Finding Answer Passages with Rank Optimizing Decision Trees

Ingo Glöckner
Intelligent Information and Communication Systems Group (IICS)
University of Hagen
59084 Hagen, Germany
Email: ingo.gloeckner@fernuni-hagen.de

***Abstract*—The paper discusses the use of decision trees for probability-based ranking. Emphasis is placed on ranking problems in question answering, where the frequency of correct candidates is very low but a single correct answer at one of the top ranks is often sufficient. Since existing tree learners handle this task poorly, decision tree induction is reformulated in such a way that it directly optimizes a given measure of ranking quality (such as mean reciprocal rank or mean average precision). This change also makes it possible to incorporate a priori knowledge about the positive or negative effect of an attribute on ranking quality. Results are further improved by applying a stratified form of bagging. In a passage reranking task using factoid questions from the QA@CLEF evaluations, the new method outperforms existing tree induction techniques by a large margin.**

***Keywords*-learning to rank; question answering; focused retrieval; probability estimation tree; mean reciprocal rank**

I. Introduction

The paper reconsiders the issue of learning models for probability-based ranking of grouped data. The focus is on reranking retrieved passages in question answering (QA). This application is characterized by a low precision of the retrieval stage (often there is only one passage with an answer among hundreds of candidates), which can be problematic for machine learning techniques. Moreover, a single correct answer is usually sufficient. This is different from typical reranking problems in information retrieval. The ranking problem resembles answer selection [1], [2] but its goal is determining a ranked list of best results instead of only one answer. In order to facilitate the manual construction of a large training set, the method is supposed to work with a simple two-class annotation of the training data. This simplification is possible since the few correct candidates available for each question do not require further differentiation.

Decision trees were chosen as the basic learning technique since they can handle multi-dimensional data with irrelevant and dependent/redundant attributes. It is obvious how to use a decision tree as a probability estimation tree (PET) based on the relative frequency of the YES class in all training instances at a given leaf. Applying a learned PET involves minimal computational effort, which makes the tree-based approach particularly suited for a fast reranking of large candidate sets.

However, it is well-known that standard tree induction techniques (like C4.5) generate PETs that produce poor probability estimates [3]. The paper proposes several improvements of decision tree induction in order to make PETs better suited for ranking answer passages:

- The splitting criteria (like information gain) usually employed for tree induction are not sensitive to the grouping of the data by question. This results in poor performance for imbalanced training sets where a small number of high-recall questions contribute the majority of positive examples. In order to better handle this situation, it is proposed to directly use information retrieval measures as the splitting criterion. This is achieved by extending the PET learning technique.
- The paper introduces two classes of ranking measures, kMRR and kMAP, that generalize the mean reciprocal rank (MRR) and mean average precision (MAP).[1] While the ordinary MRR depends only on the best rank of any correct result, and MAP depends on the ranks of all correct results, the proposed kMRR and kMAP measures keep track of the k correct items that are currently ranked best. This makes these quality measures suitable for reranking problems where only a small number of top-ranked results are of interest.
- It is often known in advance which features have a positive (or negative) effect on result quality. For example, a higher degree of lexical overlap between question and candidate passage should express in a better ranking of the passage. However, common techniques for inducing decision trees cannot utilize this kind of a priori knowledge. In order to overcome this deficiency, it is shown how efficient support for corresponding 'monotonicity specifications' can be added to the proposed tree induction technique. Enforcing these monotonicity constraints can potentially improve the quality and consistency of the learned ranking.
- Bagging (bootstrap aggregation) [5] is known to improve the probability estimates determined by PETs [3]. Since in question answering, the number of positive candidates is typically very low compared to the number of negative cases, the use of *stratified bagging* is

[1] See [4] for a discussion of common measures of retrieval quality.

assumed in this paper. That is, the proportion of positive and negative examples in the test set are maintained by a separate resampling of the positive and negative cases in order to obtain more stable results.

Since the implementation of stratified bagging is obvious, the paper focuses on explaining the specifics of the proposed tree induction method. While most approaches to decision tree induction work strictly locally, considering one leaf at a time, the tree induction is formalized in such a way that all leaf nodes of the tree constructed so far are considered in parallel when determining the next split. This global approach is needed for ensuring monotonicity constraints and for supporting the desired rank-based splitting criteria. Experimental data for factoid questions from the QA@CLEF evaluations justify the proposed solution. The resulting models are already utilized in a QA system for the German Wikipedia that can be tested on the web.[2]

Related work

Learning to rank has received considerable interest in recent years, with one trend being the direct optimization of ranking quality. Important non tree-based approaches in that line are LambdaRank [6], ListNet [7] and AdaRank [8]. The problem has also been tackled by support vector machines [9]. Turning to tree-based methods, gradient boosting trees [10], [11] have been especially successful in learning to rank tasks. The present paper is not concerned with regression trees, however. Its focus is on ordinary decision trees that use the probability of the YES class at the leaves for ranking. In this setting, Zhang and Su [12] have already considered optimizing the AUC criterion (area under the ROC curve) in order to improve the quality of probability estimates determined by decision trees. However, the AUC criterion ignores the grouping of the data by questions that is characteristic of the passage reranking problem. Drummond and Holte [13] discuss the use of decision trees with imbalanced data, arguing that it is not the splitting criterion but rather the pruning method that should be made cost sensitive to handle such data. The presence of imbalanced data is only one aspect of the reranking problem tackled in this paper, though. The solution presented here does not rely on reweighting, but rather profits from the use of a rank-sensitive splitting criterion that always focuses on those correct results for each question that currently have the highest rank.

