# Evaluating Information Retrieval Systems With Multiple Non-Expert Assessors

by

Le Li

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

**Abstract**

Many current test collections require the use of expert judgments during construction. The true label of each document is given by an expert assessor. However, the cost and effort associated with expert training and judging are typically quite high in the event where we have a high number of documents to judge. One way to address this issue is to have each document judged by multiple non-expert assessors at a lower expense. However, there are two key factors that can make this method difficult: the variability across assessors' judging abilities, and the aggregation of the noisy labels into one single consensus label. Much previous work has shown how to utilize this method to replace expert labels in the relevance evaluation. However, the effects of relevance judgment errors on the ranking system evaluation have been less explored.

This thesis mainly investigates how to best evaluate information retrieval systems with noisy labels, where no ground-truth labels are provided, and where each document may receive multiple noisy labels. Based on our simulation results on two datasets, we find that conservative assessors that tend to label incoming documents as non-relevant are preferable. And there are two important factors affect the overall conservativeness of the consensus labels: the assessor's conservativeness and the relevance standard. This important observation essentially provides a guideline on what kind of consensus algorithms or assessors are needed in order to preserve the high correlation with expert labels in ranking system evaluation. Also, we systematically investigate how to find the consensus labels for those documents with equal confidence to be either relevant or non-relevant. We investigate a content-based consensus algorithm which links the noisy labels with document content. We compare it against the state-of-art consensus algorithms, and find that, depending on the document collection, this content-based approach may help or hurt the performance.

**Acknowledgements**

I would like to thank my supervisor, Dr. Mark D. Smucker, for providing me numerous valuable suggestions during my 2-year study at School of Computer Science in University of Waterloo. I thank Prof.Gordon Cormack and Prof.Charlie Clarke for your precious time reading my thesis. I thank all my friends when I was in Singapore. Thank for your supports when I was down there. I thank all the friends I met during my internship in bay area. That 4-month internship refreshes my mind. Thank everyone in PLG lab. You guys are really amazing. Thank all the *Diaosi* I met in Waterloo. You guys will become *Gao-Shuai-Fu* someday in the future.

Thank Zhui. Without you, I couldn't have had such a wonderful journey. Wish you all the best.

## Dedication

To my mom, and brother.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 The Problem

Many current test collections are built based on the Cranfield paradigm (Cleverdon, 1967), which composes of three important factors: topics, documents and labels. Each topic contains multiple documents. Each document is associated with one label to indicate whether this document is relevant or non-relevant to the given topic. This label is from one trained experts so that it is trustable and can be used for different evaluation tasks. The ranking system evaluation cares about the order of documents while relevance evaluation doesn't. The qrel file is defined as a collection of tuples with the form (topic, document, and label) for the dataset.

However, the costs associated with the experts training and judging are very high, especially when we are faced with large number of documents needed to be judged. One popular way to relieve this issue is using pooling method. For example, the Text REtrieval Conference (TREC) (Voorhees et al., 2005), organized by National Institute of Standards and Technology (NIST), uses this method to build test collections. In TREC, each participated team need to submit the top-$k$ retrieved results for every topic based on their ranking algorithms, which is called a *run*. The aggregation of $n$ *runs* form a pool of candidate relevant documents. Those documents not in the pool are assumed to be non-relevant. Then the NIST assessors[1] only need to judge those documents in the pool. By doing so can we reduce the loads of NIST assessors, to some extent. However, to avoid the bias during

[1]The terms "NIST assessors" and "experts" are used interchangeably in this thesis since we run experiments on the datasets from TREC.

collection construction, we need to make sure we have sufficiently large pool relative to the collection size (Buckley et al., 2007). Hence, the pooling method still faces potential cost issue when we keep increasing the collection size.

Another viable solution is to have each document in the pool judged by multiple less-trained assessors at lower expense. Then for each document, we can apply a consensus algorithm to the relevance judgments to produce one single estimated label. Ideally, this single consensus label should reflect the majority opinion of those collected labels and be the same as the expert label. Accordingly, this method essentially tries to replace each expert's label with the consensus label. The reason why this method becomes more popular is due to the emerging of crowdsourcing. Crowdsourcing is a new practice of acquiring needed service from the public. Meanwhile, the online crowdsourcing platforms, like Amazon Mechanical Turk (AMT), make it possible that everyone can hire people who can access the Internet to get their requests done in a cheap and convenient manner. In information retrieval (IR) field, many studies (Alonso et al., 2008, Grady and Lease, 2010, Hosseini et al., 2012, Kazai and Milic-Frayling, 2009) have attempted to collect the documents' relevance judgments from the crowds.

However, there are two factors that can make this method difficult: the variability across the assessors, and the consensus algorithm.

Firstly, people are different by nature. For each document, it's almost impossible that every label from the less-trained assessor agrees with the expert label due to the diverse backgrounds of assessors. Some assessors may be more reliable than other assessors; some assessors may be spammers; some may be familiar with some topics presented but less knowledge in other topics. Even if all assessors are ethical, individuals might not agree on the criteria to define relevance. Actually, even though NIST assessors are trained, the variability across assessors still exist (Voorhees, 2000). We can foresee the potential disagreements among less-trained assessors with unknown accuracies. The corpus with inaccurate labels is definitely undesirable for evaluation.

Secondly, the diverse nature of assessors requires us rely more on those judgments from reliable assessors when we design the consensus algorithm. If the true labels are provided, each assessor's judging ability can be easily determined based on their performance on known documents (Kumar and Lease, 2011). However, the fact that we have zero ground-truth data needs the consensus algorithm is able to detect useful judgments so that the estimated consensus label is as close to the expert's label as possible. Hopefully, those noisy labels are comparable to, or even can replicate to, experts' labels in the evaluation. So, how to design a good consensus algorithm, given so many noisy observations and missing true labels, becomes another issue.

Many studies have tried to solve the above two issues, and use multiple non-expert assessors to replace the experts' labels in the evaluation of unranked retrieval sets. This type of evaluation mainly focuses on the accuracy of the consensus labels, not the order of documents presented. For example, we see a lot of studies discuss how to find high quality labels from crowdsourcing (Alonso and Mizzaro, 2009, Alonso et al., 2008, Hosseini et al., 2012, Jung and Lease, 2011, Kazai et al., 2012b). However, how we can replace the expert labels in the IR evaluation is less explored, especially very few studies discuss what kind of assessors are preferable if we want to achieve high correlation with experts in the IR evaluation.

## 1.2 Methodologies

This thesis uses a top-down manner to exam how can we replace the experts with multiple less-trained assessors in the ranking system evaluation.

In the very beginning, we describe the general way to find consensus for a single document as the procedure in Figure 1.1:

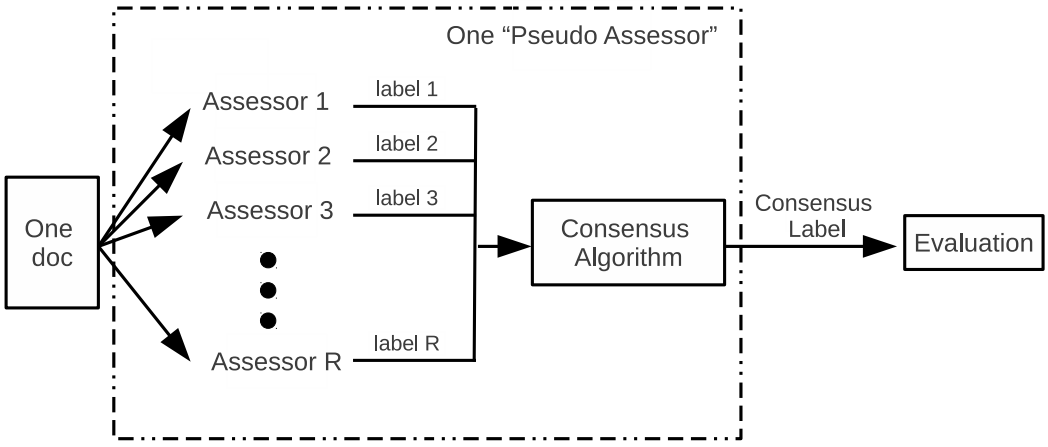



Figure 1.1: The general process of finding consensus for one document

1. One document is sent to $R$ ($R \geq 1$) assessors with different judging abilities for labeling. Let's denote the collected labels as $\mathbf{y} = \langle y^1, y^2, \ldots, y^j, \ldots, y^R \rangle$, where $y^j$ represents the label from $j^{th}$ assessor.

2. $\mathbf{y}$ are as inputs to a consensus algorithm which outputs a single label, $\tilde{y}$. $\tilde{y}$ is the estimated label for this document given a bunch of noisy labels.

3. We use this estimated label, $\tilde{y}$, as truth to evaluate other IR experiments.

Basically, the area surrounded by dotted square in Figure 1.1 acts like a black box or single "pseudo assessor" in the sense that it essentially generates only one label, $\tilde{y}$, for each incoming input document needed to be judge. So, if we think the judgments from NIST assessors as ground-truth data, then each NIST assessor can be treated as such "pseudo assessor" with best judging ability and zero mistakes, as $\tilde{y}$ is always equal to ground-truth label. The diverse nature of less-trained assessors, e.g. crowdsourcing assessors, makes it hard to ask this "pseudo assessor" to behave exactly like NIST assessor. So, our task is trying to make sure the labels from the pseudo assessor have similar performances like those from NIST assessor in different evaluation tasks. Basically, it is equivalent to the problem that investigates the effects of relevance judgments errors from one single assessor on IR evaluation. By knowing what kind of assessors we preferred, or what kind of errors we can tolerate, can provide better guidance on how to hire appropriate assessors or design desirable consensus algorithms. To fully investigate the effects of all types of relevance judgments' errors, we use signal detection theory (Macmillan and Creelman, 2004) to model assessors and simulate judgments.

Secondly, in the processing of designing good consensus algorithm, one important issue we may face is how to make a decision when we have equal confidence to believe one document to be either relevant or non-relevant. This tie situation often happens when we have even number of assessors to judge the same document in majority voting. Majority voting is commonly used as the baseline algorithm for its simplicity. It treats each judgment as an equally important vote and simply picks the class that gains more votes. The previous work does not have comprehensive guidelines on how to deal with ties. We would discuss how to break ties in decision-making process.

People are different in their judging abilities. So, we need to treat judgments different. The consensus decision should rely more on those judgments from reliable assessors. Then the problem becomes how to decide which assessors are more reliable than others, when we only have a few true labels, or even zero ground-truth data? Many studies use EM algorithm (Dawid and Skene, 1979) to iteratively estimate the qualities of assessors and their judgments. However, those studies only look at the noisy judgments. Raykar et al. (2010) proposes an EM algorithm that combines the contents and multiple judgments in medical application, which inspire us to investigate how many benefits we can get if we incorporate both document content and noisy judgments into the consensus-finding process.

