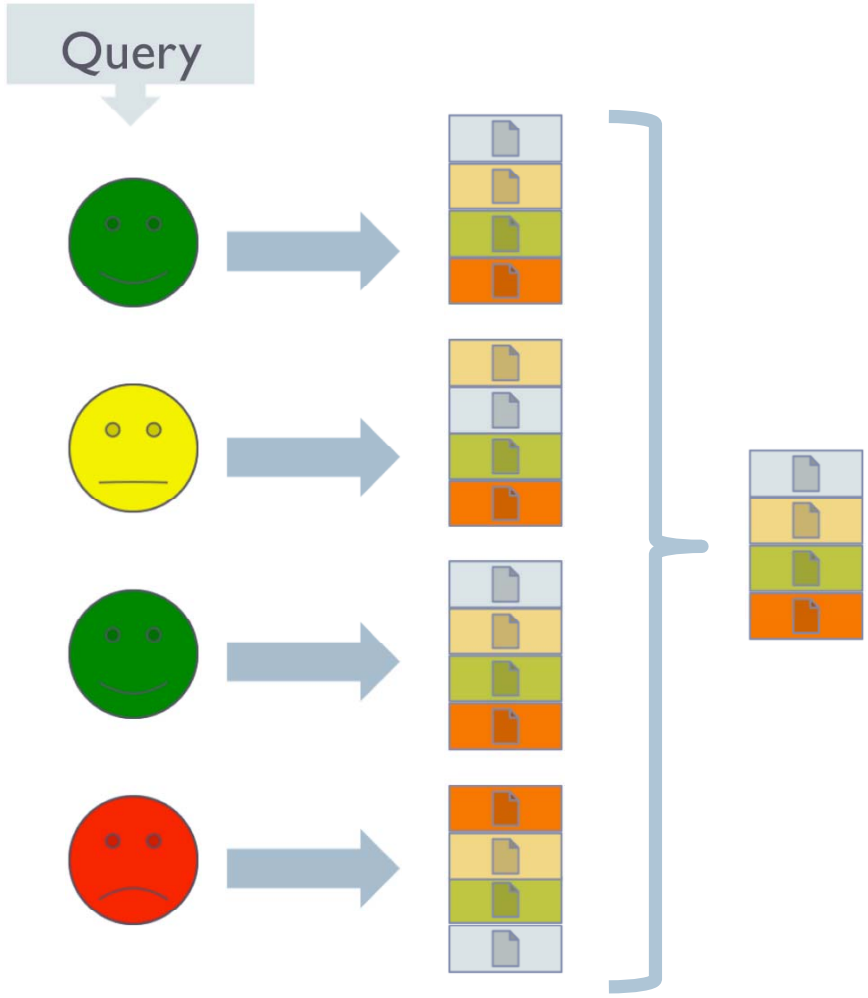
Cognitive Computation Group
University of Illinois at Urbana-Champaign

Tufts University

# Motivation

- Consider a panel of judges
  - Each independently generates (partial) rankings over objects to the best of their ability






- The need to meaningfully aggregate their output is a fundamental problem
  - Applications are plentiful in Information Retrieval and Natural Language Processing

# Multilingual Named Entity Discovery

- *Named Entity Discovery [Klementiev & Roth, ACL 06]*: given a bilingual corpus one side of which is annotated with Named Entities, find their counterparts in the other

guimaraes

| Candidate | r1 | r2 | r3 | r4 |
|---|---|---|---|---|
| гуимареша | 1 | 1 | 3 | 3 |
| муаммар | 3 | 27 | 4 | 14 |
| гимараешу | 2 | 3 | 1 | 1 |
| футбол | 59 | 2 | 31 | 12 |
| гамма | 7 | 17 | 51 | 32 |
| ... | | | | |

- NEs are often transliterated: rank according to a *transliteration model score*
- NEs tend to co-occur across languages: rank according to *temporal alignment*
- NEs tend to co-occur in similar contexts: rank according to *contextual similarity*
- NEs tend to co-occur in similar topics: rank according to *topic similarity*
- etc.

# Overview of Our Approach

- We propose a formal framework for *unsupervised* structured label aggregation
  - Judges independently generate a (partial) labeling attempting to reproduce the true underlying label based on their expertise in a given domain
  - We derive an EM-based algorithm treating the votes of individual judges and the true label as the observed and unobserved data, respectively
- *Intuition*: experts in a given domain are better at generating votes close to true ranking and will tend to agree with each other, while the non-experts will not
- We instantiate the framework for the cases of combining *permutations*, combining *top-k* lists, and combining *dependency parses*.

# Notation

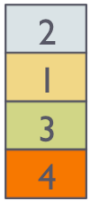
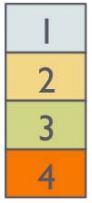
- Permutation $\pi$ over $n$ objects $x_1 \ldots x_n$
  - e = (1,2,...,n) is the identity permutation


- Set $S_n$ of all n! permutations
- Distance d : $S_n \times S_n \to R_+$ between permutations
  - E.g. *Kendall's tau distance*: minimum number of adjacent transpositions

$d_K($  ,  $) = 3$



- d is assumed to be invariant to arbitrary re-labeling of the n objects
  - $d(\pi, \pi) = d(e, \pi) = D(\pi)$. If $\pi$ is a r.v., so is $D=D(\pi)$

