Finding Answer Passages with Rank Optimizing Decision Trees

Ingo Glöckner

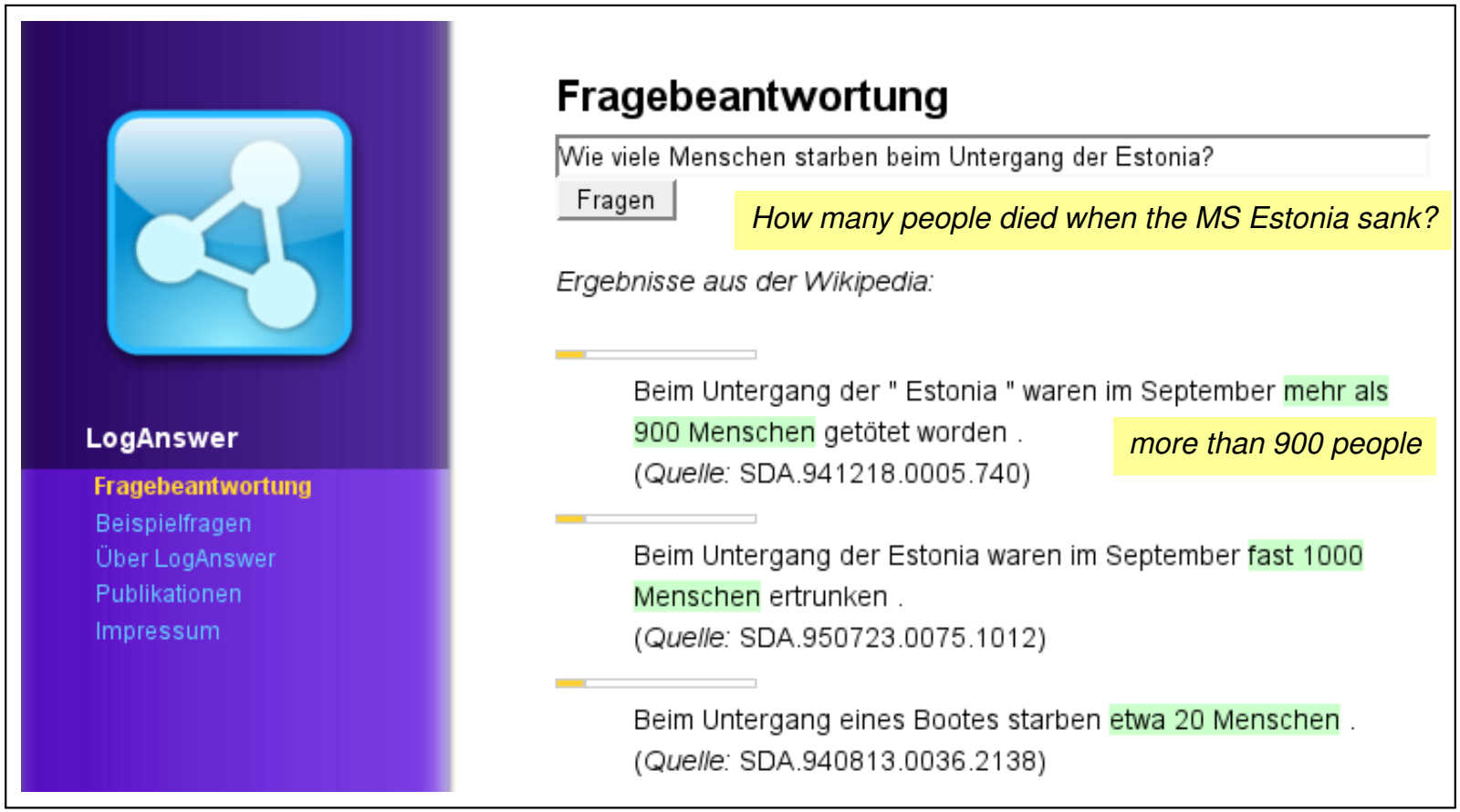
Intelligent Information and Communication Systems Group (IICS)
FernUniversität in Hagen, 58084 Hagen, Germany