II. Proposed solution

A. Preliminaries

For simplicity, all attributes (except for the class assignment) are assumed to be numeric. For n attributes, each input datum is then given by an n-tuple $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. Let $X = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(R)}\} \subset \mathbb{R}^n$ be the set of training items. In the application, each $\mathbf{x}^{(r)} \in X$ describes a text passage $p^{(r)}$ that possibly answers the question $q^{(r)}$. To allow supervised learning, an annotator must identify the subset $Y \subseteq X$ of positive cases where the text passage $p^{(r)}$ actually answers question $q^{(r)}$. In general, $Y \subseteq X$ is the YES class of a two-class learning problem.

[2]see http://www.loganswer.de/ (for questions in German)

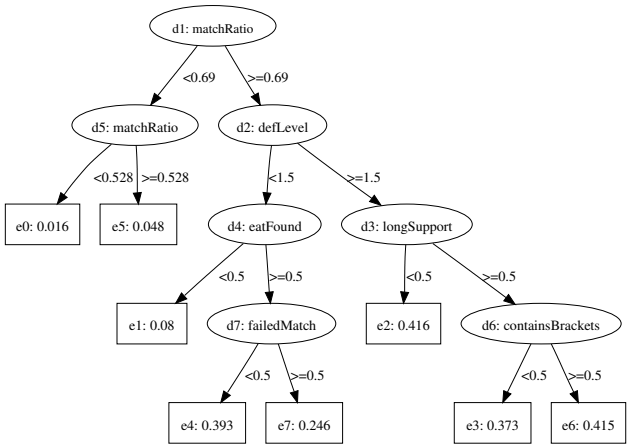

Figure 1. A probability estimation tree

B. Probability estimation trees

Since all attributes are numeric, binary trees with branching conditions of the form $x_\alpha \geq \theta$ are sufficient. A probability estimation tree (PET) is thus introduced as a binary tree τ with $s \geq 0$ inner nodes $D^\tau = \{d_1, \ldots, d_s\}$ (these nodes represent decisions) and leaf nodes $E^\tau = \{e_0, \ldots, e_s\}$ with $E^\tau \cap D^\tau = \varnothing$. Each inner node d_i, $i \in \{1, \ldots, s\}$, is labeled by an attribute $\alpha_i^\tau \in \{1, \ldots, n\}$, while each leaf node e_j, $j \in \{0, \ldots, s\}$, is labeled by a probability $p_j^\tau \in [0, 1]$. The arcs in A^τ correspond to conditions on the inputs. Since τ is a binary tree, every inner node d_i has exactly two children $c_i^\ell, c_i^r \in D^\tau \cup E^\tau$ connected to d_i by labeled arcs $d_1 \xrightarrow{<\theta_i^\tau} c_i^\ell$, $d_i \xrightarrow{\geq\theta_i^\tau} c_i^r$ for a threshold $\theta_i^\tau \in \mathbb{R}$. These arcs express the obvious condition, selecting either the left or right branch below d_i depending on the attribute value for $\alpha = \alpha_i^\tau$. Inputs with $x_\alpha < \theta$ are passed to the left child c_i^ℓ of d_i, while inputs with $x_\alpha \geq \theta$ are passed to the right child c_i^r of d_i. An example of a PET that illustrates these notions is shown in Fig. 1.

C. Determining probability estimates

By splitting inputs at each decision node until a leaf is reached, the PET partitions the input space into n-dimensional cartesian blocks

$$H_j^\tau = \prod_{\alpha=1}^{n} h(\ell_{j,\alpha}^\tau, u_{j,\alpha}^\tau) \tag{1}$$

where $\ell_{j,\alpha}^\tau \in \mathbb{R} \cup \{-\infty\}$, $u_{j,\alpha}^\tau \in \mathbb{R} \cup \{\infty\}$, $\ell_{j,\alpha}^\tau \leq u_{j,\alpha}^\tau$, and $h(\ell, u)$ is the half-open interval $h(\ell, u) = [\ell, u)$ for

$\ell \in \mathbb{R}$, while for $\ell = -\infty$, an open interval is obtained, i.e. $h(-\infty, u) = (-\infty, u)$. The partition $H^\tau = \{H_j^\tau : j = 0, \ldots, s\}$ of $\mathbb{R}^n$ determines a corresponding partition $X^\tau = \{X_j^\tau : j = 0, \ldots, s\}$ of the training instances,

$$X_j^\tau = X \cap H_j^\tau \tag{2}$$

for all $j \in \{0, \ldots, s\}$. Using the Laplace correction (add-one smoothing), the probability estimate p_j^τ associated with leaf e_j in the tree τ is calculated in the obvious way,

$$p_j^\tau = \frac{|X_j^\tau \cap Y| + 1}{|X_j^\tau| + 2} \tag{3}$$

where $Y \subseteq X$ is the set of all training items annotated as correct. This extends to a function $p^\tau : \mathbb{R}^n \to [0, 1]$ defined for arbitrary inputs $\mathbf{x} \in \mathbb{R}^n$, viz $p^\tau(\mathbf{x}) = p_j^\tau$, where j is the unique choice of $j \in \{0, \ldots, s\}$ with $\mathbf{x} \in H_j^\tau$. In the application, $p^\tau(\mathbf{x})$ estimates the probability that a text passage p described by $\mathbf{x}$ answers the considered question.