$d_K($  ,  $) = d_K($  ,  $) = D_K($  $) = 3$

# Background: Mallows Models

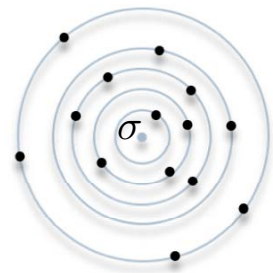
Uniform when θ = 0

"Peaky" when |θ| is large

Expensive to compute

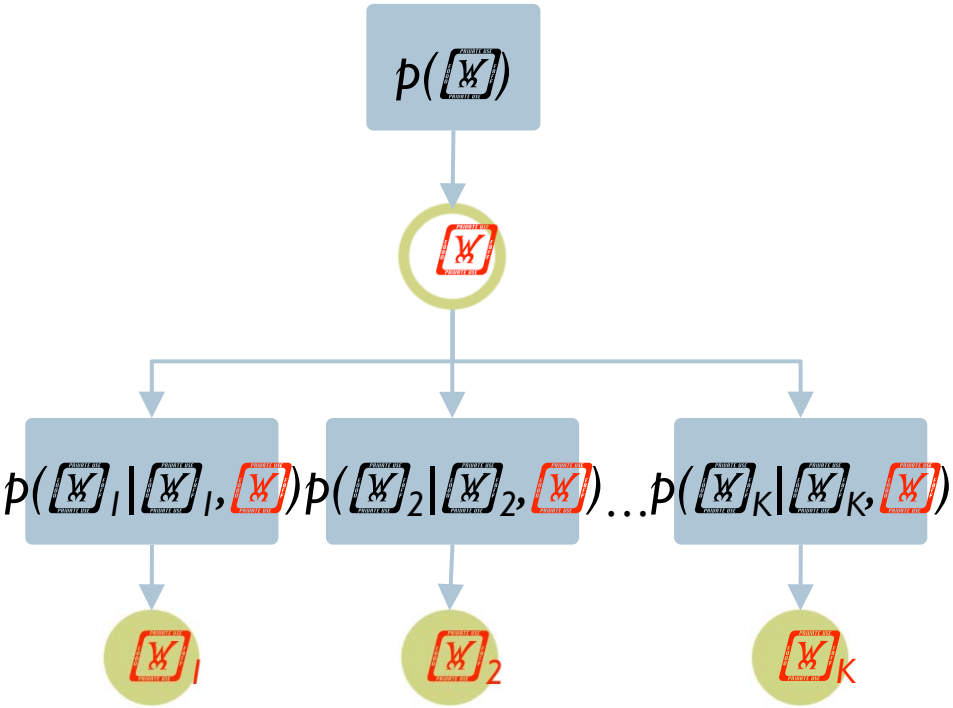
$$p(\pi|\theta,\sigma) = \frac{1}{Z(\theta,\sigma)} \exp(\theta\ d(\pi,\sigma))$$

where $Z(\theta,\sigma) = \sum_{\pi \in \mathcal{S}_n} \exp(\theta\ d(\pi,\sigma))$




- $\theta \in \mathbb{R},\ \theta \leq 0$ is the *dispersion* parameter
- $\sigma \in \mathcal{S}_n$ is the *location* parameter
- *d(·,·)* right-invariant, so $Z(\theta,\sigma)$ does not depend on $\sigma$
- If *D* can be decomposed $D(\pi) = \sum_{i=1}^{m} V_i(\pi)$ where $V_i$ are indep. r.v.'s, then $E_\theta(D)$ may be efficient to compute *[Fligner and Verducci '86]*

# Generative Story for Aggregation




Generate the *true* π according to prior p(π)

Draw $\sigma_1 \ldots \sigma_K$ *independently* from K Mallows models $p(\sigma_i|\theta_i, \pi)$, with the *same location parameter* π

$$p(\pi, \boldsymbol{\sigma}|\boldsymbol{\theta}) = p(\pi) \prod_{i=1}^{K} p(\sigma_i|\theta_i, \pi)$$

# Background: Extended Mallows Models

The associated conditional model (when votes of *K* judges $\boldsymbol{\sigma} \in \mathcal{S}_n^K$ are available) proposed in *[Lebanon and Lafferty '02]*:

$$p(\pi|\boldsymbol{\theta},\boldsymbol{\sigma}) = \frac{1}{Z(\boldsymbol{\theta},\boldsymbol{\sigma})} p(\pi) \exp\left(\sum_{i=1}^{K} \theta_i \; d(\pi,\sigma_i)\right)$$

where $Z(\boldsymbol{\theta},\boldsymbol{\sigma}) = \sum_{\pi \in \mathcal{S}_n} p(\pi) \exp(\sum_{i=1}^{K} \theta_i \; d(\pi,\sigma_i))$

Free parameters $\boldsymbol{\theta} \in \mathbb{R}^K, \; \boldsymbol{\theta} \leq \mathbf{0}$ represent the degree of expertise of individual judges.

It is straightforward to generalize both models to partial rankings by constructing *appropriate distance functions*

# Outline

- Motivation
- Problem Statement
- Overview of our approach
- Background
  - Mallows models / Extended Mallows models

Introduction and background

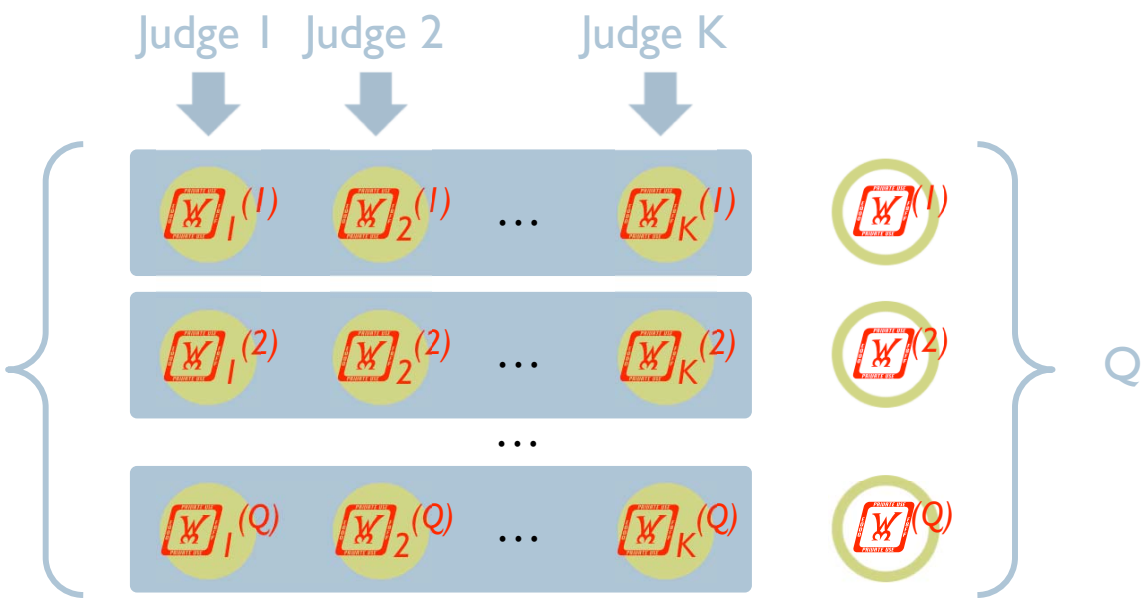
- Unsupervised Learning and Inference
  - Incorporating domain-specific expertise
- Instantiations of the framework
  - Combining permutations / top-k lists
- Experiments
- Dependency Parsing
- Conclusions