ICMLA-09, Miami Beach, FL, 13.12.2009

Overview

- The role of passage reranking for question answering
- Characteristics of the passage reranking problem
- Using decision trees for probability-based ranking
- Splitting criteria for rank-optimizing decision trees
- Capturing the qualitative effect of features
- Learning ensembles of trees by stratified bagging
- Evaluation on factoid questions from QA@CLEF
- Conclusion

The LogAnswer QA System for German (www.loganswer.de)

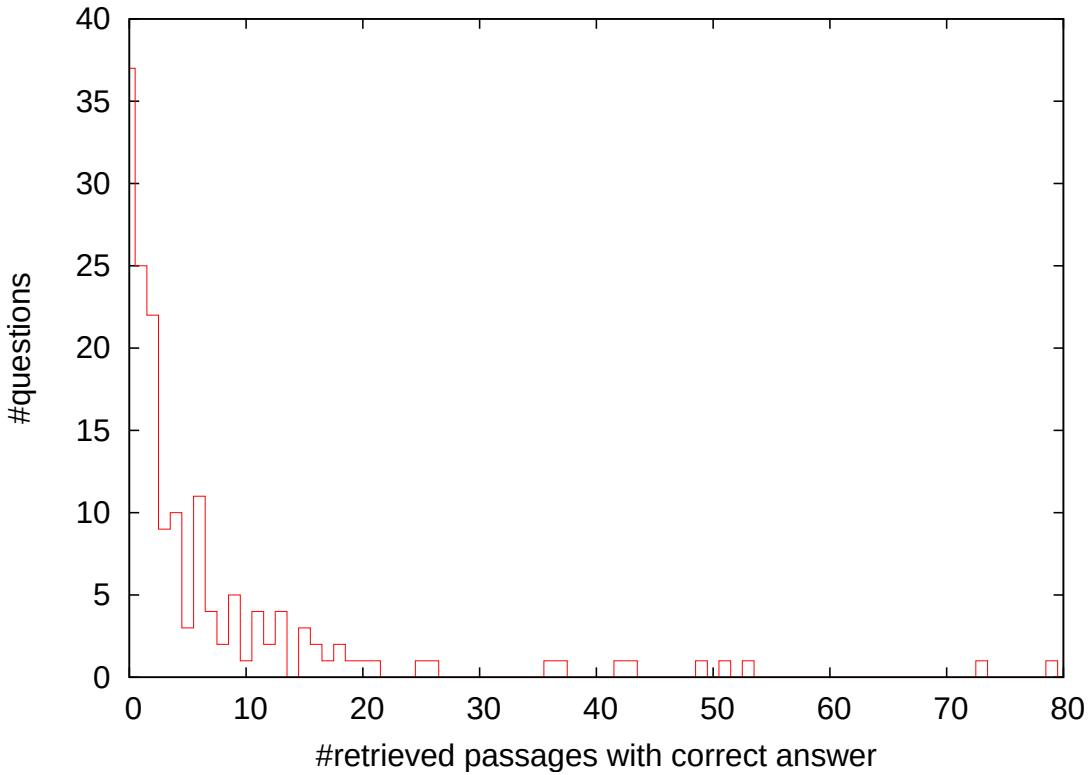


- Questions asking for a concrete fact (factoids), definition questions
- Answer phrases are highlighted in the snippets (usual search engines highlight occurrences of query terms)
- 200 passages retrieved, 3 shown: **bring best passages to front!**

Characteristics of the Passage Reranking Problem

Goal: Bring *k* best passages to front

Example data for 161 factoid questions from CLEF 2007:



- 31,091 retrieved passages (193 per question)
- Two-class annotation: 3.67% YES (contains answer), rest NO
- Most questions have **very few correct passages** (median: 2)
- Some **high-recall questions** (e.g. 79 correct passages)

