Merge-Tie-Judge: Low-Cost Preference Judgments with Ties

Kai Hui
Max Planck Institute for Informatics
Saarbrücken Graduate School of Computer Science
Saarbrücken, Germany
khui@mpi-inf.mpg.de

Klaus Berberich
htw saar
Max Planck Institute for Informatics
Saarbrücken, Germany
kberberi@mpi-inf.mpg.de

ABSTRACT

Preference judgments have been demonstrated to yield more accurate labels than graded judgments and also forego the need to define grades upfront. These benefits, however, come at the cost of a larger number of judgments that is required. Prior research, by exploiting the transitivity of preferences, successfully reduced the overall number of preference judgments required to $O(N_d \log N_d)$ for N_d documents, which is still prohibitive in practice. In this work, we reduce the overall number of preference judgments required by allowing for ties and exploiting that ties naturally cluster documents. Our novel judgment mechanism *Merge-Tie-Judge* exploits this “clustering effect” by automatically inferring preferences between documents from different clusters. Experiments on relevance judgments from the TREC Web Track show that the proposed mechanism requires fewer judgments.

ACM Reference format:
Kai Hui and Klaus Berberich. 2017. *Merge-Tie-Judge*: Low-Cost Preference Judgments with Ties. In *Proceedings of ICTIR ’17, October 1–4, 2017, Amsterdam, The Netherlands,* , 4 pages.
DOI: http://dx.doi.org/10.1145/3121050.3121095

1 INTRODUCTION

Offline evaluation in information retrieval aims at evaluating rankings of documents from rivaling systems based on a set of test queries. There exist two approaches to collect judgments, namely, graded judgments, where documents are labeled independently with a predefined grade, and preference judgments, where judges provide a relative ranking for a pair of documents. Preference judgments have been demonstrated to be a more robust mechanism than graded judgments [3, 6, 8]. Despite their advantages, they have seen little adoption in practice, and graded judgments remain prevalent. One reason for this is the inherently larger number of judgments required, given that preference judgments need to consider all pairs of documents. By assuming transitivity, this number is reduced to $O(N_d \log N_d)$ [1, 3, 9] for N_d documents, which is still impractical [7] for large N_d. More recently, we [4] demonstrated that the introduction of ties can dramatically reduce the number of judgments, when assuming transitivity. That is, when ties are included, different orders of judgments can lead to different numbers of judgments. Intuitively, ties cluster documents into a smaller number of tie partitions, where a single tie partition contains documents that are mutually tied. Ideally, one would desire to simultaneously merge the tied documents into several tie partitions, and to judge the preferences among these tie partitions instead of among documents. In this way, the number of judgments can be reduced from $O(N_d \log N_d)$ to $O(N_t \log N_t + N_d)$, where N_t is the number of tie partitions [4].

Inspired by this, we propose a novel labeling mechanism, coined *Merge-Tie-Judge*, to take advantage of the “clustering effect” of ties by dynamically merging documents into tie partitions in the judgment procedure. Akin to Song et al. [10], we assume that transitivity strictly holds, so that preferences can be inferred based on transitivity without incurring conflicts.

Contribution. A novel judgment mechanism is proposed to specifically consume preference judgments with ties, which is more robust than the existing method, and can further reduce the number of judgments required.

2 RELATED WORK

In early work, Rorvig [9] proposed that preference judgments could be used beyond the scaling-based judgments due to the applicability of simple scalability on documents. Since then, the advantages of preference judgments over graded judgments have been empirically tested and confirmed [3, 5, 8, 10]. However, the quadratic nature of the number of judgments required is overwhelming in practice [2]. Therefore, one important topic regarding preference judgments is to reduce the number of judgments. Assuming transitivity among preference judgments, the complexity is reduced from $O(N_d^2)$ to $O(N_d \log N_d)$ [1, 3, 10], by avoiding a full comparison among all document pairs. More recently, we [4] demonstrated that the introduction of ties can further dramatically reduce the number of judgments, when assuming transitivity. They analytically derived and empirically simulated the number of judgments required in preference judgments. They also argued that it is possible to further reduce the number of judgments by merging documents that are judged as ties simultaneously. Ideally, the number of judgments can be reduced from $O(N_d \log N_d)$ to $O(N_t \log N_t + N_d)$, where N_t is the number of tie partitions. Inspired by the theoretical analyses [4], we further propose *Merge-Tie-Judge* which implements this “cluster effects” in this work.