## 1.3 Contributions

We make the following contributions in this thesis:

1. We investigate the effects of relevance judgment errors on the ranking system evaluation. Based on the simulation via signal detection theory, we find that we need to make sure the consensus labels are conservative, i.e. we prefer the consensus algorithms or assessors that tend to label documents as non-relevant. This observation provides a guideline to how to design a suitable consensus algorithm or how to hire appropriate assessors if we want to replace the expert labels in the ranking system evaluation. Meanwhile, we need to make sure the overall conservativeness of the consensus labels is at a reasonable level. Labels that are very conservative hurt the correlation with expert labels. (Chapter 3)

2. We find that the test collection with higher number of documents and more non-relevant documents tends to be more difficult to judge. Even though we ask same assessor to consistently judge two datasets with different statistics, worse correlation is achieved on the larger dataset with more portions of non-relevant documents. (Chapter 3)

3. We investigate two important factors that affect the conservativeness of the consensus labels: the assessor's criterion, and the relevance standard. We find that if one factor is too liberal (conservative), we can still boost the correlation by adjusting another factor to be conservative (liberal). (Chapter 3 and 4)

4. We propose and investigate five strategies to decide the consensus labels for the documents in a tie in majority voting. (Chapter 4)

5. We investigate a content-based consensus algorithm that combines the noisy relevance judgments and the document content. Based on the comparison results against other consensus algorithms on two datasets (2005 TREC (Voorhees, 2005) Robust Track, referred as Robust2005, and Trec8 (Voorhees and Harman, 1999) ad-hoc Retrieval Track, referred as Trec8), we find that the document content may or may not help the performance. In the balanced dataset (Robust2005), we see the benefits of utilizing document contents in relevance evaluation. However, in the imbalanced dataset (Trec8), where non-relevant documents dominate the dataset, the content hurts the relevance evaluation.

Based on the IR evaluation results on two datasets, we find that the EM algorithm in Raykar et al. (2010), which doesn't need document contents, can be used to replace expert labels in IR evaluation, when it runs sufficient number of iterations, and each document is judged by at least 8 assessors. The content-based consensus algorithm can achieve the best correlation when we have a few number of assessors on Robust2005. Without proper methods to handle the imbalanced dataset (Trec8), content hurts the IR evaluation. (Chapter 5)

In sum, Chapter 2 mentions the related work on the ranking system evaluation and crowdsourcing. The background knowledge of signal detection theory, which is used for judgment simulation, is also discussed. Chapter 3 investigates the effects of relevance judgment errors on ranking system evaluation. In Chapter 4, five strategies are proposed to break ties. We also investigate how the threshold of relevance affects the IR evaluation. After that, we investigate the content-based consensus algorithm and compare it against others algorithms in the Chapter 5. Finally, this thesis ends with the summary and future direction.

# Chapter 2

# Related Work and Background Knowledge

## 2.1 The Assessors Errors in IR evaluation

The current test collections, e.g. TREC, are mainly built based on the Cranfield paradigm. To avoid the exhaustive judging, TREC relies on pooling method to form a candidate set of relevant documents for the expertise to judge the relevance. The candidate sets are from $n$ independent ranking systems, where each ranking system only submits the top-$k$ relevant documents which it believes to be relevant. The unselected documents are labeled as non-relevant by default. The reason why it works is because it follows the probability ranking principle (Robertson, 1977). However, it's very expensive to hire and train expert assessors when we are facing a large amount of documents. Various work (Aslam et al., 2006, Cormack et al., 1998, Soboroff et al., 2001, Webber et al., 2012, Zobel, 1998) have been proposed to investigate the features of pooling method.

We mainly focus on the how the assessors' errors affect the IR evaluation. In Voorhees (2000), the authors found out that even though there exist high disagreements between the relevance judgments among the NIST assessors, the agreement in terms of the IR evaluation is very high. Similar observation can be found in Carterette and Soboroff (2010), Cormack et al. (1998). This indicates that we can tolerate the errors made by assessors in ranking system evaluation, to some extent. This discovery inspires many studies that try to improve the NIST assessors efficiency with less efforts and costs in IR evaluation. For example, even though Soboroff et al. (2001) assigned relevance to documents in the pool randomly, we can still achieve significant high correlation with NIST assessors. However, the retrieval

system is not always robust to the assessor errors for IR evaluation. For example, Aslam et al. (2006) shows the system ranking is greatly affected the assessors' errors when shallow pools are given. And in Bailey et al. (2008), Kinney et al. (2008), the levels of assessors' expertise also affect the ranking system evaluation if domain knowledge is required.

From Smucker and Jethani (2011a) we know those crowds assessors have worse judging ability than NIST assessors. However, we are still not sure the extent to which we can use the judgments from assessors with various judging abilities to supplement, or even replace, those from NIST assessors in IR evaluation. This gives us more impetus on this topic.

## 2.2 Crowdsourcing

Crowdsourcing brings both opportunities and challenges to IR research. Even though it offers fast and low-cost way to collect the judgments, the diverse nature of crowdsourcing workers adds uncertain or even risky factors to the experiment results. This diversity spans from various aspects like the incentives, knowledge bases, demographics and etc. Essentially, how to maximum the quality of labels collected from the crowds while eliminating the efforts associated with experiment becomes the centric question we need to resolve. This section shows the related work on crowdsourcing. We divide the related work into following four categories:

- quality control
- how to model crowdsourcing users
- consensus algorithm
- evaluation

The purpose of quality control is to assure the quality of labels when designing and conducting the experiment that collects the labels from crowds. Designing a good crowdsourcing experiment needs to consider multiple factors that affect the outputs. For example, in Eickhoff and de Vries (2012), Kazai (2010), Kazai et al. (2011a,b, 2012b), the authors show how the crowdsourcing experiment settings (like task type and user interface) and human factors(like the payment and required efforts) change the label accuracy. As to quality control techniques, a commonly-used strategy is to make use of documents with ground-truth labels, or gold standard data. Basically we can measure the quality of assessors based on how well they did in those gold standard data and then detect the types

of spammers (Vuurens et al., 2011), filter out spam assessors in the qualification test or weight each assessor's label (Le et al., 2010, Oleson et al., 2011). Moreover, Alonso (2012), Kazai and Milic-Frayling (2009), McCreadie et al. (2010), Nowak and Rüger (2010) also mention how to assure the quality based on the inter-annotator agreement.

Understanding the user behaviors of crowdsourcing assessors is vital since it can help us understand whether a task is suitable for crowdsourcing, and what kind of assessors are preferred. User behavior analysis is a broad area which can span a lot of topics, like demographics (Ipeirotis, 2010a,b, Ross et al., 2010) or the behavioral consistency across time (Blanco et al., 2011). This thesis is especially interested in how to model crowdsourcing assessors' judging abilities in the sense that we can use these models to determine the quality of labels as well as assessors. The most straightforward way is using different evaluation metrics, like accuracy, to determine quality of assessors. For example, in Kumar and Lease (2011), the authors use the assessors' accuracies on those documents with known relevance to model the quality of assessors. The work in Raykar et al. (2010) mentions a way of using sensitivity (True Positive Rate) and specificity (1-False Negative Rate) to model the probability that an assessor makes mistakes. Besides those models by evaluation metrics, the authors in Smucker and Jethani (2011b) propose using signal detection theory to calculate the discriminative ability and criterion of crowdsourcing assessors. The comparison between lab participants and crowdsourcing assessors on relevance evaluation shows that crowdsourcing assessors obtain less discriminative ability and are more liberal. Carterette and Soboroff (2010) captures the crowdsourcing assessors' errors and comes to the conclusion that those assessors who tend to label incoming documents to be non-relevant are more preferable in terms of the ranking system evaluation.

Consensus algorithm is the attempt of finding the single label that represents the most assessors' opinion towards on one document. Meanwhile, we need to make sure this consensus label is close to the expert label. If we treat a consensus algorithm as a black box, the inputs are one document and the noisy labels associated with this document, while the output is just one single consensus label. Basically, consensus algorithm tries to reduce the gap between ground-truth labels and consensus labels. The easiest way is called majority voting, which treats each label as equally important vote. If the number of votes is beyond the threshold of majority, this algorithm treats this document to be relevant. This algorithm is commonly used as baseline algorithm. One drawback is that it ignores the fact that people are different. Unlike majority voting, EM algorithm(Dawid and Skene, 1979) captures the differences among assessors and iteratively adjusts the weights for the labels based on the estimated assessors' error rates. This algorithm is widely adopted and proven to have better performance than majority voting in many studies (Callison-Burch, 2009, Hosseini et al., 2012, Ipeirotis et al., 2010, Kumar and Lease, 2011, Li et al., 2011, Raykar

et al., 2010, Tang and Lease, 2011, Yan et al., 2011). In a similar vein, Jung and Lease (2011), Smucker (2012) determines annotator's quality from the angle of signal detection measurement. This theory firstly converts each annotator's True Positive Rate (TPR) and False Positive Rate (FPR) into $z$ score (Macmillan and Creelman, 2004) and then uses the difference between these two $z$ scores to measure how good one annotator can discriminate between relevant and non-relevant documents.

One of the EM algorithms presented in Raykar et al. (2010) catches our attention in the way that it utilizes both content and noisy judgments to find the consensus. However, one limitation of their algorithm is that they cannot properly handle the situation when the dimension of feature vector increases, since they need to solve the linear system dependent on the dimension of features, which constrains the generalization ability of the algorithm.

We focus on two types of evaluation in IR: relevance evaluation and ranking system evaluation. The difference lies in whether we need to consider the order of documents or not. Lots of existing work (Alonso and Baeza-Yates, 2011, Alonso and Mizzaro, 2009, Alonso et al., 2008, Grady and Lease, 2010, Jung and Lease, 2012, Kazai et al., 2011a) in crowdsourcing are focused on the relevance evaluation. These studies investigate the techniques, methodologies and factors that enable us to replace NIST qrels with crowdsourcing judgments. As to the ranking system evaluation, some studies show how those ranking systems from different groups of assessors are correlated with each other, for example, assessors with good discriminative ability vs. those are not (Bailey et al., 2008), assessors with domain knowledge vs. those are not (Kinney et al., 2008), liberal assessors vs. conservative (Carterette and Soboroff, 2010). Some studies explore how the crowdsourcing experiment designs, like UI (Kazai et al., 2011a) or cost allocation for each judgment (Hosseini et al., 2011a,b), affect the IR evaluation. The authors in Kazai et al. (2012a) show that systematic differences in labeling lead to differences in assessors' training and evaluation. In Hosseini et al. (2012) indicates that the consensus label from EM algorithm achieves better performance than those from MV on ranking system evaluation. However, the extent to which we are able to replace the experts with crowdsourcing judgments in IR evaluation is less explored in existing work.