Our contribution

# Our Approach

[ICML 2008]

- We propose a formal framework for <u>unsupervised rank aggregation</u> based on the *extended Mallows model* formalism
- We derive an EM-based algorithm to estimate model parameters $\boldsymbol{\theta}$
  - Observed data: votes of individual judges
  - Unobserved data: true ranking



# Learning

Denoting $\theta'$ to be the value of parameters from the previous iteration, the M step for the i[th] ranker is:

$$\underbrace{E_{\theta_i}(D)}_{\text{LHS}} = \underbrace{\sum_{\substack{(\pi^{(1)},\ldots,\pi^{(Q)}) \\ \in \mathcal{S}_n^Q}} \left( \frac{1}{Q} \sum_{q=1}^{Q} d(\pi^{(q)}, \sigma_i^{(q)}) \right) \prod_{j=1}^{Q} p\left(\pi^{(j)} | \boldsymbol{\theta'}, \boldsymbol{\sigma}^{(j)}\right)}_{\text{RHS}}$$

In general, > n! computations

Average distance between votes of the i[th] ranker and $\pi^{(1..Q)}$

Marginal of the unobserved data $\pi^{(1..Q)}$

# Learning and Inference

$$\underbrace{E_{\theta_i}(D)}_{\text{LHS}} = \underbrace{\sum_{\substack{(\pi^{(1)},\ldots,\pi^{(Q)}) \\ \in \mathcal{S}_n^Q}} \left( \frac{1}{Q} \sum_{q=1}^{Q} d(\pi^{(q)}, \sigma_i^{(q)}) \right) \prod_{j=1}^{Q} p\left(\pi^{(j)} | \boldsymbol{\theta}', \boldsymbol{\sigma}^{(j)}\right)}_{\text{RHS}}$$

## Learning (estimating $\Psi$)

- For K constituent rankers, repeat:
  - Estimate the RHS given current parameter values $\Psi$
    - Sample with Metropolis-Hastings
    - Or use heuristics
  - Solve the LHS to update $\Psi$
    - Efficient estimation can be done for particular types of distance functions

Depends on structure type, more about this later

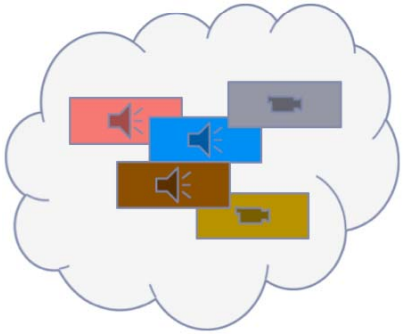
## Inference (computing the most likely ranking)

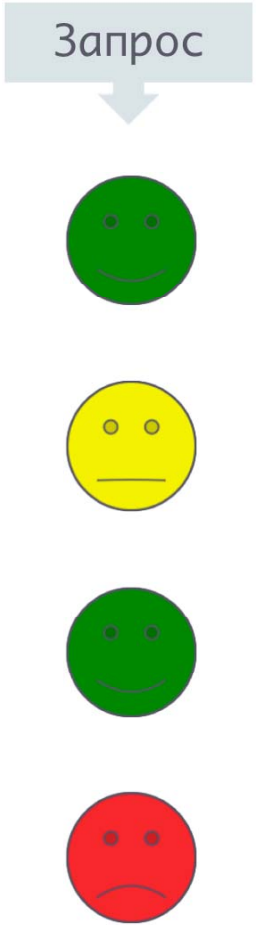
- Sample with Metropolis-Hastings or use heuristics

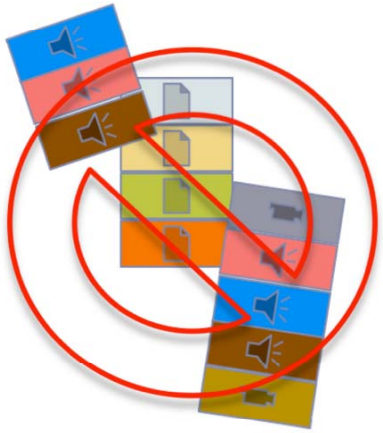
# Domain-specific expertise?

[IJCAI 2009]

- Relative expertise may not stay the same
  - May depend on the type of objects
  - May depend on the type of query





- Typically, ranked supervised data to estimate judges' expertise is very expensive to obtain
  - Especially for multiple types

# Mallows Models with Domain-Specific Expertise

The associated conditional model (when votes of *K* judges $\boldsymbol{\sigma} \in \mathcal{S}_n^K$ are available) can be derived:

$$p(\pi, t|\boldsymbol{\sigma}, \boldsymbol{\theta}, \boldsymbol{\alpha}) = \alpha_t \frac{\exp\left(\sum_{i=1}^{K} \theta_{t,i}\ d(\pi, \sigma_i)\right)}{Z(\boldsymbol{\theta}, \boldsymbol{\sigma})}$$

Free parameters $\boldsymbol{\theta} \in \mathbb{R}^{T \times K}, \boldsymbol{\theta} \leq \mathbf{0}$ represent the degree of expertise of individual judges. $\boldsymbol{\alpha} \in \mathbb{R}^T$ are the mixture weights.