Chosen Machine Learning Technique

Basic ML approach: Decision trees

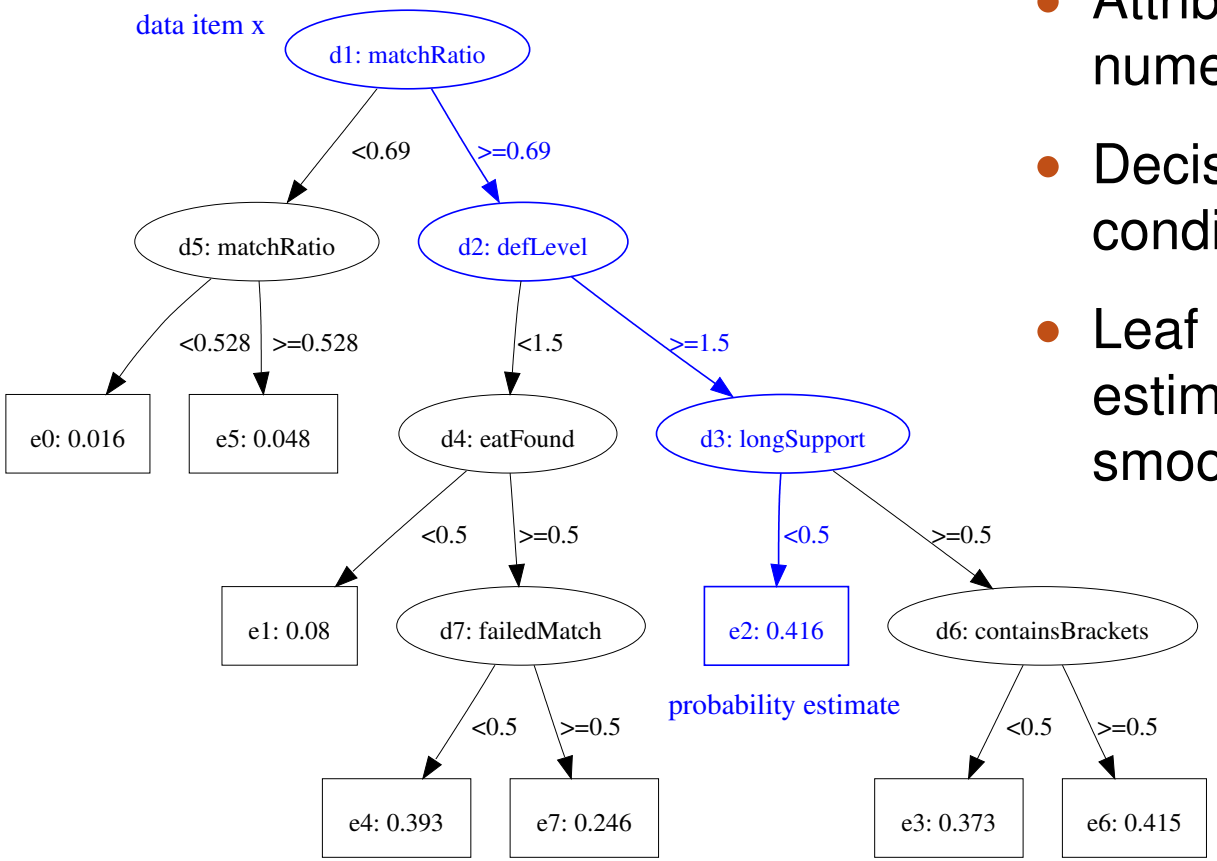
- \+ can handle multidimensional data with irrelevant and dependent/redundant attributes
- \+ ranking based on error probabilities obtained from training instances at the leaves
- \+ suited for very fast reranking of large candidate sets
- – First experiments: Standard tree induction techniques cannot cope with characteristics of passage reranking problem

Possible solution: Direct optimization of ranking quality

- Current trend in “learning to rank” approaches
- LambdaRank, ListNet, AdaRank...

⇒ *Apply this idea to decision trees, maintaining their key advantages*

A Probability Estimation Tree (PET)



- Attributes α assumed to be numeric
- Decision nodes d_i express conditions $x_\alpha \geq \theta$
- Leaf nodes e_j provide probability estimates p_j using Laplace smoothing, i.e.

$$p_j = \frac{|Y_j|+1}{|Y_j|+|N_j|+2}$$

with Y_j (N_j) training items at leaf j classified YES (NO), resp.