## 2.3 Signal Detection Theory and Dataset Simulation

### 2.3.1 Introduction

Signal detection theory (Macmillan and Creelman, 2004) is used in this thesis to model the judging ability of each assessor. One important advantage of signal detection theory

is that it can model the uncertainty in the decision-making process.

Based on this theory, we can use two normal distributions to model one assessor's judging ability. As shown in Figure 2.1, the distribution on the right captures the assessor's stimulus to relevant documents while the left distribution is for non-relevant documents. Given these two distributions, we can model this assessor by two variables: $d'$ and $c$. $d'$ captures how difficult one annotator can discriminate between the relevant and the non-relevant documents. The larger $d'$ is, the farther apart the two distributions are, and the better discriminative ability this annotator obtains. The criterion $c$ indicates the threshold of relevance of the assessor. The assessor labels the document as relevant if the relevance is larger than the criterion; otherwise non-relevant.

If one annotator tends to label incoming documents to be relevant to avoid missing relevant documents (but at the risk of high false positive rate), then this annotator is liberal with a negative criterion. If $c = 0$, the annotator is neutral. A conservative annotator has a positive criterion. As we can see, one advantage of using this model is that the measurement of assessor's ability to discriminate is independent of each assessor's criterion.

Given the model of the signal detection theory, we can model the discriminative ability and criterion as:

$$d' = z(TPR) - z(FPR) \tag{2.1}$$

$$c = -\frac{1}{2}(z(TPR) + z(FPR)) \tag{2.2}$$

Where $TPR$ and $FPR$ are true positive rate and false positive rate of this assessor, respectively. Function $z$ is the inverse of the normal distribution function. Figure 2.2 shows example $d'$ curves.

### 2.3.2 How To Simulate Judgments[1]

If one assessor's $d'$ and $c$ are given, we can use them to calculate the TPR and FPR of this assessor using Equations 2.3 and 2.4, which are derived from Equations 2.1 and 2.2. Assuming one document's true label is given, essentially, we generate the simulated judgment by tossing a biased coin that the degree of bias is related with the probability that

[1]The simulation method and software described in this section was created by Mark Smucker and used with permission.

this assessor make mistake. To be more precise, if the true label is 1, then this assessor flips the coin with bias TPR; if the true label is 0, the bias is FPR. In dataset from TREC, the true labels can be found in the official qrel file.

Moreover, it's reasonable to assume that people differ in their judging abilities and behaviors. So, if one document is assessed by more than one assessor, we should model this difference by drawing from normal distributions with a given mean and standard deviation. Algorithm 1 shows the details how the simulated judgments are generated.

$$TPR = CDF(d' * 0.5 - c) \tag{2.3}$$

$$FPR = CDF(-d' * 0.5 - c) \tag{2.4}$$

Where CDF means Cumulative Density Function.

**Algorithm 1** Simulate The Judgments From One Assessor

**INPUT**: $mean\_d', SD\_d', mean\_c, SD\_c, trueLabels$
$d' \leftarrow$ random number from $\mathcal{N}(mean\_d', SD\_d')$ ▷ $\mathcal{N}(\mu, \sigma)$: normal distribution
$c\prime \leftarrow$ random number from $\mathcal{N}(mean\_c, SD\_c)$
$TPR \leftarrow CDF(d' * 0.5 - c)$ ▷ Cumulative Distribution Function
$FPR \leftarrow CDF(-d' * 0.5 - c)$

**for** $i = 1$: size($trueLabels$) **do**
  $judge_i \leftarrow trueLabels_i$
  $flip \leftarrow rand(0, 1)$ ▷ a random number in $[0, 1)$
  **if** $trueLabel_i == 0$ **then**
    **if** $flip \leq FPR$ **then**
      $judge_i \leftarrow 1$
    **end if**
  **else**
    **if** $flip > TPR$ **then**
      $judge_i \leftarrow 0$
    **end if**
  **end if**
**end for**
**RETURN** $judge$

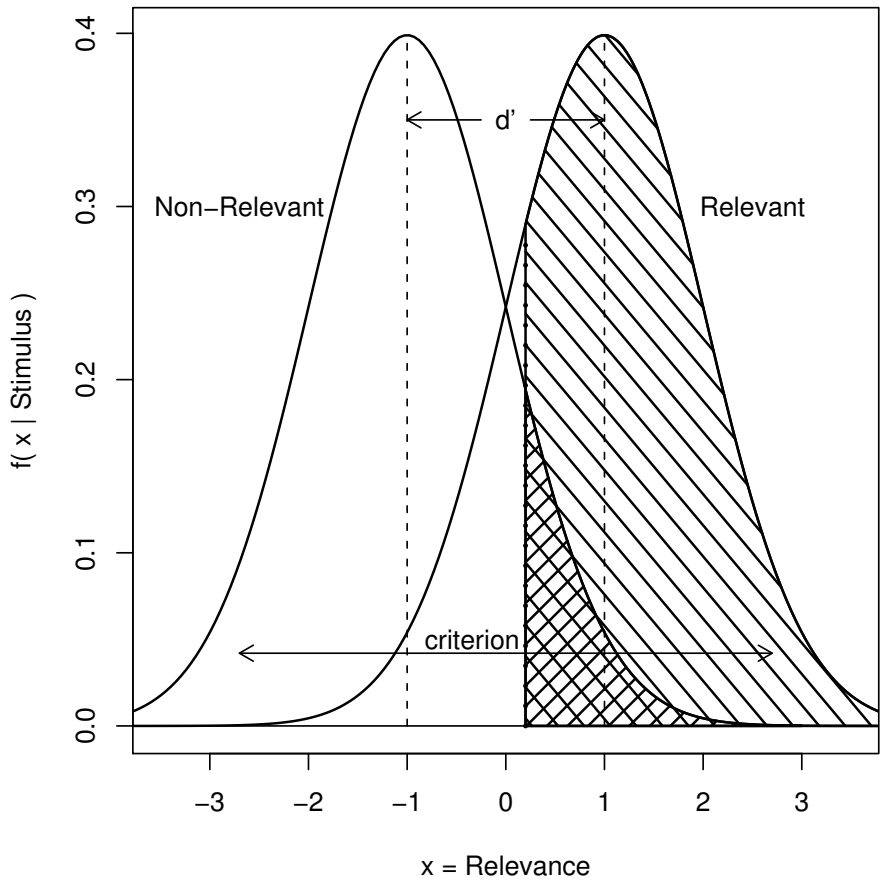



Figure 2.1: Sample figure for signal detection theory. The figure is from Prof. Mark Smucker, and is based on the Figure 2.2 in Macmillan and Creelman (2004).

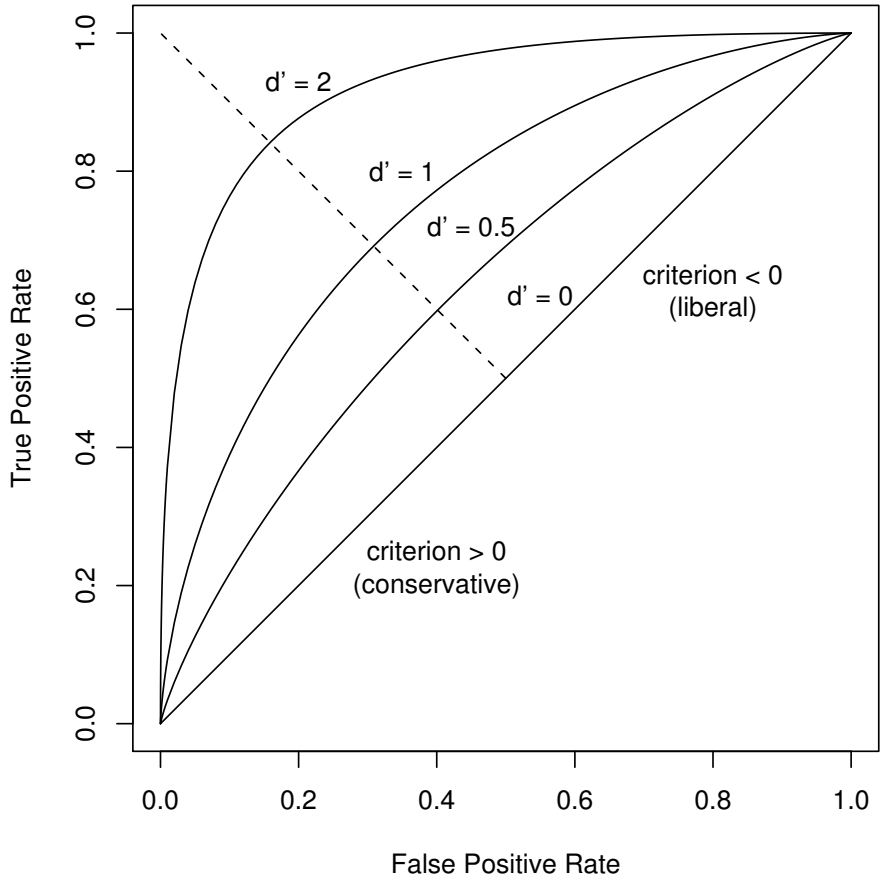



Figure 2.2: Sample figure for $d'$ and $c$, copied from Smucker and Jethani (2011b) with permission granted. This figure is based on Figure 1.1 and 2.1 in Macmillan and Creelman (2004).

# Chapter 3

# The Effects of Relevance Judgment Errors On Ranking System Evaluation

As explained in the Section 1.2, we can treat the consensus algorithm as a single pseudo assessor with some probabilities of making errors. In other words, we can investigate how a consensus algorithm affects the ranking system evaluation by analyzing the effects of relevance judgment errors made by a single assessor. By doing so can we provide a broad guidance on what kind of consensus algorithm or assessors are preferable. So that even though with errors, the judgments are still exchangeable with NIST assessors' in IR evaluation.