Note: it is straightforward to generalize these models to other structured labels (e.g. partial rankings) by constructing *appropriate distance functions*

# Learning

$$\alpha_t = \frac{1}{Q}\sum_{j=1}^{Q}\sum_{\pi^{(j)}\in\mathcal{S}_n} p(\pi^{(j)}, t|\boldsymbol{\sigma}^{(j)}, \boldsymbol{\theta'}, \boldsymbol{\alpha'}) \quad (1)$$

$$E_{\theta_{t,i}}(D) = \frac{1}{\alpha_t Q}\sum_{j=1}^{Q}\sum_{\pi^{(j)}\in\mathcal{S}_n} d(\pi^{(j)}, \sigma_i^{(j)}) p(\pi^{(j)}, t|\boldsymbol{\sigma}^{(j)}, \boldsymbol{\theta'}, \boldsymbol{\alpha'}) \quad (2)$$

$$(3)$$

- For each of i<sup>th</sup> ranker and t<sup>th</sup> type:
  - Estimate $\alpha_t$ (1) and $E_{\theta_{t,i}}(D)$ (2) given current parameter values $\boldsymbol{\theta}'$ and $\boldsymbol{\alpha}'$
  - Solve 3 to update $\theta_{t,i}$
- Repeat

# Instantiating the Framework

- We have *not committed* to a particular type of structure
- In order to instantiate the framework:
  - Design a distance function appropriate for the setting
    - If a function if right invariant and decomposable [LHS] estimation can be done quickly
  - Design a sampling procedure for learning [RHS] and inference

# Case 1: Combining Permutations [LHS]

- Kendall tau distance $D_K$ is the minimum number of adjacent transpositions needed to transform one permutation into another

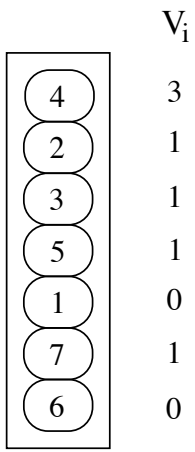
- Can be decomposed into a sum of independent random variables:

$$D_K(\pi) = \sum_{i=1}^{n-1} V_i(\pi) \text{ where } V_i(\pi) = \sum_{j>i} I(\pi^{-1}(i) - \pi^{-1}(j))$$

- And the expected value can be shown to be:

$$E_\theta(D_K) = \frac{ne^\theta}{1-e^\theta} - \sum_{j=1}^{n} \frac{je^{\theta j}}{1-e^{\theta j}}$$

Monotonically decreasing, can find $\theta$ *with line search quickly*



# Case 1: Combining Permutations [RHS]

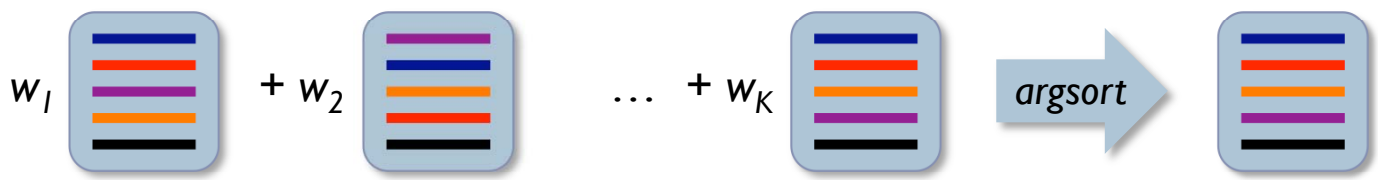
## Sampling from the base chain of random transpositions

- Start with a random permutation
- If chain is at $\pi$, randomly transpose two objects forming $\pi'$
  - If $a = p(\pi'|\boldsymbol{\theta}, \boldsymbol{\sigma})/p(\pi|\boldsymbol{\theta}, \boldsymbol{\sigma}) \geq 1$ chain moves to $\pi'$
  - Else, chain moves to $\pi'$ with probability $a$
- Note that we can compute distance incrementally, i.e. add the change due to a single transposition
- Convergence
  - *n log(n)* if *d is Cayley's distance [Diaconis '98]*, likely similar for some others
  - No convergence results for general case, but it works well in practice

# Case 1: Combining Permutations [RHS]

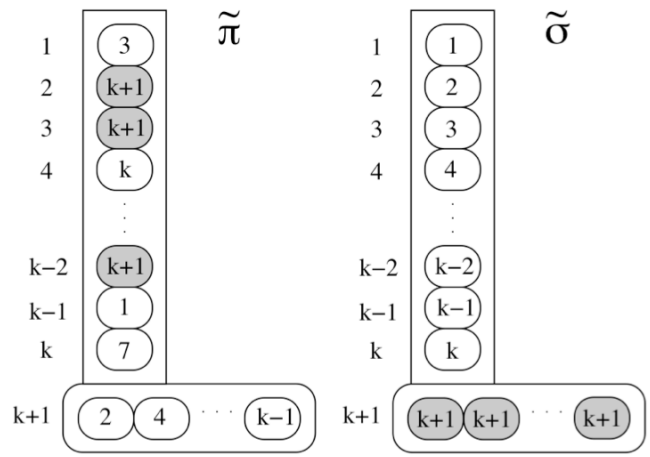
An alternative heuristic: weighted *Borda* count, i.e.