3 METHOD

3.1 Mechanism Framework

In the *Merge-Tie-Judge* mechanism, the “cluster effect” is leveraged explicitly. Intuitively, a cluster of documents grows via incoming tie judgments, by either adding a tied document to it or by merging it with other clusters. The follow-up judgments are made based on these merged clusters, the number of which keeps decreasing as more and more tie judgments are made.

```
Input All documents $\mathcal{D}$, document pairs $\mathcal{E}$, initialization of tie probabilities $\mathcal{P}^t$, hyper parameter JudNumber4SVM.
Output Sorted clusters of tied documents: $\mathcal{J}ud = \{jud_{ij} = c_i \succ c_j \text{ or } c_i \prec c_j | c_i, c_j \in C\}$, where $c_. = \{d_i | d_i \in \mathcal{D}$ that are tied$\}$.
/* $C_{\succ c}$ and $C_{\prec c}$: sets of clusters that are better
   or worse than $c$, initialized as empty.                    */
Initialization Clusters $C = \{c_i = \{d_i\} | d_i \in \mathcal{D}\}$, $TranTracker = \{(c_i, C_{\prec c}, C_{\succ c}) | c_i \in C\}$;
StopCondition $\forall c_i, c_j \in C, c_i \prec c_j$ or $c_i \succ c_j \in \mathcal{J}ud$
while not StopCondition do
    /* Select two clusters for judgment.                       */
    if $|\mathcal{J}ud| <$ JudNumber4SVM then
        $c_i, c_j, \mathcal{P}^t$ = ActiveSVM($C, \mathcal{J}ud$);
    else
        $c_i, c_j$ = SelectClusterPair4Judgment($C, \mathcal{P}^t$);
    end
    $jud_{ij}$ = ManualJudgment($c_i, c_j$);

    if $jud_{ij}$ is $c_i \sim c_j$ then
        /* Merge $c_i$, $c_j$ into new cluster $c_n$, and
           update $C$.                                          */
        $C$, $c_n$=MergeClusters($c_i$, $c_j$, $C$)
    end
    /* Update $\mathcal{J}ud$, adding incoming and inferred
       judgments.                                              */
    $\mathcal{J}ud$=UpdateTranTracker(TranTracker, $\mathcal{J}ud$, $jud_{ij}$)

    /* Update $\mathcal{P}^t$ for document pairs that are
       judged, 0 for non-tie and 1 for tie.                    */
    $\mathcal{P}^t$=UpdateTieProbability($\mathcal{P}^t$, $\mathcal{J}ud$)
end
```

Algorithm 1: *Merge-Tie-Judge*

The basic data structure here is the cluster of tied documents, which is initialized with a single document. When collecting judgments, two clusters are merged when at least one document pair from either side is judged as a tie; meanwhile the transitivity relationship for an individual cluster c is tracked with *TranTracker*, recording clusters that are judged better than ($C_{\succ c}$) or worse than it ($C_{\prec c}$). When a document pair is judged as a non-tie ($\prec$ or $\succ$), the transitivity property is applied over all clusters involved, and the *TranTracker* is updated accordingly. For example, if a new judgment indicates $c_i \succ c_j$, then we need to update *TranTracker* for c_i and c_j, as well as for clusters that are better than c_i, i.e., $c \in C_{\succ c_i}$, and that are worse than c_j, i.e., $c \in C_{\prec c_j}$, adding $jud_{ij} = c_i \succ c_j$ as well as the inferred pairwise judgments to $\mathcal{J}ud$, namely, $\{jud_{kl} = c_k \succ c_l | c_k \in C_{\succ c_i}, c_l \in C_{\prec c_j}\}$.