## 3.1 Simulated Dataset

It's hard to find assessors with all types of errors in practice. So, we mainly rely on signal detection theory to simulate the noisy judgments from various types of assessors by varying two variables, $d'$ and $c$, as mentioned in Algorithm 1. What are the candidate values of $d'$ and $c$ for simulation? Smucker and Jethani (2011a) estimate the average $d'$ and $c$ of NIST assessors to be 2.3 and 0.37, respectively, across 10 topics in 2005 TREC (Voorhees, 2005) Robust Track. In Smucker and Jethani (2011b), the reported $d'$ and $c$ of crowds assessors are 1.9 and 0.14, respectively, with the same experiment settings in Smucker and Jethani (2011a). If we think the judgments from NIST assessors as the upper bound

that the consensus algorithm or hired assessors could be, then the results from these two papers indicate that one assessor should have $d'$ and $c$ values close to NIST. Meanwhile, as shown in Figure 2.2, $d' = 0$ means the assessor labels the document by tossing an unbiased coin, i.e. random guess. So, we set the range of $d'$ as $[0.5, 3]$ with step size 0.5. As to criterion, the reported $c$ suggests that both NIST and crowds assessors are conservative (NIST assessors are more conservative than the crowds). At the same time, the behaviors of liberal assessors are also worthy of investigation. So, we set the $c$ with the range of $[-3, 3]$ with step size 0.1 In total, we simulated 366 different types of assessors who make random errors based on each one's $d'$ and $c$ values.

One thing to note here is that we set the standard deviations of $d'$ and $c$ to 0 since 1) we only need to investigate the behavior of a single assessor; and 2) the options for $d'$ and $c$ are sufficient to cover almost all types of crowds assessors we may encounter. So, we don't need to generate a random number from normal distribution.

The simulated judgments are generated based on the two datasets' qrels: 2005 TREC (Voorhees, 2005) Robust Track and Trec8 (Voorhees and Harman, 1999) ad-hoc Retrieval Track. To eliminate the random errors of the simulation, we generate 100 simulated runs for each assessor, which is equivalent to each assessors independently judges same set of documents 100 times, and then take average of the results. The details about how to do average are discussed later. For convenience, we refer the two datasets as Robust2005 and Trec8 in this thesis.

## 3.2 Statistics of Robust2005 and Trec8

This section describes the statistics of these two datasets.

Both datasets contain 50 topics. Robust2005 has 1.03 million documents in total. The pool of Robust2005 contains 37,800 documents, which indicates the mean pool size per topic is 756 (minimum 350, maximum 1390). The average relevance rate per topic in the pool is 18.3%.

Trec8 has around 50 million documents. The pool size is 86,750 for all 50 topics. The mean pool size per topic is 1735 documents. The average relevance rate per topic in the pool is 5.5%.

Table 3.1 summaries the pools' statistics of Robust2005 and Trec8. As we can see, Trec8 contains more number of documents as well as higher portion of non-relevant document per topic.

Robust2005 has 74 test runs while Trec8 has 129 test runs.

Table 3.1: The pools' statistics of Robust2005 and Trec8

| Dataset | Mean rel. rate per topic | std of rel. rate per topic | Mean # of docs per topic | std of the # of docs per topic |
|---|---|---|---|---|
| Robust2005 | 18.3% | 12.5% | 756 | 215 |
| Trec8 | 5.5% | 4.8% | 1735 | 427 |

## 3.3 Experiment Details and Evaluation Metrics

To measure the degree of correlation between the simulated and NIST assessors in IR evaluation, we test the submitted runs of those two TREC tracks against qrels from simulated and NIST assessors, respectively. To be more precise, as shown in Figure 3.1, we use each assessor's simulated runs as pseudo qrels, and then calculate the Mean Average Precision (MAP) values against the current pseudo qrels. Each test run corresponds to one MAP value against one qrel. For example, Robust2005 has 74 test runs and we can get 74 MAPs against one qrel file. Hence, we can measure the correlation of two qrel files based on the association between two lists of MAPs derived from identical test runs. The higher correlation between the rankings of test runs based on NIST qrel and rankings from pseudo assessor, the higher chance of replacing one with another in IR evaluation.

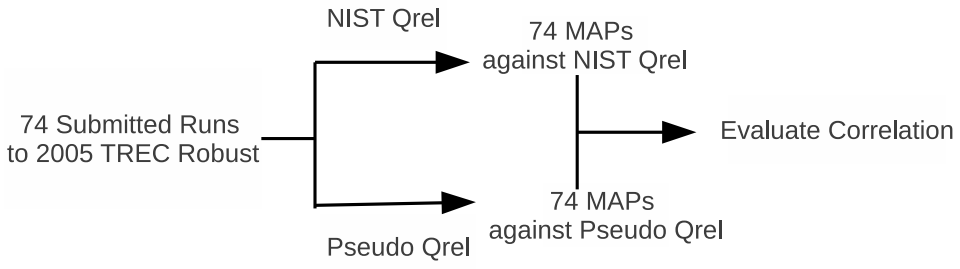



Figure 3.1: How to find correlation between one pseudo qrel and NIST qrel on Robust2005

The metric of measuring correlation we use in this thesis is Average Precision Correlation Coefficient (APCorr) (Yilmaz et al., 2008). The APCorr is very similar to another commonly-used correlation measurement Kendall's $\tau$ (Kendall, 1938), but giving more

weights to the errors nearer to the top of rankings. If two ranking systems perfectly agree with each other, both APCorr and Kendall's $\tau$ are 1. If two ranking systems are independent, the APCorr is 0.5 while Kendall's $\tau$ is 0. If totally disagree, the APCorr is 0 and Kendall's $\tau$ is -1. This chapter presents the results of both correlation metrics to show their relationship. In the later chapters, only APCorr is used as the correlation measurement since there two metrics are very similar.

Moreover, the Root Mean Square Error (RMSE) is also adopted to calculate the errors between two lists of MAPs. The smaller the RMSE value is, the closer two measurements are in terms of the quantity difference.

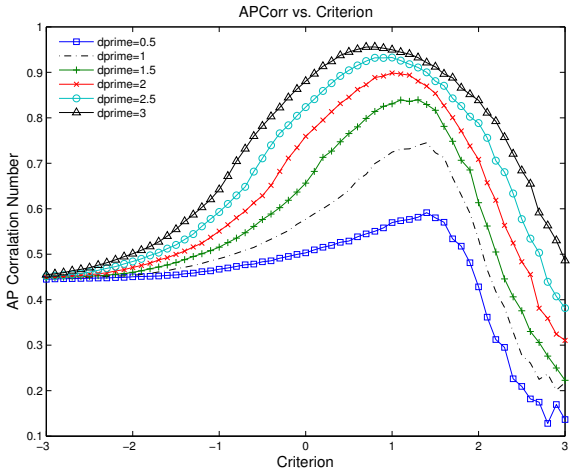
There are two strategies to average the results. The first one is *average the correlation*. Each assessor has 100 pseudo qrels in our simulation, which results in 100 correlation results. Then the mean value of those 100 correlations is used as output. Another strategy is *average the MAP*. This one averages the MAPs across 100 simulated runs first and then uses the mean MAP to calculate the correlation. This plausible method is essentially equivalent to calculate majority voting score across 100 assessors with identical judging ability. As shown in later chapter about the performance of majority voting, even 10 assessors' majority voting can achieve very high correlated results with NIST assessors, no need to mention 100 assessors. The correlation values of from "voted" MAPs are almost close to 1 in our experiment. So, *average the correlation* is used in the experiment.

## 3.4 Results and Discussions

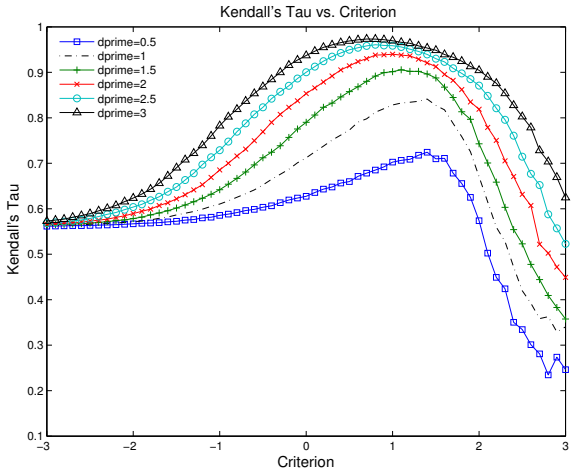
Figures 3.2 and 3.3 show the simulation results on Robust2005 and Trec8, respectively.

In the results of APCorr and Kendall's $\tau$, we can see that the larger $d'$ always have larger correlation with NIST qrels at same $c$. This is easy to understand that when two assessors are with the same criterion, the one with better discriminative ability is simply more preferable.

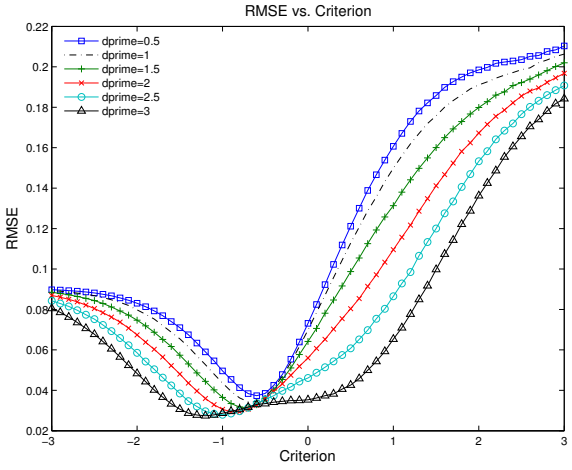
Also, as shown in Table 3.2 and 3.3, the larger $d'$ one assessor obtains, the smaller $c$ needed to achieve the best APCorr and Kendall's $\tau$ correlation. For example, when $d' = 0.5$, the $c$ corresponding to the maximal APCorr is 1.4. But when $d' = 2$, the optimal $c$ equals to 1. It means if we cannot guarantee our consensus algorithms or hired assessors have better discriminative ability than NIST assessors, we can still achieve good correlation by making this pseudo assessor more conservative (larger $c$) than the NIST assessors. An alternative is that we can relax the conservativeness of the pseudo assessor if the assessor's $d'$ is larger than NIST. However, finding an assessor with larger $d'$ than NIST is somehow