- Linearly combine ranks of each object and argsort
- Model parameters $\boldsymbol{\theta}$ represent relative expertise, so it makes sense to weigh rankers as $w_i = e^{(-\theta_i)}$



# Case 2: Combining Top-k [LHS]

- We extend Kendall tau to top-k




*r* grey boxes
*z* white boxes
*r* + *z* = *k*

$$\tilde{D}_K(\tilde{\pi}) = \sum_{\substack{i=1 \\ \tilde{\pi}^{-1}(i) \notin Z}}^{k} \tilde{U}_i(\tilde{\pi}) + \frac{r(r+1)}{2} + \sum_{\substack{i=1 \\ \tilde{\pi}^{-1}(i) \in Z}}^{k} \tilde{V}_i(\tilde{\pi})$$

Bring grey boxes to bottom

Switch with objects in (k+1)

Kendall's tau for the k elements

# Case 2: Combining Top-k [LHS & RHS]

- R.v.'s $\tilde{V}_i$ and $\tilde{U}_i$ are independent, we can use the same trick to show that [LHS] is:

$$E_\theta(\tilde{D}_K) = \frac{ke^\theta}{1-e^\theta} - \sum_{j=r+1}^{k} \frac{je^{j\theta}}{1-e^{j\theta}} + \frac{r(r+1)}{2} - r(z+1)\frac{e^{\theta(z+1)}}{1-e^{\theta(z+1)}}$$

  - Also monotonically decreasing, can again use line search
  - Both $\tilde{D}_K$ and $E_\theta(\tilde{D}_K)$ reduce to Kendall tau results when same elements are ranked in both lists, i.e. *r = 0*

- Sampling / heuristics for [RHS] and inference are similar to the permutation case

# Outline

- Motivation
- Problem Statement
- Overview of our approach
- Background
  - Mallows models / Extended Mallows models

Introduction and background

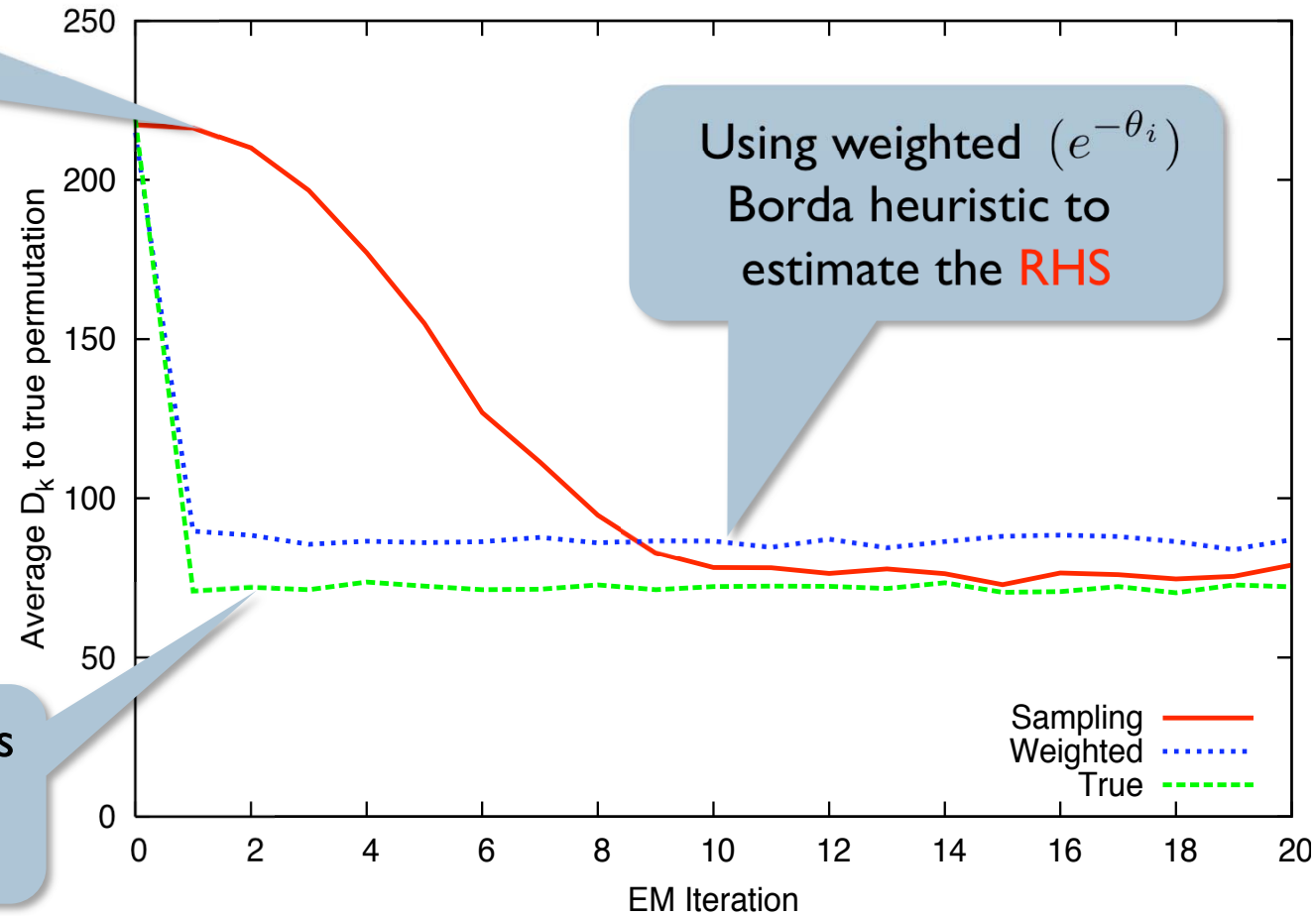
- Unsupervised Learning and Inference
  - Incorporating domain-specific expertise
- Instantiations of the framework
  - Combining permutations / top-k lists
- Experiments
- Dependency Parsing
- Conclusions

Our contribution

# *Exp. 1* Combining permutations

Using sampling to estimate the RHS

- *Judges*: $K$ = 10 (Mallows models)
- *Objects*: $n$ = 30
- $Q$ = 10 sets of votes