Another important data structure is $\mathcal{P}^t$, which tracks an estimate of the probability of being tied for every pair of clusters, and $\mathcal{P}^t_{ij} = P(c_i \sim c_j)$. During iterations, the next cluster pair to judge is selected according to $\mathcal{P}^t$: the cluster pair with the largest tie probability is chosen for judgment. It could be initialized randomly or based on prior knowledge, which is introduced in next section. After initialization, we keep updating $\mathcal{P}^t$ to reflect new judgments, and to compute tie probabilities among emerging clusters. In *UpdateTieProbability*, the tie probability between two emerging clusters is computed as the sum of the tie probability between old cluster pairs residing in either side. This aggregation represents the union of the involved document pairs are tied. Thus, the larger the clusters, the more likely they are picked out for judgment.

3.2 ActiveSVM for Tie Inference

As mentioned, the tie probability $\mathcal{P}^t$ introduced above can be initialized randomly. We argue that manual judgment procedure is special in the sense that it is too expensive to repeat dozens of times in practice. Thereby, the average number of judgments could be misleading, especially when the variance is large. In other words, though the average number of judgments is acceptable, the judgment mechanism may still result in an extreme large number of judgments in practice. Therefore, we propose to reduce the variance by introducing a predictor, i.e., *ActiveSVM*($C, \mathcal{J}ud$) in Algorithm 1. The prediction function will be triggered when positive *JudNumber4SVM* is set. The prediction can be cast as a supervised binary classification problem, where an ActiveSVM method is employed to initialize $\mathcal{P}^t$ to encode the prior knowledge.

Active Support Vector Machine. The support vector machine (SVM) is employed to make predictions. A document pair d_i, d_j is denoted as p_{ij}, meanwhile its corresponding label is $y_{ij} = 0$ or 1, corresponding to non-tie and tie. The classification problem aims at learning the function between the feature vector of each pair, i.e., $\Phi(p_{ij})$, and the binary label. Given that the ultimate target is to collect labels over document pairs with fewer manual judgments, the number of judgments for making prediction is desired to be small. Henceforth, the *ActiveSVM* with **Ratio Margin** strategy proposed by Tong and Koller [11] is used, which is designed to approach the optimized hyperplane with a small number of labeled data points. In each iteration, for each unlabeled document pair, two new classifiers w_+ and w_- are trained by hypothetically assigning the document pair a positive (tie) or a negative judgment (non-tie) respectively. The next pair to label is selected according to ratio between m_+ and m_-, by picking out $min(\frac{m_+}{m_-}, \frac{m_-}{m_+})$, where $m_.$ is the sum of margins of the classifier defined as follows.

$$m_+ = \sum_{p_{ij} \in \text{Support Vectors}} w_+ \Phi(p_{ij})$$

$$m_- = \sum_{p_{ij} \in \text{Support Vectors}} w_- \Phi(p_{ij})$$

4 EVALUATION

We now describe the experiments to examine the proposed *Merge-Tie-Judge* mechanism by comparing it with *Quick-Sort-Judge* [10]. To empirically compare the number of judgments from graded judgments and from preference judgments, we equivalently answer a question: "to approach the same ground truth generated with the graded judgments from TREC, how many judgments are required with preference judgments". We denote this number as *equivalent judgment number* (EquJn). As discussed in Section 3.2, we further examine the robustness of the proposed mechanism.

4.1 Evaluation Setting

Our experiments are based on 2011–2014 Trec Web Track's[1] queries and the corresponding labeled documents (qrel) for adhoc tasks, including 200 queries and 64k graded judgments. Only the judged documents are considered in this work. The judgments from Trec contain at most six relevance levels: junk pages (*Junk*), non-relevance (*NRel*), relevance (*Rel*), high relevance (*HRel*), key pages (*Key*) and navigational pages (*Nav*), corresponding to six graded levels, i.e., -2, 0, 1, 2, 3, 4. The concrete assignments varied from year to year, where *Junk* and *NRel* are always merged as *NRel* in this work, given the limited occurrences of *Junk* judgments (less than 5%). To employ the system rankings from rivaling systems as features in *ActiveSVM*, we also obtained the runs submitted by participants of the Trec Web Track. There are 62 runs from 2011, 48 runs from 2012, 61 runs from 2013, and 42 runs from 2014.