(a) APCorr

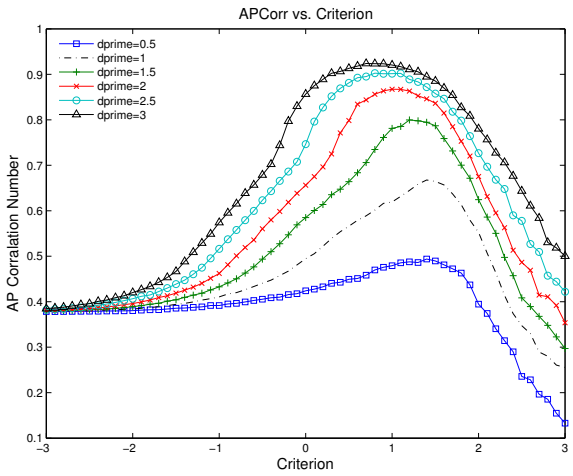



(b) Kendall's $\tau$

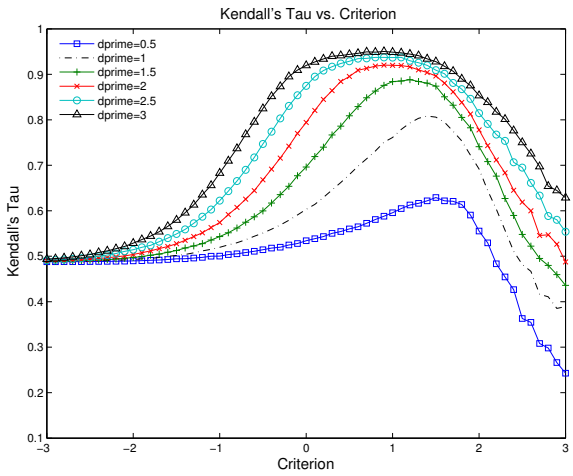



(c) RMSE

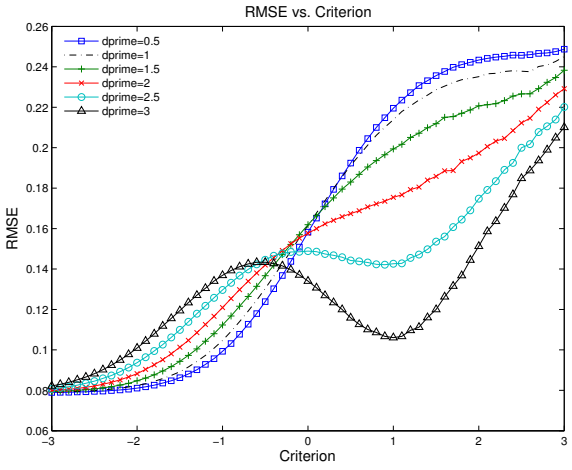
Figure 3.2: The effects of the pseudo assessor's erros on evaluation, Robust2005




(a) APCorr




(b) Kendall's $\tau$




(c) RMSE

Figure 3.3: The effects of the pseudo assessor's erros on evaluation, Trec8

Table 3.2: The criterion when the correlation is maximal for each $d'$ on Robust2005

| | AP Corr | | Kendall $\tau$ | |
|---|---|---|---|---|
| | max score | $c$ | max score | $c$ |
| $d' = 0.5$ | 0.591 | 1.4 | 0.724 | 1.4 |
| $d' = 1$ | 0.745 | 1.4 | 0.841 | 1.4 |
| $d' = 1.5$ | 0.840 | 1.3 | 0.906 | 1.1 |
| $d' = 2$ | 0.898 | 1.0 | 0.940 | 1.0 |
| $d' = 2.5$ | 0.932 | 1.0 | 0.961 | 0.8 |
| $d' = 3$ | 0.956 | 0.8 | 0.973 | 0.7 |

Table 3.3: The criterion when the correlation is maximal for each $d'$ on Trec8

| | AP Corr | | Kendall $\tau$ | |
|---|---|---|---|---|
| | max score | $c$ | max score | $c$ |
| $d' = 0.5$ | 0.494 | 1.4 | 0.629 | 1.5 |
| $d' = 1$ | 0.668 | 1.4 | 0.808 | 1.4 |
| $d' = 1.5$ | 0.800 | 1.2 | 0.888 | 1.2 |
| $d' = 2$ | 0.867 | 1.0 | 0.920 | 0.9 |
| $d' = 2.5$ | 0.903 | 0.8 | 0.937 | 0.8 |
| $d' = 3$ | 0.923 | 0.7 | 0.950 | 0.8 |

less optimal strategy to get relevance judgments. This is because we may need to invest more money and spend more time to train an assessor with better discriminative ability than the existing NIST assessors.

An important observation from the results of APCorr and Kendall's $\tau$ is that conservative assessors ($c > 0$) are more preferable than liberal assessors ($c < 0$) in ranking system evaluation. Intuitively, conservative assessors tend to label less relevant labels compared with liberal assessors. So, even with same discriminative ability, the conservative assessor has both smaller TPR and FPR values than liberal assessor. This indicates that it is critical to control the false positives, even at the expense of low true positives. Meanwhile if a conservative assessor labels one document as relevant, then it's very likely that NIST assessor will give same label. This is because the conservative assessors only give relevant label when the demanding criterion is met. So, even though conservative assessors may have lower TPR than liberal assessors, but the relevant labels given by conservative assessors tend to be more accurate than liberal assessors.

Hence, if we want to find an alternative to NIST qrel for ranking system evaluation, we should make sure that the judgments are from a conservative assessor or consensus algorithm. Similar observation can be found in Carterette and Soboroff (2010), though our simulation methods are different.

However, if we keep increasing the criterion of assessors, the correlation decreases no matter how good discriminative ability one assessor obtains. The reason for that is because an assessor with very high criterion means a relevant label is never being made until this picky assessor is very sure this document is relevant. This type of behavior inevitably downgrades the performance somehow. So we like conservative assessor at a reasonable range, but not too much.

Meanwhile, we notice that even though one assessor judges consistently across different datasets, the correlation results on Trec8 tend to be worse than that on Robust2005. The main reason for that is because Trec8 contains much more number of documents and smaller portion of relevant documents than Robust2005 (shown in Table 3.1). This makes the assessor harder to judge.

As to RMSE figure of Robust2005, we can see that the best RMSEs for all $d'$s happen when the assessors are liberal ($c = 0.6$), which seems to contradict to the conclusion in APCorr and Kendall's $\tau$ figures. However, RMSE is, essentially, a measurement of accuracy by calculating the absolute differences between the estimated and real data. It's not used to detect the correlation between ranking systems. To illustrate this point, we draw the distribution of all test runs against two qrels files (one qrel is from NIST and another one is from pseudo-assessor), as shown in Figures 3.5. They are both from Robust2005 dataset.

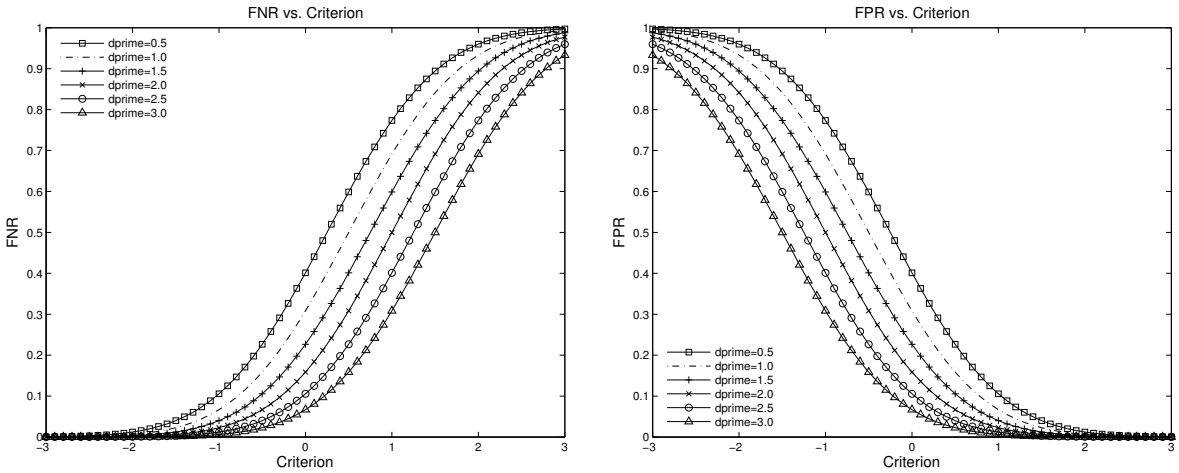
we only show the results of $30^{th}$ simulated run of $d' = 1, 2$ at criterion $c = \{1, -0.8\}$ (all other cases' figures show similar pattern). In each figure, each circle represents one specific test run. The $x$ coordinate of this circle is this test run's MAP against the NIST qrels while the $y$ coordinate is the MAP against the qrels from pseudo assessor. Ideally, if all circles locate in the dotted diagonal line, this mean the pseudo assessor judges exactly like NIST qrel (now RMSE = 0 and APCorr = 1). In practice, if two rankings have over 0.9 of Kendall's $\tau$ correlation, these two rankings are effectively equivalent and exchangeable (Carterette and Allan, 2005, Sanderson and Joho, 2004, Voorhees, 2005, 2001, Yilmaz and Aslam, 2006). As to APCorr, we also use 0.9 as the standard to measure whether two ranking systems are exchangeable or not, though the APCorr values tends to be smaller than Kendall's $\tau$ under same experiment settings in all experiment we did.

As we can see, assessors with $c = -0.8$ achieve the better RMSE than those with $c = 1$. However, $c = 1$ shows better AP Corr. Even though those circles of $c = -0.8$ (Figures 3.5(a) and 3.5(c)) are more closer than $c = 1$ (Figures 3.5(b) and 3.5(d)), i.e. smaller RMSE, they look less fit to a straight line as those circles in $c = 1$ (worse correlation). Moreover, when $c = 1$, Figures 3.5(b) and 3.5(d) show that a larger $d'$ makes all circles more fit to a straight line (higher correlation) as well as closer to the diagonal line (smaller RMSE). We also notice when the pseudo assessor with $d' = 2$ and $c = 1$ in Figure 3.5(d) has 0.94 Kendall's $\tau$, which means we can use this assessor's judgments to replace NIST. $d' = 2$ and $c = 1$ actually are very close to the reported $d'$ and $c$ values of NIST assessor in Smucker and Jethani (2011a). Similar run distribution plots on Trec8 dataset can be viewed in the Figure A.1 in Appendix A.

However, the RMSE figure of Trec8 (Figure 3.3) looks different from Robust2005 in the way that larger $d'$ doesn't imply better RMSE results when $c$ is liberal. We think there are several potential reasons for that.

The first reason is that it's hard to guarantee we can achieve both good RMSE and correlation results. Figure 3.6 shows the MAPs distributions of $d' = 3, c = \{-2, -1, 0, 1, 2\}$ on Trec8. We don't show the MAPs distributions of $c = \{-3, 3\}$ because assessors with $c < -2$ label almost all documents to be relevant while those with $c > 2$ label almost all documents to be non-relevant (as shown in Figure 3.4). In reality, we seldom observe such behaviors on ethnical assessors. Those pseudo qrels with good APCorr results do not have the best RMSE and the circles look more away from the diagonal line. Essentially, we need to make a tradeoff between RMSE and correlation.