The $\mathrm{SC}_{\mathrm{kmrr}}$ Splitting Criterion

Starting point: Good ranking needed, only top-k results are of interest
Idea (Generalization of Mean Reciprocal Rank):

- Determine positions of the k correct results currently ranked best.
- Ideally, we should find the first correct result at rank 1, the second correct result at rank 2 etc.
- Penalize ranks that are worse than the ideal result.

$$\mathrm{SC}_{\mathrm{kmrr}}(\tau) = \frac{1}{|Q|} \sum_{q \in Q} \sum_{i=1}^{k_q^*} \frac{w_{i,k_q^*}}{\mathrm{rank}_{q,i}^{\tau} - i + 1}\,, \tag{1}$$

where $\mathrm{rank}_{q,i}^{\tau}$ is the rank of the ith best correct result for question q (with pessimistic tie breaking), $k_q^* = \min\{k, \#\text{yes items for } q\}$, and

$$w_{i,k_q^*} = \frac{2(k_q^* - i + 1)}{k_q^*\,(k_q^* + 1)}\,. \tag{2}$$

The $\mathrm{SC}_{\mathrm{kmap}}$ Splitting Criterion

Alternative: One can also start from mean average precision (MAP) and develop a variant that focuses on the k best correct results:

$$\mathrm{SC}_{\mathrm{kmap}}(\tau) = \frac{1}{|Q|} \sum_{q \in Q} \sum_{i=1}^{k_q^*} \frac{i \cdot w_{i,k_q^*}}{\mathrm{rank}_{q,i}^{\tau}} \,. \tag{3}$$

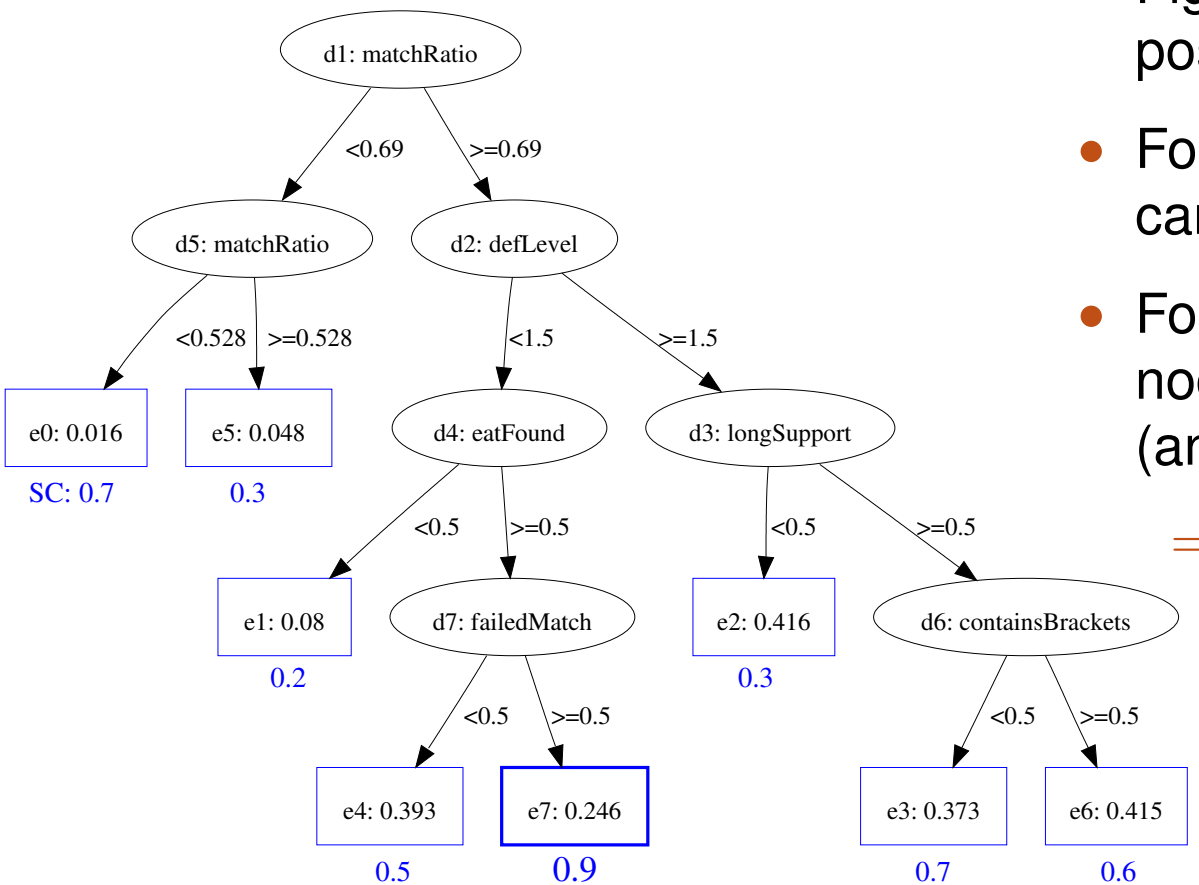
This criterion puts more weight on top ranked results with small i, so uniform weights $w_{i,k_q^*} = \frac{1}{k_q^*}$ are appropriate for $\mathrm{SC}_{\mathrm{kmap}}$.