Collecting preference judgments. Ideally, we should rejudge the documents with preference judgments and compare them with the original graded judgments from Trec. Given the unaffordable number of document pairs to judge, instead, we employ the existing graded judgments from the Trec Web Track to create preference judgments in a straightforward manner. In particular, the preference judgments are created for two documents according to the comparison of their graded judgments, namely, if the label for d_1 is l_1 and the label for d_2 is l_2, the preference between these two documents is $d_1 \succ d_2$ when $l_1 > l_2$; $d_1 \sim d_2$ when $l_1 = l_2$ and $d_1 \prec d_2$ otherwise. In this way, after collecting judgments for all document pairs, we can simulate the same ground truth from graded judgments. Note that, in practice, it is unlikely to create exactly the same ground truth from judgments collected with different methods. This setting is mainly for allowing comparisons between preference and graded judgments under the same conditions, and also for guaranteeing the same amount of ranking information is collected by different competing mechanisms.

Competing mechanisms. The number of judgments with graded judgments is simply the total number of documents to judge, denoted as #Document. As a comparisons, we implement *Quick-Sort-Judge* (*QSJ*) from Song et al. [10]. The *Quick-Sort-Judge* is similar to a randomized *QuickSort* method, where, during each iteration, a document is randomly chosen as a pivot document. Thereafter, all remaining documents are grouped into worse than ($\prec$), better than ($\succ$) or tied with ($\sim$) per manual judgments. The mechanism terminates when all documents have been recursively sorted, and the results are based on 300 repetitions.

We examine the proposed *Merge-Tie-Judge* (*MTJ*), where ActiveSVM is used to initialize $\mathcal{P}^t$ as described in Section 3.2. We employ rankings from different systems in Trec as features to train ActiveSVM. In our preliminary experiments, the optimal setting of *JudNumber4SVM* varies a lot over different queries, as results of the difference of the quality of system rankings (features) and of the instinct difficulty to make prediction etc.. Therefore, we further demonstrate that the number of judgments over one year is robust within a range of settings for *JudNumber4SVM* $\in [5, 35]$. In addition, we also include the variants of *Merge-Tie-Judge* when initialized $\mathcal{P}^t$ randomly, namely, employing random probability in place of ActiveSVM, and denote it as *Merge-Tie-Judge-Random* (*MTJR*), whose results are also based on 300 repeats.

[1]http://trec.nist.gov/tracks.html

4.2 Equivalent Number of Judgments

In this section, we examine the *equivalent judgment number* from different competing mechanisms. The results for *Merge-Tie-Judge* and its comparison relative to *Quick-Sort-Judge* are summarized in Table 1. For *Merge-Tie-Judge*, we report the results under the average, best, and worst situations with different *JudNumber4SVM* in Table 1. We can see that the number of judgments is reduced significantly by 5.9%, 6.8% and 4.7% under average, best, and worst situations respectively, in comparison with *Quick-Sort-Judge*. Beyond that, the results from Merge-Tie-Judge, together with Merge-Tie-Judge-Random and *Quick-Sort-Judge* are visualized in Figure 1, where different mechanisms are reported in terms of mean, minimum, maximum, and the 95% confidence interval. We can see that there is no overlap of the 95% confidence interval from two variants of *Merge-Tie-Judge* and the one from *Quick-Sort-Judge*. Therefore, we can conclude that in terms of the number of judgments, both *Merge-Tie-Judge* and *Merge-Tie-Judge-Random* are significantly better than *Quick-Sort-Judge*.