Secondly, the fact that Trec8 contains lots of non-relevant documents in the pool causes problem. Trec8 has a larger averaged pool depth and smaller average relevance rate in the pool than Robust2005. Our hypothesis is that the assessor generates higher number of



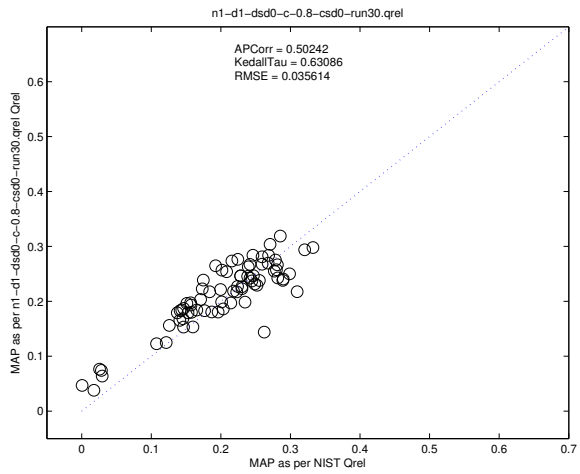

(a) FNR vs. Criterion

(b) FPR vs. Criterion

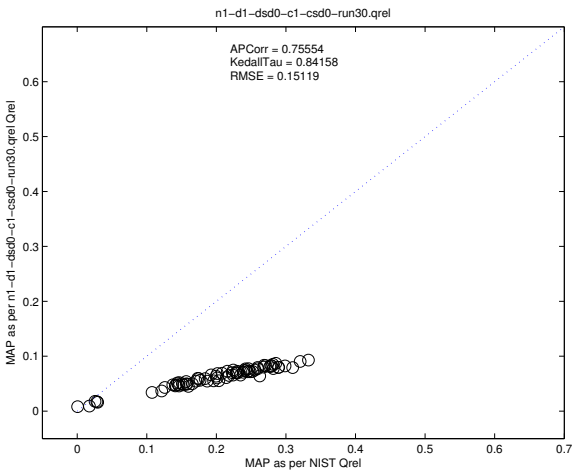
Figure 3.4: How FNR and FPR change according to $d'$ and $c$

false positives on Trec8, though this assessor has same false positive rate across datasets.

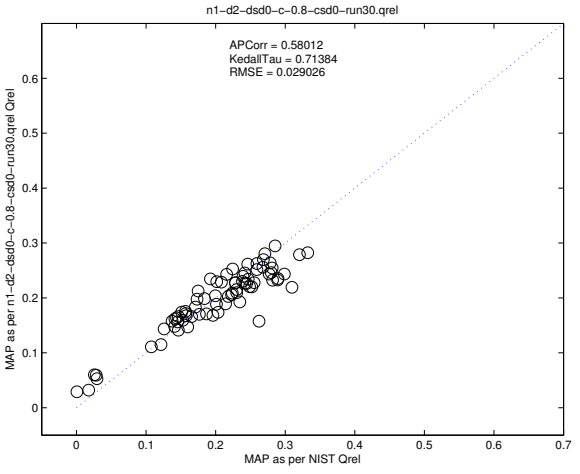
Meanwhile, our simulation does not reflect that non-random errors of assessors. For example, Webber et al. (2012) find that the retrieval depth is related with assessors' errors, and assessors is likely to make mistakes on following two types of documents: 1) documents ranked low by the runs (the bottom of the pool) but labeled as relevant; 2) documents ranked high by the runs (the top of the pool) but labeled as non-relevant. However, our simulation cannot capture such non-random errors.



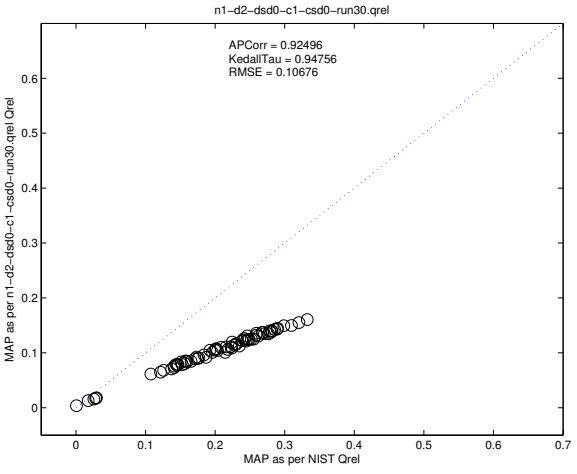

(a) $d' = 1, c = -0.8$, Simulated Run 30




(b) $d' = 1, c = 1$, Simulated Run 30

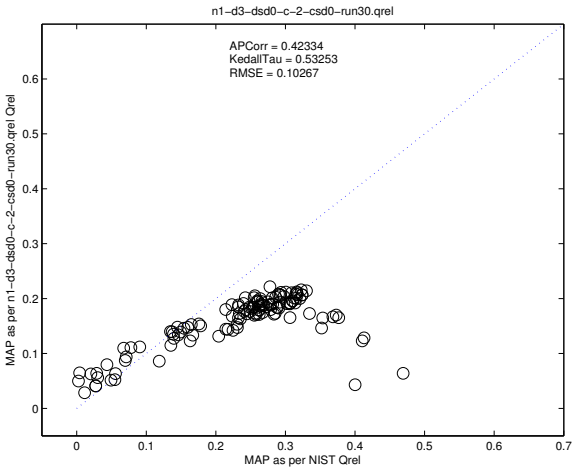



(c) $d' = 2, c = -0.8$, Simulated Run 30

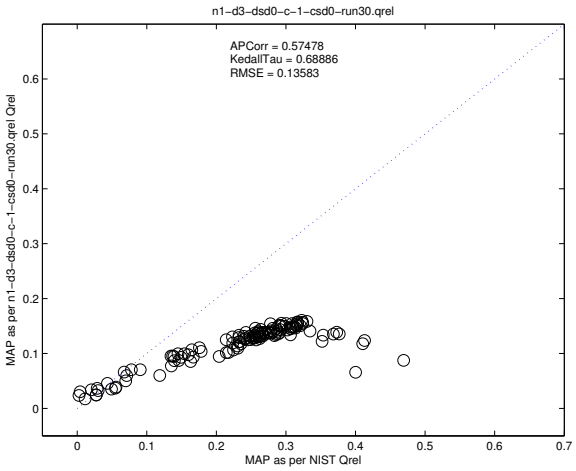



(d) $d' = 2, c = 1$, Simulated Run 30

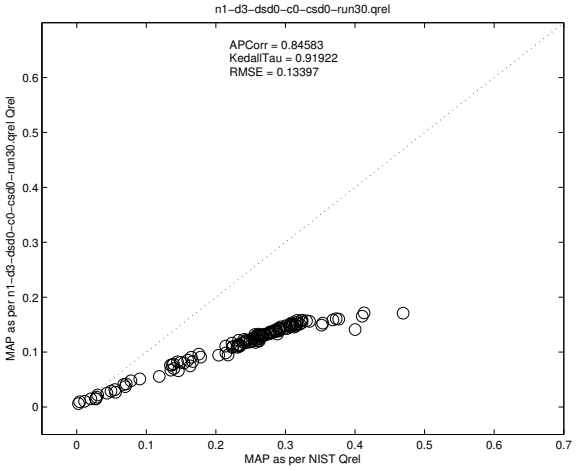
Figure 3.5: MAPs distribution against NIST qrels and pseudo qrels on Robust2005



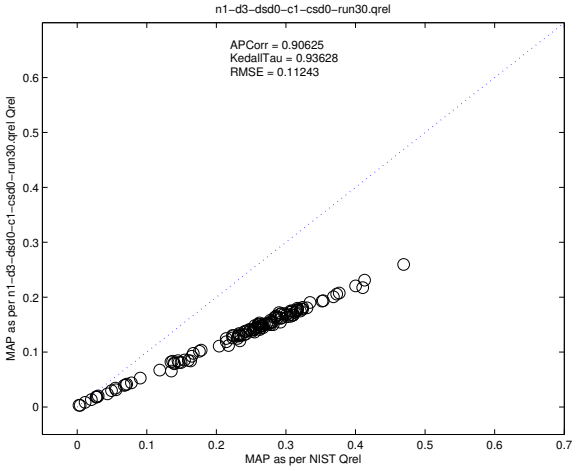

(a) $d' = 3, c = -2$, Simulated Run 30



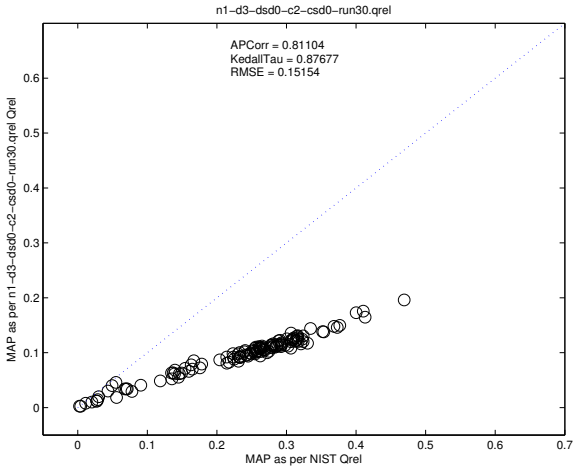

(b) $d' = 3, c = -1$, Simulated Run 30




(c) $d' = 3, c = 0$, Simulated Run 30




(d) $d' = 3, c = 1$, Simulated Run 30




(e) $d' = 3, c = 2$, Simulated Run 30

Figure 3.6: MAPs distribution against NIST qrels and pseudo qrels on Trec8, $d' = 3, c = \{-2, -1, 0, 1, 2\}$

# Chapter 4

# Deal With Ties In Consensus Decision-Making

This chapter mainly focuses on two issues.

The first issue is how to deal with the tie situations in decision-making process. The tie means we have same confidences to believe one document to be relevant as well as non-relevant. For example, if we set the threshold of relevance as 0.5, a document is labeled as relevant if its probability is larger than 0.5 and non-relevant for smaller than 0.5. However, how to make decision if the document's relevance is exactly 0.5? This issue has been less explored in previous work. In majority voting, the ties often happens when we have even number of assessors and half of the assessors think one document is relevant. In this chapter, we propose five strategies to break ties.

In the meantime, we investigate the effects of varying threshold of relevance based on the following two reasons: Firstly, even though 0.5 is commonly used as the threshold of relevance, we are still not sure are there any benefits if we become more strict about the minimum requirement of agreement, which is similar to redefine the definition of "majority". Secondly, as shown in Chapter 3, we know that it's better making our consensus algorithm be conservative. It's very natural to think of using the relevance standard to control the criterion of the consensus algorithm. Essentially, a higher relevance standard means the algorithm produces more non-relevant labels.