Both criteria depend on the whole tree τ:

- $\mathrm{SC}_{\mathrm{kmrr}}$ and $\mathrm{SC}_{\mathrm{kmap}}$ cannot be expressed as local splitting criteria that depend only on the training items at the leaf to be split
- Splits have a global effect (can change the preferred splits in other nodes of the tree)

$\Rightarrow$ **Modification of tree induction procedure needed**

Global Search for the Best Split



- Figure shows best SC values of possible splits at each leaf
- For local splitting criteria, any node can be chosen for splitting
- For SC_{kmrr} and SC_{kmap}, splitting one node can change the SC scores (and best splits) of other nodes

  $\Rightarrow$ Global search for the best split over all leaf nodes of current tree
  (i.e. e7 is chosen for splitting)
- Can stop after *s* splits, no separate pruning needed

Capturing the Qualitative Effect of an Attribute

Some features have a **known positive/negative effect** on results:

More lexical overlap of question/snippet $\Rightarrow$ *better ranking*

Capturing this knowledge might improve the quality and consistency of the learned ranking.

Constraining decision tree learning to conforming models:

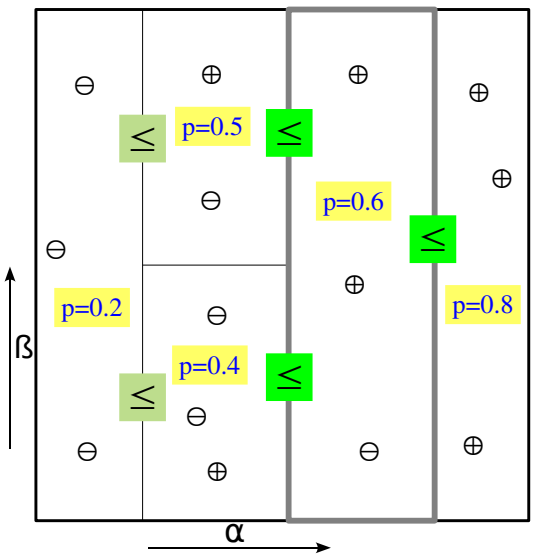
1. Declare some features as having a positive or a negative effect on probability estimates
2. Check the resulting inequalities that express these monotonicity requirements for each potential split

Remarks:

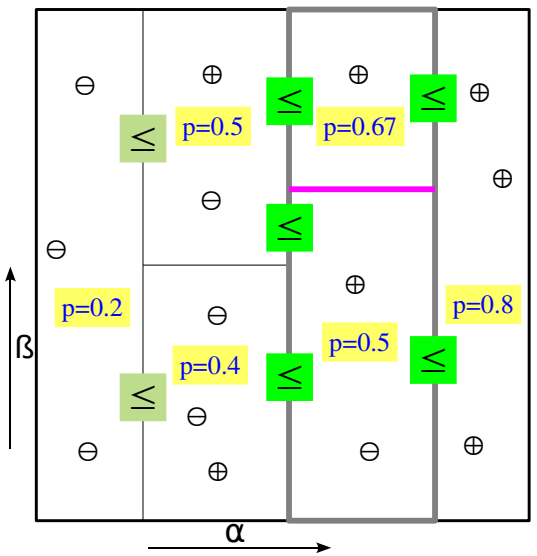
- Check can be restricted to neighboring leafs
- Splitting one leaf can affect the admissible splits of all other nodes