D. Global search for the best split

This section introduces a non-local method for inducing PETs that chooses the best split over all leaves of the tree constructed so far.

The tree induction starts with the singleton tree τ comprising a single leaf e_0, no inner nodes at all (i.e., $s = 0$) and no arcs, $A^\tau = \varnothing$. Since the initial node covers the full input space, the associated partitioning is $H^\tau = \{H_0^\tau\}$ with $H_0^\tau = \mathbb{R}^n$, i.e. $\ell_{0,j}^\tau = -\infty$, $u_{0,j}^\tau = \infty$ for all $j \in \{1, \ldots, n\}$. According to (3), e_0 is labeled by $p_0^\tau = \frac{|Y|+1}{|X|+2}$.

Now suppose that a tree τ with $s - 1$ decision nodes has already been constructed. Denote by $\tau' = \tau(j_s, \alpha_s, \theta_s)$ the tree with s inner nodes constructed by splitting leaf e_{j_s} of τ at attribute α_s with threshold θ_s. Formally, τ' has inner nodes $D^{\tau'} = D^\tau \cup \{d_s\}$ for some new node $d_s \notin D^\tau$ and leaf nodes $E^{\tau'} = E^\tau \cup \{e_s\}$ for a new node $e_s \notin D^{\tau'} \cup E^\tau$ (notice that e_{j_s} is kept in the set of leaf nodes, it now represents the left child of d_s, while e_s represents the right child of d_s). The set of arcs of τ' is given by $A^{\tau'} = \{z_1 \xrightarrow{\gamma} z_2 : z_1 \xrightarrow{\gamma} z_2 \in A^\tau, z_2 \neq e_{j_s}\} \cup \{z_1 \xrightarrow{\gamma} d_s : z_1 \xrightarrow{\gamma} e_{j_s} \in A^\tau\} \cup \{d_s \xrightarrow{<\theta_s} e_{j_s}, d_s \xrightarrow{\geq \theta_s} e_s\}$. The new decision node d_s is labeled by the attribute α_s. The remaining decision nodes in $D^{\tau'} \setminus \{d_s\} = D^\tau$ are labeled as in τ. In order to determine the labels $p_j^{\tau'}$ of the leaf nodes e_j of τ', consider the partition $\{H_j^{\tau'} : j = 0, \ldots, s\}$. If $j \neq j_s$ and $j \neq s$ (i.e. e_j is neither the left or right child of the node that was split), then $H_j^{\tau'} = H_j^\tau$. For the left child with index j_s, $H_{j_s}^{\tau'}$ is defined by (1), letting $\ell_{j_s}^{\tau'} = \ell_{j_s}^\tau$, $u_{j_s,\alpha_s}^{\tau'} = \theta_s$, and $u_{j_s,\alpha'}^{\tau'} = u_{j_s,\alpha'}^\tau$ for $\alpha' \neq \alpha_s$. For the right child with index s, $H_s^{\tau'}$ is defined by (1), letting $u_s^{\tau'} = u_s^\tau$, $\ell_{s,\alpha_s}^{\tau'} = \theta_s$ and $\ell_{s,\alpha'}^{\tau'} = \ell_{c_s,\alpha'}^\tau$ for $\alpha' \neq \alpha_s$. By (3), this determines the labels $p_j^{\tau'}$ for e_j, $j \in \{0, \ldots, s\}$.

In the following, a scoring function is assumed that controls the selection of the best split. That is, each PET τ is assigned an evaluation $\mathrm{SC}(\tau) \in \mathbb{R}$ (concrete choices are discussed below). Given such a scoring function, the gain in the score that results from a split can be expressed as

$$\Delta\mathrm{SC}(\tau; j_s, \alpha_s, \theta_s) = \mathrm{SC}(\tau(j_s, \alpha_s, \theta_s)) - \mathrm{SC}(\tau).$$

In each tree induction step, that split $(j_s, \alpha_s, \theta_s)$ is chosen for expanding the tree that maximizes $\Delta\mathrm{SC}(\tau; j, \alpha, \theta)$. The former tree τ with $s - 1$ decision nodes is then replaced by the new tree $\tau' = \tau(j_s, \alpha_s, \theta_s)$ with s decision nodes. Tree induction stops as soon as no admissible split with $\Delta\mathrm{SC}(\tau; j, \alpha, \theta) > 0$ exists anymore.

We further constrain the admissible splits (j, α, θ) used for determining the optimal $\Delta\mathrm{SC}(\tau; j_s, \alpha_s, \theta_s)$. Firstly, splits of pure leaf nodes are disallowed. So, if $X_j^\tau \cap Y = \varnothing$ or if $X_j^\tau \subseteq Y$, i.e. if X_j^τ contains only positive or only negative cases, then further splitting of leaf e_j is not possible.