4.3 Robustness

As mentioned, the robustness of the judgment mechanism is also important. Especially, one may desire, even in the worst case, the number of judgments from a mechanism still to be close to the number of judgments on average. From Figure 1, *Quick-Sort-Judge* fails to meet this expectation since the largest number of judgments could be multiple times larger. We report the coefficient of variation (Cv) to quantify the robustness in Table 1, which equals the ratio of the standard deviation to the mean. We can see that *Merge-Tie-Judge* is much more robust among a wide range of *JudNumber4SVM*, and the coefficient of variance is only 3.2% of the one from *Quick-Sort-Judge*. In addition, in Figure 1, from the distance between minimum and maximum number of judgments, as well as from the length of their confidence interval, it is obvious that both *Merge-Tie-Judge* are much more robust. Finally, compared with *Merge-Tie-Judge-Random*, *Merge-Tie-Judge* enjoy a better robustness with *ActiveSVM*, reducing the possible number of judgments into a small range.

5 DISCUSSION

The transitivity has been examined among weak preference judgments via crowdsourcing [5]. It has been demonstrated that transitivity holds for only 75% document triples. However, from Table 2 therein, it can be also seen that it is among tie judgments the transitivity does not hold, namely, only 32% of document triples are transitive. Whereas, for the remaining situations, the transitivity still holds for more than 90% of document triples. In other words, given $d_1 \succ d_2$, $d_2 \succ d_3$, no matter whether ties are allowed or not, one can always infer that $d_1 \succ d_3$ with transitivity. Similarly, given $d_1 \succ d_2$, $d_2 \sim d_3$, in more than 90% situations, one can infer that $d_1 \succ d_3$. However, given three document $d_1 \sim d_2$ and $d_2 \sim d_3$, one can not infer that $d_1 \sim d_3$, because they are not transitive anymore.

Table 1: *Equivalent judgment number* when using *Quick-Sort-Judge* (*QSJ*) and *Merge-Tie-Judge* with *ActiveSVM*. The statistics reported are based on *JudNumber4SVM* ∈ [5, 35]. Both absolute number of judgment (EquJn) and the relative comparison with document number (% more) are reported for average, best (minimum number of judgments) and worst (maximum number of judgments) situations. The number in the bracket is the relative reduction w.r.t. mean value in *QSJ*. The coefficient of variation (Cv) is reported in the rightmost column.

Year	Average EquJn	% more	Best EquJn	% more	Worst EquJn	% more	Cv	*QSJ*	% more	Cv	#Document
Wt11	23,818 (5.2%)	22.9%	23,680 (5.7%)	22.2%	23,977 (4.6%)	23.7%	0.0027	25,122	29.6%	0.159	19,381
Wt12	21,087 (6.7%)	31.3%	20,845 (7.7%)	29.8%	21,289 (5.8%)	32.6%	0.0046	22,587	40.7%	0.176	16,055
Wt13	19,557 (6.4%)	35.1%	19,365 (7.3%)	33.8%	19,887 (4.8%)	37.4%	0.0055	20,897	44.4%	0.167	14,474
Wt14	21,106 (5.5%)	46.3%	20,887 (6.5%)	44.8%	21,546 (3.5%)	49.3%	0.0083	22,331	54.8%	0.154	14,429
Summary	85,568 (5.9%)	33.0%	84,777 (6.8%)	31.8%	86,699 (4.7%)	34.8%	-	90,937	41.3%	-	64,339