$\Rightarrow$ *Global search for best split is essential*

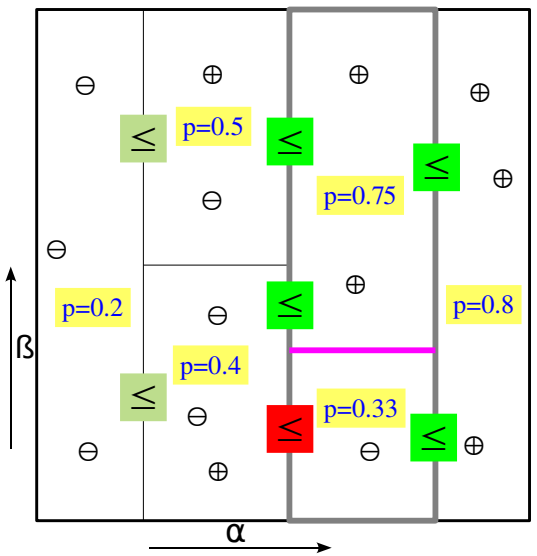
Enforcing Monotonicity Requirements



a) Starting point



b) Admissible split



c) Illegal split

- α declared as having a positive effect on results
- In order to verify that a split is admissible, only inequalities for neighboring cells must be checked

Bagging: Constructing Ensembles of Trees

Bagging (bootstrap aggregation; Breiman 1996) is known to improve the probability estimates of PETs (Provost/Domingos 2003).
Training: For $i = 1, \ldots, b$, where b is the desired number of trees in the ensemble:

1. (Resampling) Build a new training set X_i from the original training set X by randomly selecting $m = |X|$ training items from X (with replacement)
2. Apply the tree induction procedure to X_i in order to learn the ith tree τ_i of the ensemble.

Probability estimate of ensemble: arithmetic mean of probability estimates of learned trees $\tau_1, \ldots, \tau_b$

Why Passage Reranking Needs Stratified Bagging

Problem: In question answering, the number of positive candidates is typically very low compared to the number of negative cases.
$\Rightarrow$*Positive candidates can be lost in the resampling step*

Solution: Maintain the proportion of positive and negative examples in the test set by a separate resampling of positive and negative cases.

Let X be the training set, Y the set of training items in the YES class (answer passages) and $N = X \setminus Y$.
Stratified resampling:

1. Build a new set of positive training items Y_i by randomly selecting $y = |Y|$ training items from Y (with replacement);
2. Build a new set of negative training items N_i by randomly selecting $n = |N|$ training items from N (with replacement);
3. Join the resampled positive and negative examples into the new training set $X_i = Y_i \cup N_i$.

Evaluation: A Passage Reranking Experiment

Reranking of Retrieved Passages for the LogAnswer QA system

Training set: Passages for factoid questions from CLEF 2007 (as described earlier)

Test set: Passages for factoids from CLEF 2008:

- 127 factoid questions from QA@CLEF 2008
- 24,594 retrieved passages (ca. 194 per question)
- 832 passages with a correct answer (3.4%), median: 3 correct answers per question.
- For each question, there is at least one correct result (otherwise there can be no 'good' ranking)

See `http://www.loganswer.de/resources/icmla09_training.arff` (training set), `icmla09_test.arff` (test set)

Feature Set Used for Reranking Experiment

10 simple features:

failedMatch: #lexical concepts or numerals of question not found in passage

matchRatio: % lexical concepts or numerals of question that match passage

failedNames: #proper names of question not found in passage

containsBrackets: Presence of parentheses in the passage

eatKnown: Reports identification of expected answer type (EAT) of the question

testableEat: Signals if presence of the EAT in passages can be tested

eatFound: Reports an occurrence of the EAT in the passage

defLevel: Indicates a defining verb or apposition in a passage ($defLevel = 2$), a relative clause ($defLevel = 1$), or neither construction ($defLevel = 0$)

longSupport $= (snippetLength - 100)$ DIV 200, identifies overlong passages

irScore: The tf-idf based passage score determined by the retrieval module.