## 4.1 Strategies For Ties Situations

If tie situation happens, it means we may need to move to other useful metrics or information to help us make decisions. Let *thresRel* denote the threshold of relevance. So, we try to determine the correct labels for those documents with the probability of *thresRel*. As far as we know, very few studies have systematically addressed this issue.

Five strategies are proposed in this thesis.

1. *Larger*: This strategy just simply label those ties documents as non-relevant since this strategy believes a relevant document should have larger relevance than *thresRel*.

2. *LargerEqual*: On the contrary, this strategy labels all ties documents to be relevant since it thinks a relevant document has a larger than or equal to *thresRel* relevance.

3. *CoinThresRel*: this strategy determines the consensus decision by "tossing a coin", where the chance of being relevant is related with the *thresRel* . As shown in Equation 4.1, if the random number in $[0, 1]$ is larger than or equal to *thresRel*, we think this document is relevant; otherwise non-relevant.

$$\text{Consensus Decision} = \begin{cases} \text{Relevant}, & \text{if } rand(0,1) \geq thresRel. \\ \text{Non-Relevant}, & \text{otherwise.} \end{cases} \tag{4.1}$$

4. *CoinPrevalence*: This strategy also tosses a coin but it trusts the prevalence information more. We use the same way in Raykar et al. (2010) to calculate the prevalence (shown in Equation 4.2).

$$prevalence = \frac{\sum_{i=1}^{N} \mu_i}{N} \tag{4.2}$$

Where $N$ is the number of all documents (not just those documents with ties) judged in this topic and $\mu_i$ is the relevance of the $i^{th}$ document. After the prevalence is calculated, we can toss a coin and compare the random number with the prevalence, as shown in Equation 4.3

$$\text{Consensus Decision} = \begin{cases} \text{Relevant}, & \text{if } rand(0,1) \leq prevalence. \\ \text{Non-Relevant}, & \text{otherwise.} \end{cases} \tag{4.3}$$

5. *MajorClass*: This strategy, as shown in Equation 4.4, intends to label those uncertain documents to the "major class" it believes in, where the "major class" is related with the estimated prevalence. However, if the estimated prevalence equals to the *thresRel*, we cannot apply this method since the major class is uncertain, too. In this case, we can fall back to the *CoinPrevalence* strategy.

$$\text{CONDITION} : prevalence \neq thresRel.$$
$$\text{Consensus Decision} = \begin{cases} \text{Relevant}, & \text{if } prevalence > thresRel. \\ \text{Non-Relevant}, & \text{if } prevalence < thresRel. \end{cases} \tag{4.4}$$

As we can see, the first two strategies are pretty straightforward. They simply label the uncertain documents to be relevant or non-relevant based on some certain of strong believes. The third one makes useful of *thresRel* to guess the relevance. *CoinPrevalence* and *MajorClass* both leverage the prevalence information of dataset in their own ways while the other three do not.

## 4.2 Experiment Settings

We use the same way in Chapter 3 to do simulation. For each document, 10 cases are simulated according to the number of the assessors. The minimal number of assessors who judge the same document is 1. The maximum is 10. Meanwhile, we set the standard deviations of $d'$ and $c$ to be non-zero to add varieties of assessors. This is because people are different by nature. If $csd = 0$ ($dsd = 0$), it means all assessors are with identical criterion (discriminative ability). We need variations among assessors who judge the same set of documents. The detail simulation settings are:

- $n$ =1 to 10, where $n$ is the number of assessors who judge the same documents
- $d = \{1, 2\}$, where $d$ is the discriminative ability of assessor
- $dsd = 1$, where $dsd$ is the standard deviation of $d$
- $c = \{-0.5, 0, 0.5\}$, where $c$ is the criterion of assessor
- $csd = 0.5$, where $csd$ is the standard deviation of $c$

As we can see, each dataset has $10*2*1*3*1 = 60$ cases. Each case has 100 simulated runs to eliminate the random errors in simulation. We use the *average the correlation* to aggregate the results. For convenience, we use $nX$–$dX$–$dsdX$–$cX$–$csdX$ to represent each simulation situation, where X is the value of the variable it follows. For example, $n2$–$d2$–$dsd1$–$c-0.5$–$csd0.5$ means we simulate two assessors, each one's $d'$ is generated by a Gaussian distribution with the mean of 2 and standard deviation of 1. The $c$ is derived from another Gaussian distribution $\mathcal{N}(-0.5, 0.5)$. Furthermore, $dX$–$dsdX$–$cX$–$csdX$ covers all 10 cases when we gradually increase the number of assessors from 1 to 10.

Particularly, the simulation can be divided into the following 6 categories:

- $d1$–$dsd0.5$–$c-0.5$–$csd0.5$: low discriminative ability, liberal;
- $d1$–$dsd0.5$–$c0$–$csd0.5$: low discriminative ability, neutral;
- $d1$–$dsd0.5$–$c0.5$–$csd0.5$: low discriminative ability, conservative;
- $d2$–$dsd0.5$–$c-0.5$–$csd0.5$: high discriminative ability, liberal;
- $d2$–$dsd0.5$–$c0$–$csd0.5$: high discriminative ability, neutral;
- $d2$–$dsd0.5$–$c0.5$–$csd0.5$: high discriminative ability, conservative;

Moreover, based on the conclusion in Chapter 3, it's important to keep the false positives low at the expense of true positives. We vary the threshold of relevance from 0.5 to 1 with step size 0.1. The higher the $thresRel$ is, the fewer documents that will be judged as relevant. Essentially, the $thresRel$ defines the "majority".

## 4.3 Results and Discussions

### 4.3.1 Comparison Between Strategies For Ties

Firstly, we compare all five strategies to break ties in majority voting. We use 0.5 as the relevance standard. The APCorr results on Robust2005 and Trec8 are shown in Figures 4.1 and 4.2, respectively.

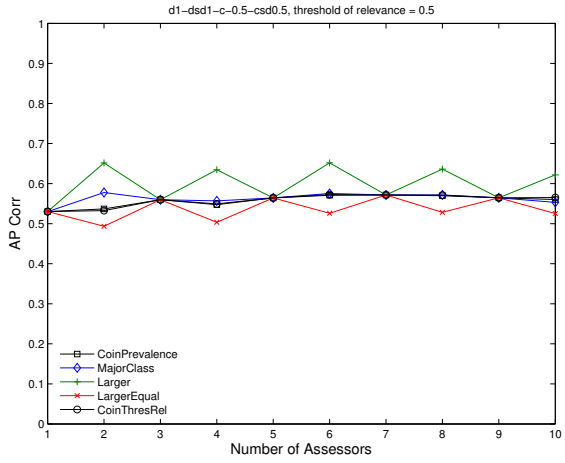
Many interesting observations can be derived from the two figures with $thresRel = 0.5$. Firstly, based on the APCorr results, we can rank the strategies in the following decreasing

order: *Larger*, *MajorClass*, *CoinPrevalence*, *CoinThresRel*, and *LargerEqual*. The reason why *Larger* performs the best while *LargerEqual* is the worst is due to the fact that true relevant documents only occupy a small portion across all topics in both datasets. So, guessing the uncertain documents as non-relevant seems to be less risky than relevant guess. It also happens to match with our goal that controls the number of false positives. However, *Larger* strategy is based on the strong belief that a document tends to be non-relevant. If we don't have such prior information, the performance of *Larger* may not be that good. For example, if we are given a dataset where most documents are relevant, *LargerEqual* is very likely to outperform *Larger* in this case.

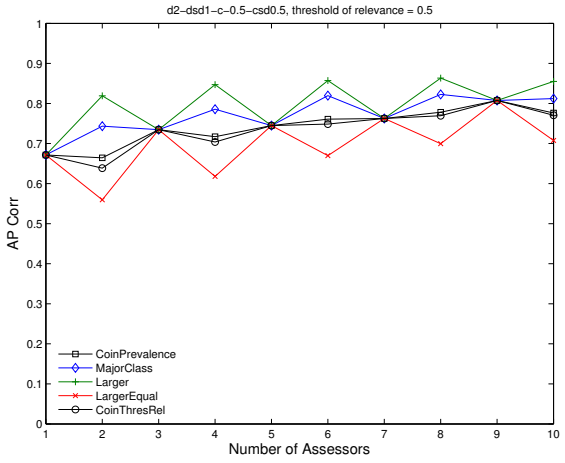
Secondly, *MajorClass* achieves the second best in all figures. Basically, this strategy looks like a generalized form of *Larger* and *LargerEqual* in the sense that *Larger* (*LargerEqual*) always believes the major class to be non-relevant (relevant) while the major class of *MajorClass* is derived from the estimated prevalence. Actually, in the experiment on both datasets, a few topics' prevalence are larger than the $thresRel$, which makes this strategy slightly worse than *Larger*. However, this strategy is more scalable than others since it's also suitable to the situations that we have zero knowledge about the statistics of dataset. *CoinPrevalence* and *CoinThresRel* strategies are worse than *MajorClass* since they two have higher possibilities to label the uncertain documents to be relevant.

The reader may notice that all strategies achieve same APCorr results when $n$ is odd. This is because there are no ties when $n$ is odd and the $thresRel$ is 0.5.

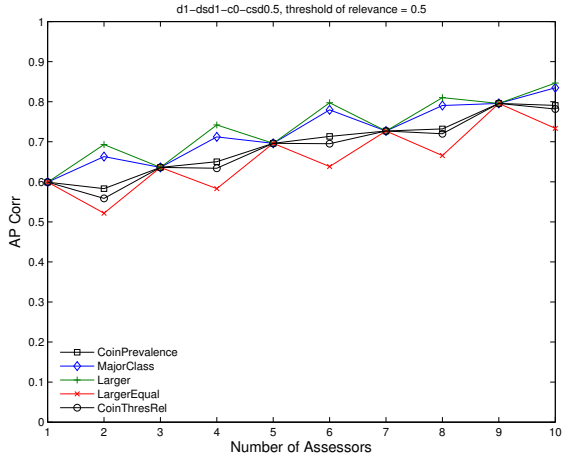
In sum, two ways of dealing with ties are recommended. If we have strong belief that implies the correct label of the uncertain documents, then we should use *LargerEqual* or *Larger* strategy; otherwise, we should use *MajorClass* if we don't have any prior information about the dataset.