Using weighted $(e^{-\theta_i})$ Borda heuristic to estimate the RHS

Using true rankings to evaluate the RHS

# *Exp. 2* Meta-search dispersion parameters

- *Judges*: $K$ = 4 search engines (S1, S2, S3, S4)
- *Documents*: Top $k$ = 100
- *Queries*: $Q$ = 50 queries

Define Mean Reciprocal Page Rank (MRPR): mean rank of the page containing the correct document

- Our model gets ***0.92***

| | $S1$ | $S2$ | $S3$ | $S4$ |
|---|---|---|---|---|
| $\theta$ | -0.065 | 0.0 | -0.066 | -0.049 |
| $MRPR$ | 0.86 | 0.43 | 0.82 | 0.78 |

Model parameters correspond to ranker quality

# *Exp. 3* Top-k rankings: robustness to noise

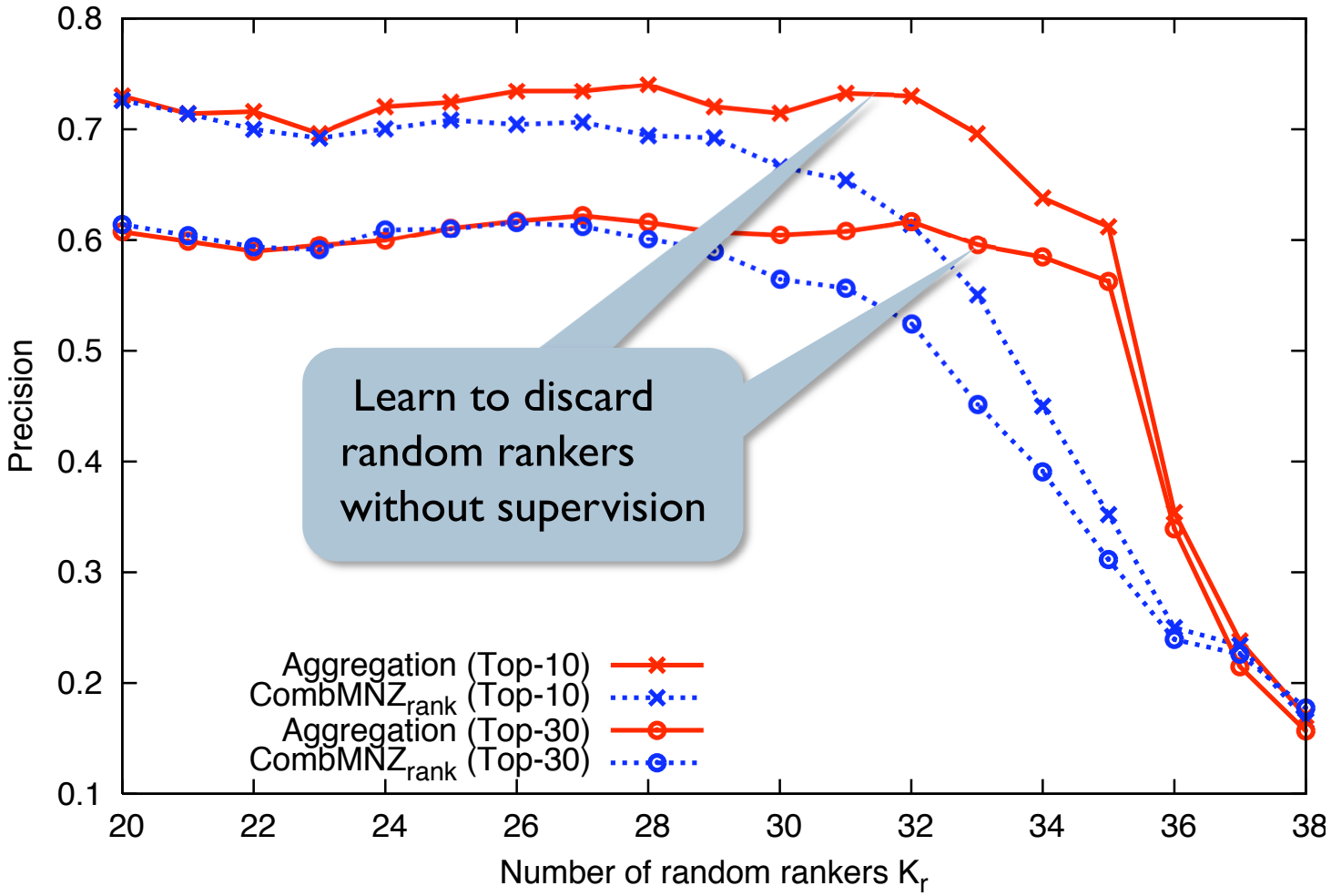
- *Judges: K* = 38 TREC-3 ad-hoc retrieval shared task participants
- *Documents*: Top *k* = 100 documents
- *Queries: Q* = 50 queries

Replaced $K_r \in [0, K]$ randomly chosen participants with random rankers.
Baseline: rank objects according to score:

$$CombMNZ_{rank} = N_x \times \sum_{i=1}^{K} (k - r_i(x, q))$$

where $r_i(x, q)$ is the rank of *x* returned by *i* for query *q*, and $N_x$ is the number of participants with *x* in top-k