Second, only a limited set of admissible thresholds $\Theta_{\mathrm{adm}}^\tau(j, \alpha)$ is considered. Given a possible split position $j \in \{0, \ldots, s - 1\}$ and split attribute $\alpha \in \{1, \ldots, n\}$, let $V = \{x_\alpha : \mathbf{x} \in X_j^\tau\}$ be the set of observed values of the instances in X_j^τ. The elements of V can be ordered such that $V = \{v_0, \ldots, v_m\}$ with $v_0 < v_1 < \cdots < v_m$. In principle, a set of possible thresholds is then given by

$$\Theta^\tau(j, \alpha) = \{(v_b + v_{b-1})/2 : b = 1, \ldots, m\}, \tag{4}$$

using the arithmetic mean of adjacent values for separating the instances. Additional conditions can be imposed in order to reduce $\Theta^\tau(j, \alpha)$ to a final set of admissible thresholds $\Theta_{\mathrm{adm}}^\tau(j, \alpha) \subseteq \Theta^\tau(j, \alpha)$. Specifically, a certain minimum leaf size $M \in \mathbb{N}$ is required, i.e. every $\theta \in \Theta_{\mathrm{adm}}^\tau(j, \alpha)$ must further satisfy that $\big|\{\mathbf{x} \in X_j^\tau : x_\alpha \geq \theta\}\big| \geq M$ and $\big|\{\mathbf{x} \in X_j^\tau : x_\alpha < \theta\}\big| \geq M$. In the experiments, a value of $M = 2$ was used.

E. Incorporating monotonicity constraints

The framework will now be extended by including support for monotonicity specifications. Hence let $\mathrm{MON} \subseteq \{1, \ldots, n\}$ be a set of attributes declared as showing a positive effect on the probability estimates.[3] We say that a PET τ respects the monotonicity requirements expressed by MON if for all $\alpha \in \mathrm{MON}$, $(x_1, \ldots, x_n) \in \mathbb{R}^n$ and $x'_\alpha \geq x_\alpha$, it holds that $p^\tau(x_1, \ldots, x_n) \leq p^\tau(x_1, \ldots, x_{\alpha-1}, x'_\alpha, x_{\alpha+1}, \ldots, x_n)$.

Obviously, it is sufficient for τ to respect MON if the condition holds for pairs of neighboring blocks H_j^τ of τ. In order to formalize this simplified monotonicity check, let e_j, $j \in \{0, \ldots, s\}$, be a given leaf of τ. The neighbors of e_j can be partitioned according to the contacting attribute.

[3] Attributes with a negative effect on probability estimates can be handled in analogy to the positive ones by reversing all inequalities.

The lower (upper) neighbors of e_j w.r.t. $\alpha \in \{1, \ldots, n\}$ are

$$L^\tau_{j,\alpha} = \big\{j' \in \{0, \ldots, s\} : u^\tau_{j',\alpha} = \ell^\tau_{j,\alpha} \text{ and } \forall \alpha' \neq \alpha, \max(\ell^\tau_{j,\alpha'}, \ell^\tau_{j',\alpha'}) < \min(u^\tau_{j,\alpha'}, u^\tau_{j',\alpha'})\big\}.$$
$$U^\tau_{j,\alpha} = \big\{j' \in \{0, \ldots, s\} : \ell^\tau_{j',\alpha} = u^\tau_{j,\alpha} \text{ and } \forall \alpha' \neq \alpha, \max(\ell^\tau_{j,\alpha'}, \ell^\tau_{j',\alpha'}) < \min(u^\tau_{j,\alpha'}, u^\tau_{j',\alpha'})\big\}.$$

We abbreviate $L^\tau_{j,\mathrm{MON}} = \bigcup\{L^\tau_{j,\alpha} : \alpha \in \mathrm{MON}\}$, $U^\tau_{j,\mathrm{MON}} = \bigcup\{U^\tau_{j,\alpha} : \alpha \in \mathrm{MON}\}$, and $N^\tau_{j,\mathrm{MON}} = L^\tau_{j,\mathrm{MON}} \cup U^\tau_{j,\mathrm{MON}}$ for all MON-neighbors of j.

A PET τ with s inner nodes respects the monotonicity requirements in MON if and only if for all $j \in \{0, \ldots, s\}$ and $j' \in U^\tau_{j,\mathrm{MON}}$, it holds that $p^\tau_j \leq p^\tau_{j'}$.

Knowing that an existing tree τ with $s-1$ inner nodes respects the monotonicity constraints in MON, the check that $\tau' = \tau(j, \alpha, \theta)$ also respects MON can be simplified. It is then sufficient to consider the children $c^\ell_s = e_j$ and $c^r_s = e_s$ of the newly added node d_s in τ' and verify that

$$p^{\tau'}_{j'} \leq p^{\tau'}_j \;\; \forall j' \in L^{\tau'}_{j,\mathrm{MON}}, \quad p^{\tau'}_j \leq p^{\tau'}_{j''} \;\; \forall j'' \in U^{\tau'}_{j,\mathrm{MON}}.$$

In this form, the criterion permits an efficient monotonicity check. It is used to further constrain the admissible thresholds $\theta \in \Theta^\tau_{\mathrm{adm}}(j, \alpha)$ used for finding the best split. Since PET induction starts with a singleton tree that trivially respects MON, this ensures that all trees constructed in later induction steps also respect the monotonicity requirements.