Positive effect on passage ranks: *matchRatio*, *irScore*
Negative effect: *failedMatch*, *failedNames*, *longSupport*

Description of Quality Metrics

Factoid questions can be fully answered by a *single* text passage containing the queried fact.
The position of the first correct answer in the result list for q, i.e. $\mathrm{rank}^{\tau}_{q,1}$, thus corresponds to the effort of a user searching for a correct answer in the result list.
The mean reciprocal rank averages the inverse of these ranks:

$$\mathrm{MRR} = \frac{1}{|Q|} \sum_{q \in Q, \mathrm{yes}_q > 0} \frac{1}{\mathrm{rank}^{\tau}_{q,1}}. \tag{4}$$

QA systems usually list only a few results, so the number of questions with at least one correct result at rank k or better is also important:

$$\mathrm{ANS}_k = \left| \{ q \in Q : \mathrm{yes}_q > 0, \mathrm{rank}^{\tau}_{q,1} \leq k \} \right| . \tag{5}$$

The test set contains 127 questions with at least one correct candidate passage, so ANS_k is bounded by 127 in the following.

Baseline/Standard tree learners (sorted by MRR)

model	MRR	ANS_1	ANS_2	ANS_3	ANS_4	ANS_5
RB	0.53	52	64	76	87	92
JBU	0.49	47	63	72	76	79
JB	0.48	44	62	71	77	82
RBU	0.47	43	59	67	74	78
IR	0.45	40	52	62	76	80
RTU	0.44	42	52	58	70	76
JTU	0.43	37	54	62	70	74
JT	0.42	40	48	58	65	70
RT	0.40	32	49	59	68	76

Retrieval baseline: IR is the original ranking determined by the retrieval system (no ML)

Naming of models: 'J' (Weka J48, Quinlan's C4.5 rev. 8), 'R' (Weka REPTree, Reduced Error Pruning); 'B' (Bagging of 10 trees), 'T' (single tree); 'U' (unpruned); see papers for more details

First result: *All single trees ('T') worse than IR baseline, bagging ('B') always better than baseline*

Results for Rank Optimizing Decision Trees

model	MRR	ANS_1	ANS_2	ANS_3	ANS_4	ANS_5
MB3/50	0.63	63	80	90	95	100
MB3/40	0.62	62	78	89	94	100
PB5/50	0.62	61	79	87	98	106
MB5/50	0.62	61	78	89	100	105
MB5/30	0.62	60	76	87	102	107
MB5/40	0.62	60	78	88	100	105
MB3/30	0.61	60	77	90	94	101
PB5/40	0.61	58	79	86	98	106
PB5/30	0.60	58	77	86	96	104
MB1/30	0.60	57	76	88	98	101
MB1/40	0.60	57	76	89	98	101
MB1/50	0.60	57	76	89	98	101
PB3/40	0.60	57	74	88	96	103
PB3/50	0.60	57	76	88	95	102
PB3/30	0.59	56	75	86	94	101

Naming scheme: 'M' ($\mathrm{SC}_{\mathrm{kmrr}}$), 'P' ($\mathrm{SC}_{\mathrm{kmap}}$), 'B' (stratified bagging of ten trees). k/s: tracked results/number of splits

Results: *better than baselines (84% questions answered); insensitive to parameters; small trees*

Ablation Results Compared to PB5/50

model	MRR	ANS_1	ANS_2	ANS_3	ANS_4	ANS_5
PB5/50	0.62	61	79	87	98	106
PB5/50S	0.60	59	72	87	96	100
PB5/50N	0.59	58	76	83	91	98
PT5/50	0.50	45	62	75	82	87

- PB5/50 chosen as the reference for ablation experiments
- PB5/50S uses ordinary bagging instead of stratified bagging
  $\Rightarrow$ *clear loss of* ANS_2 *and* ANS_5
- PB5/50N trained without monotonicity specifications
  $\Rightarrow$ *clear loss of* ANS_4 *and* ANS_5
- PT5/50 single learned tree (no bagging)
  $\Rightarrow$ *better than all trees obtained by standard tree induction*
  $\Rightarrow$ *only slightly above IR baseline.*
  This result once again confirms the importance of bagging when using decision trees for estimating probabilities.

Conclusions

- New method for learning probability estimating trees
  – Directly optimize a criterion for ranking quality
  – Consider known qualitative effect of attributes on the ranking
  – Combine several PETs by stratified bagging
- Experiments on factoid questions from QA@CLEF
  – Good results with very simple features
  – Usual PET learners outperformed by large margin
- Since only a few top-ranked results are of interest for QA, very small trees are sufficient
- Provost and Domingos' view that larger trees rank better does not seem to apply in this case
- Small size of the trees found by the new method suggests that they capture very stable abstractions