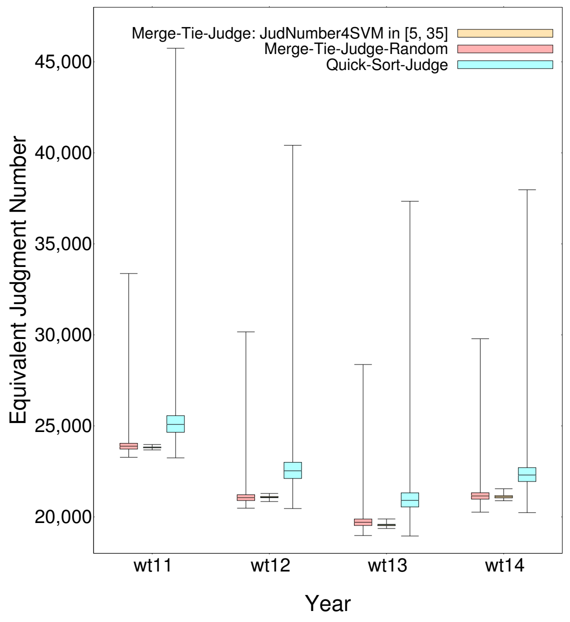

Figure 1: Comparison of number of judgments from different mechanisms. The x-axis is different years and y-axis represents the *equivalent judgment number*. The fewer *equivalent judgment number* the better. *Quick-Sort-Judge* is reported as a baseline; *Merge-Tie-Judge*, which implements *ActiveSVM*, and *Merge-Tie-Judge-Random* are both reported. For random mechanism, the mean, minimum, maximum as well as the 95% confidence interval are plotted.

As mentioned, this is due to the effects that it is hard to tell whether two documents should be judged as tied or non-tied, when their relevance are very close [5].

On the other hand, in *Merge-Tie-Judge*, non-ties are used to determine the relative order of tie partitions or of documents, meanwhile ties are used to grow the tie partitions. Given the degraded transitivity brought by ties, one can regard such tie partitions as documents, among which the mutual preferences are not of interest, and only the strict preferences among tie partitions are employed for evaluation. Recall that we [5] demonstrated the quality from weak preferences is significantly better than the one from graded judgments. Thus judgment mechanisms employing ties, like *Merge-Tie-Judge* and Quick-Sort-Judge [10], actually exchange 33% more judgments relative to graded judgments (Table 1) for a better judgment quality. Should one desire to collect strict preferences among all documents, which has the best judgment quality, at least 800% more judgments have to be made [4].

6 CONCLUSION

In this work, a robust judgment mechanism, named *Merge-Tie-Judge*, is proposed, utilizing the "cluster effect" of tie judgments, bringing down the number of judgments. For future work, in awareness of the importance of ties, they are worth being carefully defined and exploited to make them more applicable. Moreover, novel judgment mechanisms are desired, addressing the conflicts among preference judgments by tolerating intransitive judgments.

REFERENCES

[1] N. Ailon and M. Mohri. An efficient reduction of ranking to classification. *arXiv preprint arXiv:0710.2889*, 2007.

[2] M. Bashir, J. Anderton, J. Wu, P. B. Golbus, V. Pavlu, and J. A. Aslam. A document rating system for preference judgements. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 909–912. ACM, 2013.

[3] B. Carterette, P. N. Bennett, D. M. Chickering, and S. T. Dumais. Here or there: Preference Judgments for Relevance. In *Advances in Information Retrieval*, ECIR '08, pages 16–27. Springer, 2008.

[4] K. Hui and K. Berberich. Low-cost preference judgment via ties. In *Advances in Information Retrieval*, ECIR '17, pages 626–632. Springer, 2017.

[5] K. Hui and K. Berberich. Transitivity, time consumption, and quality of preference judgments in crowdsourcing. In *Advances in Information Retrieval*, ECIR '17, pages 239–251. Springer, 2017.

[6] G. Kazai, N. Craswell, E. Yilmaz, and S. Tahaghoghi. An analysis of systematic judging errors in information retrieval. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 105–114, New York, NY, USA, 2012. ACM.

[7] S. Niu, J. Guo, Y. Lan, and X. Cheng. Top-k learning to rank: labeling, ranking and evaluation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 751–760. ACM, 2012.

[8] K. Radinsky and N. Ailon. Ranking from pairs and triplets: Information quality, evaluation methods and query complexity. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 105–114, New York, NY, USA, 2011. ACM.

[9] M. E. Rorvig. The simple scalability of documents. *Journal of the American Society for Information Science*, 41(8):590, 1990.

[10] R. Song, Q. Guo, R. Zhang, G. Xin, J.-R. Wen, Y. Yu, and H.-W. Hon. Select-the-best-ones: A new way to judge relative relevance. *Information processing & management*, 47(1):37–52, 2011.

[11] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.