F. Group-sensitive learning

In the question answering context, the training data is naturally grouped by questions. In order to balance the effect of learning evenly among the questions from which the data was sampled, a group-sensitive scoring criterion is needed. The criterion should also reflect that the top-ranked results for each question are most important, since these are the results actually shown to the user. An appropriate criterion should keep track of the k highest-ranked correct results for each question, and encourage splits that improve the rank of these results.

The $\mathrm{SC}_{\mathrm{kmrr}}$ *splitting criterion*

In order to introduce a splitting criterion which accounts for these considerations, some more notation is needed. Hence let $Q = \{q^{(r)} : r = 1, \ldots, R\}$ be the set of all questions from which the training items were sampled. For a question $q \in Q$, the corresponding training items are given by $X(q) = \{\mathbf{x}^{(r)} \in X : q^{(r)} = q\}$, and the number of correct results for the question is $\mathrm{yes}_q = |X(q) \cap Y|$. Suppose we are interested in the k best results, where $k \in \mathbb{N} \setminus \{0\}$. If the number of correct results for the question is smaller than k, then it only makes sense to keep track of the correct answers that are actually available. Depending on the question, the following $\mathrm{SC}_{\mathrm{kmrr}}$ criterion therefore restricts attention to the $k^*_q = \min\{k, \mathrm{yes}_q\}$ correct results with the highest ranks:

$$\mathrm{SC}_{\mathrm{kmrr}}(\tau) = \frac{1}{|Q|} \sum_{q \in Q} \sum_{i=1}^{k^*_q} \frac{w_{i,k^*_q}}{\mathrm{rank}^\tau_{q,i} - i + 1}, \tag{5}$$

where $\mathrm{rank}^\tau_{q,i}$ is the rank of the ith best correct result for question q, and w_{i,k^*_q} is the weight associated with this ith best correct result given a total of k^*_q considered positive results for question q. In the $\mathrm{SC}_{\mathrm{kmrr}}$-based experiments presented below, weights with a linear decay were used:

$$w_{i,k^*_q} = \frac{2(k^*_q - i + 1)}{k^*_q\,(k^*_q + 1)}. \tag{6}$$

The ranks are obtained by sorting the training items $\mathbf{x} \in X(q)$ for each question in decreasing order of $p^\tau(\mathbf{x})$. If several items share the same probability score, a pessimistic ordering is assumed (i.e. all incorrect results come first, followed by the correct ones). Formally, let $P^\tau(q) = \{p^\tau(\mathbf{x}) : \mathbf{x} \in X(q)\}$ be the set of all probability scores of training items for question q. We can write $P^\tau(q) = \{p_1, \ldots, p_Z\}$ with $p_1 > p_2 \cdots > p_Z$. For $z \in \{1, \ldots, Z\}$, let

$$y_z = |\{\mathbf{x} \in X(q) \cap Y : p^\tau(\mathbf{x}) = p_z\}|$$
$$n_z = |\{\mathbf{x} \in X(q) \setminus Y : p^\tau(\mathbf{x}) = p_z\}|.$$

Now let $i \in \{1, \ldots, k^*_q\}$. Since there are at least k^*_q correct results for question q, there exists a unique $z^* \in \{1, \ldots, Z\}$ with $\sum_{z=1}^{z^*-1} y_z < i$ and $\sum_{z=1}^{z^*} y_z \geq i$. The rank of the ith best correct result for q given τ is $\mathrm{rank}^\tau_{q,i} = i + \sum_{z=1}^{z^*} n_z$.

The $\mathrm{SC}_{\mathrm{kmap}}$ *splitting criterion*

As an alternative to the $\mathrm{SC}_{\mathrm{kmrr}}$ criterion, which generalizes the mean reciprocal rank (MRR) of the best correct result for each question, one can also start from mean average precision (MAP) [4, p. 147] and develop a variant that focuses on the k best correct results. This results in the following criterion,

$$\mathrm{SC}_{\mathrm{kmap}}(\tau) = \frac{1}{|Q|} \sum_{q \in Q} \sum_{i=1}^{k^*_q} \frac{i \cdot w_{i,k^*_q}}{\mathrm{rank}^\tau_{q,i}}. \tag{7}$$

Note that $\mathrm{SC}_{\mathrm{kmap}}$ puts more weight on top ranked results with small i, therefore uniform weights $w_{i,k^*_q} = \frac{1}{k^*_q}$ were used for $\mathrm{SC}_{\mathrm{kmap}}$. The original MAP score is recovered for $k \geq \max\{\mathrm{yes}_q : q \in Q\}$, assuming a uniform weighting.