# *Exp. 3* Top-k rankings: robustness to noise



# Outline

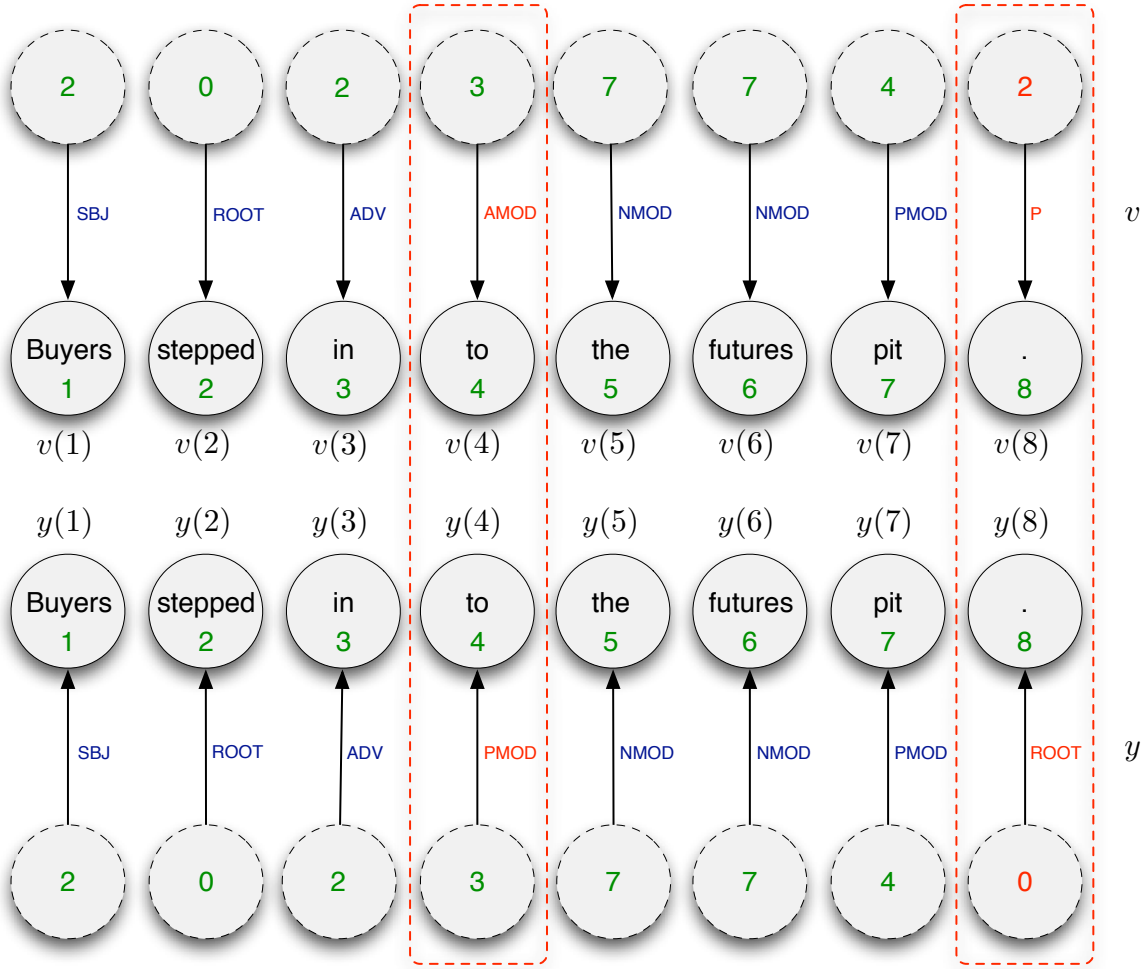
- Motivation
- Problem Statement
- Overview of our approach
- Background
  - Mallows models / Extended Mallows models

Introduction and background

- Unsupervised Learning and Inference
  - Incorporating domain-specific expertise
- Instantiations of the framework
  - Combining permutations / top-k lists
- Experiments
- Dependency Parsing
- Conclusions

Our contribution

# Dependency Parses




- Let $v(i)$ denote a pair of head offset and label, i.e. a *link*
- Labeled attachment score is link accuracy, trivially computed from *Hamming* distance:

$$d_H(v, y) = \sum_{i=1}^{n} [\![ v(i) \neq y(i) ]\!]$$

# Dependency Parses: Parameter Estimation

- Parameter estimation of the type-agnostic model can be done directly
- Let us assume there are exactly $|\mathcal{S}|$ possibilities for each link, and that $j^{\text{th}}$ (of $Q$) sentences has $n^{(j)}$ words, $\sum_{j=1}^{Q} n^{(j)} = N$.
- On each round of training the learning procedure for the type-agnostic model is equivalent to:

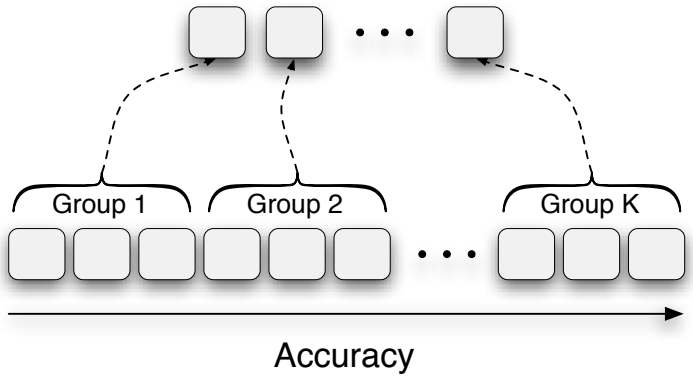
$$\theta_i = \log R'_i - \log(1 - R'_i) - \log\left(|\mathcal{S}| - 1\right)$$

$$R'_i = \frac{1}{N} \sum_{j=1}^{Q} \sum_{l=1}^{n^{(j)}} \frac{\sum_{v \in \mathcal{S}} [\![ v \neq y^{(j)}_{i,(l)} ]\!] \exp\left(\sum_{i=1}^{K} \theta'_i [\![ v \neq y^{(j)}_{i,(l)} ]\!]\right)}{\sum_{v \in \mathcal{S}} \exp\left(\sum_{i=1}^{K} \theta'_i [\![ v \neq y^{(j)}_{i,(l)} ]\!]\right)}$$

- With small $|\mathcal{S}|$, parameter estimation can be done quickly!