G. Computational considerations

A brute-force implementation of PET induction will compute the best split from scratch after each induction step. However, it is possible to avoid many redundant computations by reusing intermediate results computed in earlier steps. Hence let τ be a PET with $s-1$ inner nodes. For $j \in \{0, \ldots, s-1\}$, let

$$\Delta^\tau_j = \max\{\Delta\mathrm{SC}(\tau; j, \alpha, \theta) : \alpha \in \{1, \ldots, n\}, \theta \in \Theta^\tau_{\mathrm{adm}}(j, \alpha)\}.$$

Further let $\alpha_j^\tau \in \{1,\ldots,n\}$ and $\theta_j^\tau \in \Theta_{\mathrm{adm}}^\tau(j,\alpha_j^\tau)$ denote optimal choices of these parameters with $\Delta_j^\tau = \Delta\mathrm{SC}(\tau; j, \alpha_j^\tau, \theta_j^\tau)$. The best split can be determined by iterating over $j \in \{0,\ldots,s-1\}$, choosing j_s with $\Delta_{j_s}^\tau = \max\{\Delta_j^\tau : j = 0,\ldots,s-1\}$, and expanding the tree to $\tau' = \tau(j_s, \alpha_s, \theta_s)$ where $\alpha_s = \alpha_{j_s}^\tau$ and $\theta_s = \theta_{j_s}^\tau$. It is not necessary to compute the relevant data $(\Delta_j^\tau, \alpha_j^\tau, \theta_j^\tau)$ afresh in each induction step. Suppose the admissible thresholds in $\Theta_{\mathrm{adm}}^\tau(j,\alpha)$ depend on a minimum leaf size M and MON specification as described above. Then $\Theta_{\mathrm{adm}}^{\tau'}(j,\alpha) = \Theta_{\mathrm{adm}}^\tau(j,\alpha)$ for all $j \notin N_{j_s,\mathrm{MON}}^{\tau'} \cup N_{s,\mathrm{MON}}^{\tau'} = N_{j_s,\mathrm{MON}}^\tau \cup \{j_s, s\}$. Depending on the scoring function, this often results in $(\Delta_j^{\tau'}, \alpha_j^{\tau'}, \theta_j^{\tau'}) = (\Delta_j^\tau, \alpha_j^\tau, \theta_j^\tau)$, allowing reuse. For local splitting criteria like information gain, $(\Delta_j^{\tau'}, \alpha_j^{\tau'}, \theta_j^{\tau'})$ needs recomputing only if $j \in N_{j_s,\mathrm{MON}}^\tau \cup \{j_s, s\}$. For the group-sensitive $\mathrm{SC}_{\mathrm{kmrr}}$ or $\mathrm{SC}_{\mathrm{kmap}}$ metrics, however, the affected groups must also be taken into account. Thus, $(\Delta_j^{\tau'}, \alpha_j^{\tau'}, \theta_j^{\tau'})$ must be recomputed if either

(a) $j \in N_{j_s,\mathrm{MON}}^\tau \cup \{j_s, s\}$, or
(b) $\{q^{(r)} : x^{(r)} \in X_{j_s}^\tau\} \cap \{q^{(r)} : x^{(r)} \in X_j^\tau\} \neq \varnothing$.

On the other hand, if j is not a neighbor of j_s relative to the attributes in MON, and if the items in X_j^τ result from other questions than those in $X_{j_s}^\tau$, then the best split parameters $\alpha_j^{\tau'}$, $\theta_j^{\tau'}$ for leaf e_j and the corresponding gain $\Delta_j^{\tau'}$ remain the same in the new tree τ' obtained by splitting e_{j_s} in τ.

H. Controlling the tree size

The proposed method for inducing PETs by a global search for the best split generates a sequence of trees with increasing size such that the next tree in the sequence achieves the maximum increase of the quality score possible by splitting one leaf of the predecessor in the tree sequence.[4] Therefore one can simply stop the tree induction after any desired number of splits. Moreover, for small k, the method terminates automatically at very small tree sizes so that additional pruning is not mandatory.

III. Evaluation

The proposed technique has already been used for learning passage and answer ranking models: In the LogAnswer QA system, switching from usual decision trees to the $\mathrm{SC}_{\mathrm{kmrr}}$ method achieved a 50% increase of correct answers [14]; in the IRSAW system [15], there was a 30% increase.

The potential of the approach to learn useful rankings is now demonstrated by a passage reranking experiment.

A. Basic experimental setup

The retrieval subsystem of the IRSAW QA system [15] is used for retrieving passages. The passage size of IRSAW is currently limited to single sentences. A fixed number of 200 passages is retrieved for each question. The IRSAW retrieval component uses a bag-of-words retrieval model with tf-idf scoring. The ranking determined by the passage retrieval system forms the baseline for the subsequent reranking experiments.

[4]Local techniques like C4.5 do *not* find sequences with these properties.

B. Description of feature set

The following simple feature set was used as the basis for passage reranking (see [14]):

failedMatch: Number of lexical concepts and numerals in the question that do not match the candidate passage.
matchRatio: Proportion of lexical concepts and numerals in the question that find a match in the candidate passage.
failedNames: Number of proper names that occur in the question, but not in the passage.
containsBrackets: Presence of parentheses in the passage.
eatKnown: Signals if the system has successfully determined the expected answer type (EAT) of the question.
testableEat: Signals if an EAT test is possible for the passage (depending on answer type and parseability).
eatFound: Reports an occurrence of the EAT in the passage.
defLevel: Indicates a defining verb or apposition in a passage ($\mathit{defLevel} = 2$), a relative clause ($\mathit{defLevel} = 1$), or neither construction ($\mathit{defLevel} = 0$).
longSupport $= (\mathit{snippetLength} - 100)$ DIV 200, with DIV integer division, is used to penalize overlong passages.
irScore: The tf-idf based score of the retrieval component.

The *matchRatio* and *irScore* features were ascribed a positive effect on passage ranks; *failedMatch*, *failedNames* and *longSupport* were ascribed a negative effect.

C. Description of the training set

The 200 questions for German of the QA@CLEF 2007 evaluation [16] formed the starting point for building the training set. These questions contain pronouns and nominal anaphors that were resolved manually since coreference resolution is not of interest here. Only the 164 'factoids' in the test set (i.e., questions asking for a concrete fact) were considered since there are too few questions of other types for applying ML techniques. Three of the factoid questions were skipped due to parsing errors. The IRSAW retrieval system was applied to the remaining 161 questions. The resulting 31,091 passages were then annotated for containment of a correct answer.[5] In this annotation, 1,140 of the retrieved passages (3.67%) were judged correct. The passages that contain an answer are not evenly distributed across questions. For example, there is one question with 79 candidate passages that contain an answer, but a median of only 2 correct passages per question. The low number of correct candidates for most questions, combined with a few high-recall questions, demonstrates the need for an ML technique that can handle such imbalanced data.

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

Table I
BASELINE RESULTS (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

D. Description of the test set

The 160 factoid questions in the QA@CLEF 2008 test set for German were used as the starting point for building the test set for the experiment. Ten so-called NIL questions with no answer in the document collection were skipped as they are uninteresting for a reranking test. An automatic technique was used for resolving coreference, and 7 questions with wrong results of anaphor resolution or spelling errors were dropped. The remaining 143 questions were processed by IRSAW, resulting in 27,712 retrieved passages that were then annotated for containment of correct answers. There were 832 passages with a correct answer (3.0%) and a median of 3 correct answers per question. 16 questions with no correct result were omitted (in this case, every ranking is just a list of wrong results). The final test set contains 24,594 passages for the remaining 127 questions with at least one correct candidate.

E. Description of quality metrics

The factoid questions of the test set can be fully answered by a *single* text passage containing the queried fact. Therefore the position of the first correct answer in the result list for a question q, i.e. $\mathrm{rank}^{\tau}_{q,1}$, is of special importance, since it corresponds to the effort of a user who searches sequentially for a correct answer in the result list. The mean reciprocal rank takes the values of $\mathrm{rank}^{\tau}_{q,1}$ for all considered questions into account:

$$\mathrm{MRR} = \frac{1}{|Q|} \sum_{q \in Q, \mathrm{yes}_q > 0} \frac{1}{\mathrm{rank}^{\tau}_{q,1}}. \qquad (8)$$

In order to obtain deterministic results when several results share the same score, 'pessimistic' tie breaking (with all wrong results shown first) is assumed. Thus the actual results of the models are slightly better than the shown MRR scores with worst-case tie breaking. A QA system usually lists only a few results, so the number of questions with at least one correct result at rank k or better is also important:

$$\mathrm{ANS}_k = \left| \{ q \in Q : \mathrm{yes}_q > 0, \mathrm{rank}^{\tau}_{q,1} \leq k \} \right| . \qquad (9)$$

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

Table II
MAIN RESULTS (SORTED BY MRR)

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

F. Experimental results

The results of several baseline models are shown in Table I. The IR run corresponds to the original ranking of the retrieval system according to the *irScore*. The other models were obtained from decision tree inducers of the Weka toolkit [17]: JT is the result of the Weka J48 learner (a reimplementation of Quinlan's C4.5 revision 8) with Laplace smoothing. JB was obtained by bagging ten J48 trees with Laplace smoothing. JTU uses J48 with Laplace smoothing and no pruning (minimal leaf size $M = 5$); JBU results from bagging ten such trees. Alternatively, the Weka REPTree learner based on reduced error pruning was used: RT (REPTree with Laplace correction added after learning), RB (bag of ten such trees), RTU (REPTree with pruning switched off, minimum leaf size $M = 5$), and RBU (bag of ten such unpruned trees).

The table once again confirms that single PETs obtained from common tree induction methods are not suited for probability-based ranking: All trees achieve lower MRR than the original ranking of the IR system. But all learned bags outperform the results of the IR baseline. For example, RB has 12 more correct passages on top-1 rank than IR.

Results for the novel models using $\mathrm{SC}_{\mathrm{kmrr}}$ and $\mathrm{SC}_{\mathrm{kmap}}$ are shown in Table II. The following naming scheme is used: 'M' (splits based on $\mathrm{SC}_{\mathrm{kmrr}}$ with linear weighting), 'P' (splits based on $\mathrm{SC}_{\mathrm{kmap}}$ with uniform weighting), 'B' (stratified bagging of ten trees). Unless stated otherwise, the trees are trained such that the above monotonocity specifications are taken into account. The k/s notation specifies the k value for $\mathrm{SC}_{\mathrm{kmrr}}$ or $\mathrm{SC}_{\mathrm{kmap}}$ and the split limit s, i.e. the number of tree induction steps until training stops.[6]

Table II shows that each bag of PETs obtained from the proposed approach clearly outperforms all of the models in Table I obtained by standard tree induction techniques.