# Dependency Parses: Aggregation

- Dependency parsers are CoNLL-2007 shared task participants
  - 10 languages: Arabic, Basque, Catalan, Chinese, Czech, English, Greek, Hungarian, Italian, and Turkish
  - 131 to 690 sentences and 4513 to 5390 words, depending on the language
  - Between 20 and 23 systems, depending on the language
- Varied the number of participants attempting to represent expertise in the entire pool




- Baseline: majority vote on each link (ties broken randomly)

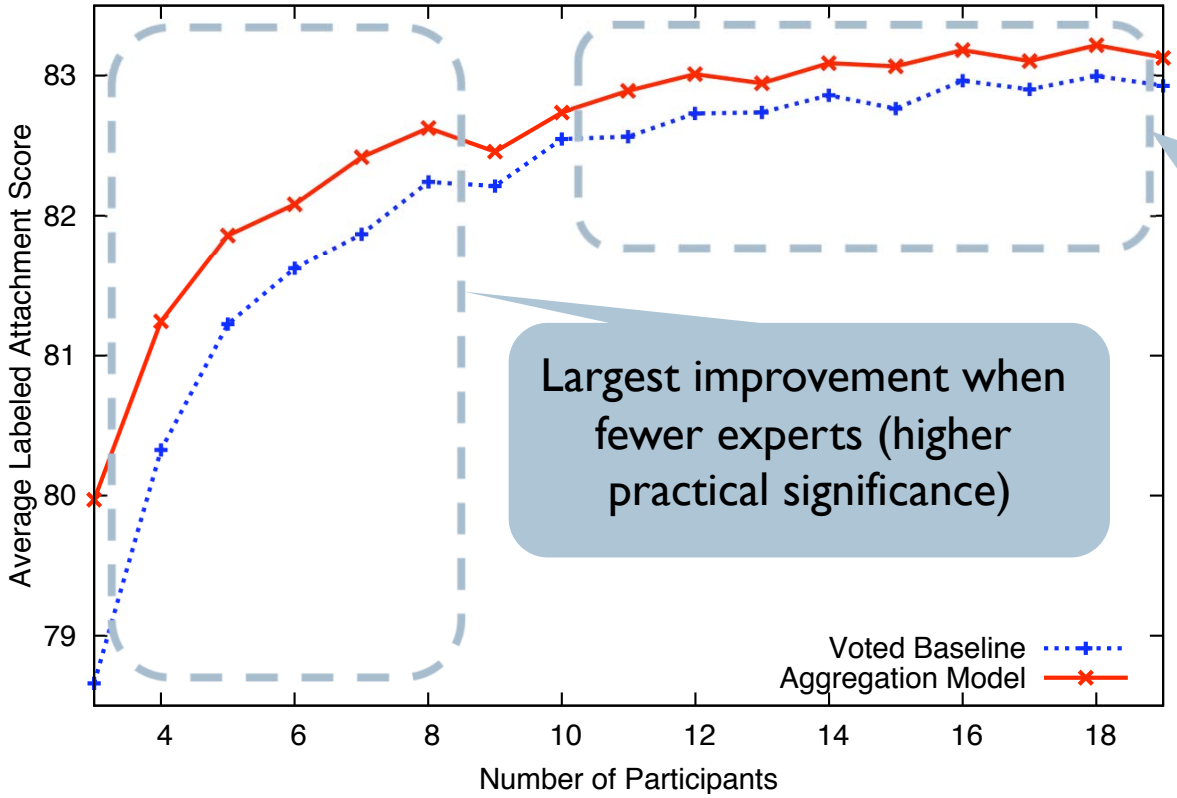
# Predicting Relative Performance

- Estimation of relative expertise correlates with true relative expertise
- Results for Italian
  - Labeled Attachment

| Participant | $\theta$ | True Rank | True Performance |
|---|---|---|---|
| jni@msi.vxu.se | -2.54 | 1 | 84.40 |
| sagae@is.s.u-tokyo.ac.jp | -2.52 | 2 | 83.91 |
| nakagawa378@oki.com | -2.29 | 3 | 83.61 |
| johan.hall@vxu.se | -2.28 | 5 | 82.48 |
| carreras@csail.mit.edu | -2.21 | 4 | 83.46 |
| chenwl@nict.go.jp | -2.13 | 7 | 82.04 |
| attardi@di.unipi.it | -2.01 | 8 | 81.34 |
| xyduan@nlpr.ia.ac.cn | -2.00 | 9 | 80.75 |
| ivan.titov@cui.unige.ch | -1.95 | 6 | 82.26 |
| dasmith@jhu.edu | -1.92 | 10 | 80.69 |
| michael.schiehlen@ims.uni-stuttgart.de | -1.83 | 11 | 80.46 |
| bcbb@db.csie.ncu.edu.tw | -1.79 | 12 | 78.79 |
| prashanth@research.iiit.ac.in | -1.76 | 13 | 78.67 |
| richard@cs.lth.se | -1.67 | 14 | 77.55 |
| nguyenml@jaist.ac.jp | -1.51 | 16 | 75.06 |
| joyce840205@gmail.com | -1.50 | 17 | 74.65 |
| s.v.m.canisius@uvt.nl | -1.49 | 15 | 75.57 |
| francis.maes@lip6.fr | -1.30 | 18 | 73.63 |
| zeman@ufal.mff.cuni.cz | -0.68 | 19 | 62.13 |
| svetoslav.marinov@his.se | -0.49 | 20 | 59.75 |

# Dependency Parses: Aggregation

- Performance measured as average accuracy over 10 languages




Number of good experts grows: voted baseline is harder to beat

# Conclusions

- Propose a formal mathematical and algorithmic framework for aggregating (partial) structured labels *without* supervision
  - Show that learning can be made efficient for decomposable distance functions
- Instantiate the framework for combining *permutations*, combining *top-k* lists, and *dependency parses*
  - Introduce a novel distance function for *top-k* and *dependency parses*

# Thanks!