[6]Note that $\mathrm{SC}_{\mathrm{kmrr}}$ or $\mathrm{SC}_{\mathrm{kmap}}$ coincide for $k = 1$, so no PB1/x models are shown. The tree induction auto-terminates quickly after 30..70 steps. Split limits in the range 30..50 achieved the best performance.

Table III
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

The results obtained by using $\mathrm{SC}_{\mathrm{kmrr}}$ or $\mathrm{SC}_{\mathrm{kmap}}$ are rather similar and apparently not very sensitive to the number of tracked correct results k and the split limit s. Assuming that the top five passages are presented to the user, 84% of the test questions can be answered based on the MB5/30 model.

For ablation experiments, PB5/50 was chosen as the reference, since it has both good MRR and ANS_5 (i.e., many questions can be answered when showing a top five list). The relative effect of the proposed changes to ordinary decision tree induction on the quality of the ranking are shown in Table III. PB5/50S was obtained from ordinary instead of stratified bagging. This results in a clear loss of ANS_5. PB5/50N was obtained by ignoring the monotonicity specifications. The result is again an obvious loss of ANS_5 compared to PB5/50. Finally, PT5/50 is the result of a single learned tree (instead of ten trees combined by averaging). Although PT5/50 outperforms all of the single trees in Table I obtained by usual tree induction, it is only slightly better than the IR baseline. This confirms the importance of bagging if one uses decision trees for estimating probabilities.

IV. CONCLUSIONS

The method for learning probability estimating trees presented in the paper directly optimizes a criterion for ranking quality. It can also handle a priori knowledge on the qualitative effect of attributes on the ranking. Several PETs are combined by stratified bagging. Experiments on factoid questions from QA@CLEF show that the approach outperforms usual PET learning methods by a large margin. Since only a few top-ranked results are of interest for QA, very small trees are sufficient to yield this quality level. In particular, Provost and Domingos' view that larger trees rank better [3] does not seem to apply in this case. The compact size of the trees found by the new method suggests that they capture very stable abstractions.

ACKNOWLEDGMENT

Funding by the DFG (Deutsche Forschungsgemeinschaft) under contract HE 2847/10-1 is gratefully acknowledged.

REFERENCES

[1] V. Jijkoun and M. de Rijke, "Answer selection in a multi-stream open domain question answering system," in *Proc. ECIR'04*. Springer, 2004, pp. 99–111.

[2] J. Suzuki, Y. Sasaki, and E. Maeda, "SVM answer selection for open-domain question answering," in *Proc. COLING '02*. Morristown, NJ: ACL, 2002, pp. 1–7.

[3] F. Provost and P. Domingos, "Tree induction for probability-based ranking," *Machine Learning*, vol. 52, no. 3, pp. 199–215, 2003.

[4] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[5] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[6] C. J. Burges, R. Ragno, and Q. V. Le, "Learning to rank with nonsmooth cost functions," in *Proceedings of Neural Information Processing Systems (NIPS)*, 2006, pp. 395–402.

[7] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proc. ICML'07*. ACM, 2007, pp. 129–136.

[8] J. Xu and H. Li, "AdaRank: a boosting algorithm for information retrieval," in *Proc. SIGIR '07*, 2007, pp. 391–398.

[9] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proc. SIGIR'07*. ACM, 2007.

[10] K. Chen, R. Lu, C. K. Wong, G. Sun, L. Heck, and B. Tseng, "Trada: tree based ranking function adaptation," in *Proc. CIKM '08*. ACM, 2008, pp. 1143–1152.

[11] Z. Zheng, K. Chen, G. Sun, and H. Zha, "A regression framework for learning ranking functions using relative relevance judgments," in *Proc. SIGIR '07*. ACM, 2007, pp. 287–294.

[12] H. Zhang and J. Su, "Learning probabilistic decision trees for AUC," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 892–899, 2006.

[13] C. Drummond and R. C. Holte, "Exploiting the cost (in)sensitivity of decision tree splitting criteria," in *Proc. ICML '00*. Morgan Kaufmann, 2000, pp. 239–246.

[14] I. Glöckner and B. Pelzer, "Combining logic and machine learning for answering questions," in *Evaluating Systems for Multilingual and Multimodal Information Access: CLEF 2008, Revised Selected Papers*, C. Peters, T. Deselaers, N. Ferro, J. Gonzalo, G. Jones, M. Kurimo, T. Mandl, A. Peñas, and V. Petras, Eds. Springer, 2009, pp. 401–408.

[15] S. Hartrumpf, I. Glöckner, and J. Leveling, "Efficient question answering with question decomposition and multiple answer streams," in *Evaluating Systems for Multilingual and Multimodal Information Access: CLEF 2008, Revised Selected Papers*, C. Peters, T. Deselaers, N. Ferro, J. Gonzalo, G. Jones, M. Kurimo, T. Mandl, A. Peñas, and V. Petras, Eds. Springer, 2009, pp. 421–428.

[16] D. Giampiccolo, P. Forner, A. Peñas, C. Ayache, D. Cristea, V. Jijkoun, P. Osenova, P. Rocha, B. Sacaleanu, and R. Sutcliffe, "Overview of the CLEF 2007 multilingual question answering track," in *Working Notes for the CLEF 2007 Workshop*, Budapest, Hungary, 2007.

[17] I. H. Witten and E. Frank, *Data Mining. Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.