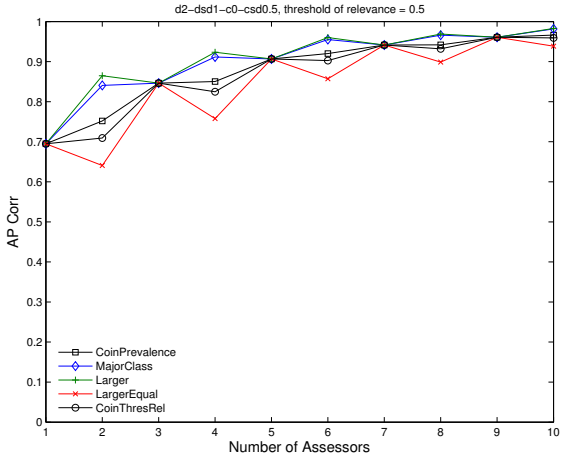

(a) $d' = 1, c = -0.5$



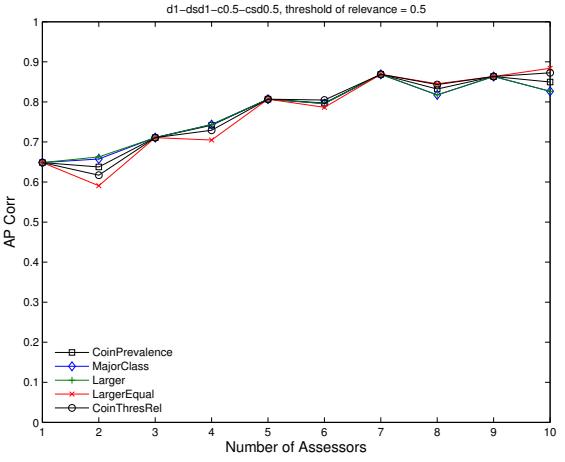

(b) $d' = 2, c = -0.5$



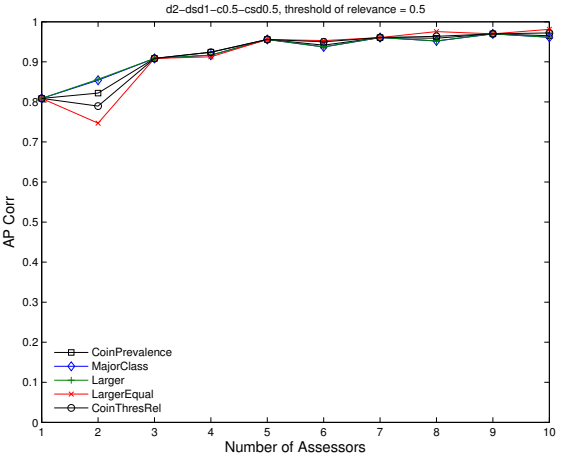

(c) $d' = 1, c = 0$




(d) $d' = 2, c = 0$

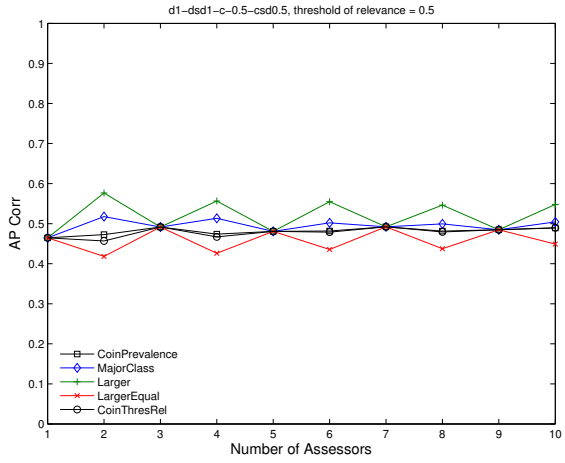



(e) $d' = 1, c = 0.5$

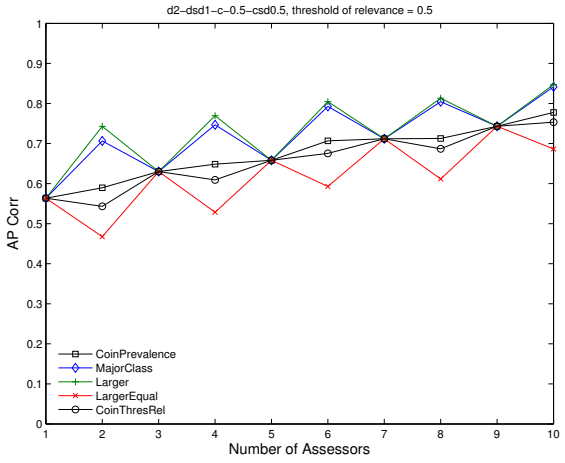



(f) $d' = 2, c = 0.5$

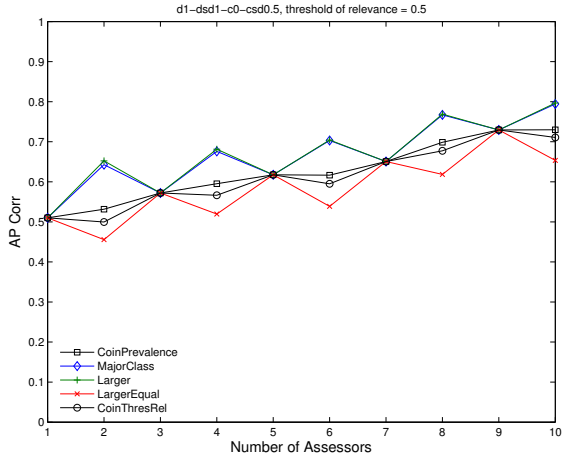
Figure 4.1: The APCorr results of 5 strategies to break ties on Robust2005, threshold of relevance = 0.5



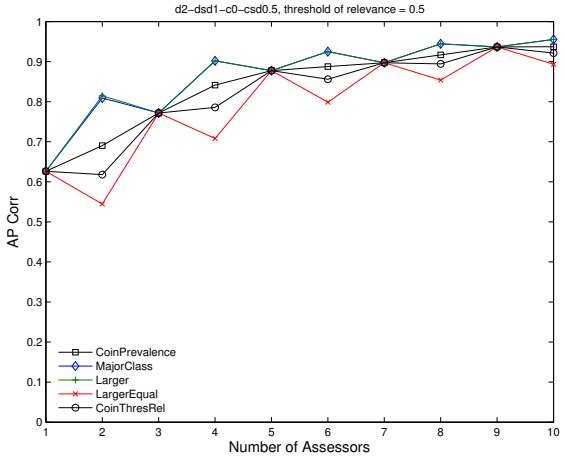

(a) $d' = 1, c = -0.5$



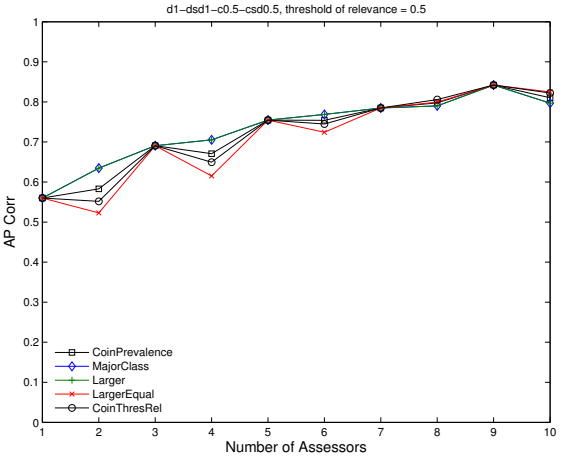

(b) $d' = 2, c = -0.5$



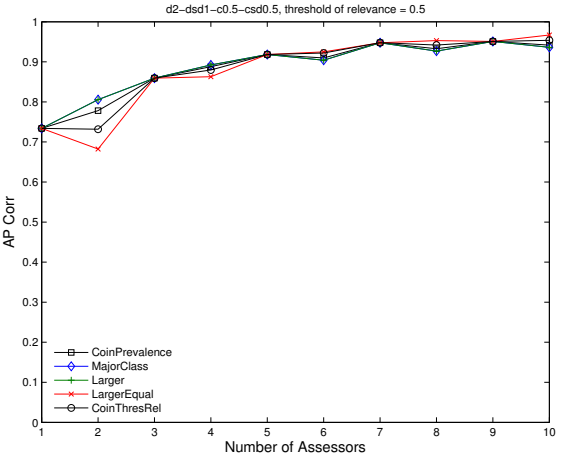

(c) $d' = 1, c = 0$




(d) $d' = 2, c = 0$




(e) $d' = 1, c = 0.5$




(f) $d' = 2, c = 0.5$

Figure 4.2: The APCorr results of 5 strategies to break ties on Trec8, threshold of relevance = 0.5

### 4.3.2 The Effects of Varying Threshold of Relevance on Correlation

In previous sections, we have shown that we need to control the false positive since it is harmful to the rankings. Besides making the assessors more conservative (positive $c$), there is another way to make the overall rankings be more conservative by redefining the decision rule. Even though 0.5 is commonly adopted as the threshold of relevance, it's necessary to understand the behaviors if we gradually increase the $thresRel$.

Figure 4.3 and 4.4 shows how the decision rule affects the ranking system evaluation for $d1$–$dsd1$–$c0$–$csd0.5$ on two datasets. The reason why we are especially interested in the results of $d1$–$dsd1$–$c0$–$csd0.5$ is because the $d' = 1$ and $c = 0$ are very close to the estimated crowdsourcing assessors' values as reported in Smucker (2012).

The APCorr increases as $thresRel$ increase from 0.5 to 0.7 and reaches the best performance when $thresRel = 0.7$. $c = 0$ means those simulated judgments are neutral in general. By being more strict to the relevance standard can we make the overall judgments be more conservative. However, keeping increasing the $thresRel$ doesn't seem to be a good idea when $thresRel$ is larger than 0.7. The assessors only think a document is relevant only when they are very sure that the document satisfies their demanding criterion. Even though in this case we guarantee the false positives are low, the extreme conservative assessors are not that preferable.

Hence, we need essentially make sure the conservativeness of consensus labels are at a reasonable level. Liberal or too conservative consensus labels are bad to the ranking systems.

So far, we know there are two factors that affect the conservativeness of the consensus labels. Also we know how the correlation with the experts labels changes when both the judgments' criterion and decision rule are conservative. We still do not know what if one factor is conservative and another factor is liberal.

Our conclusion is that if one factor is conservative while another one is liberal, the estimated consensus labels can still achieve high correlation with expert labels. We have two observations to support this point.

Firstly, in Figure 4.3 and 4.4, when $threRel$ is larger than or equals to 0.7, we can see that *LargerEq* performs the best. This is because when we already have a very conservative judging system, a comparatively liberal strategy, like *LargerEq*, keeps the system into a reasonable degree of conservativeness.

Secondly, we generate the APCorr results of varying threshold of relevance for judgments from liberal assessors on Robust2005, as shown in Figure 4.5. In Figure 4.5(a), we

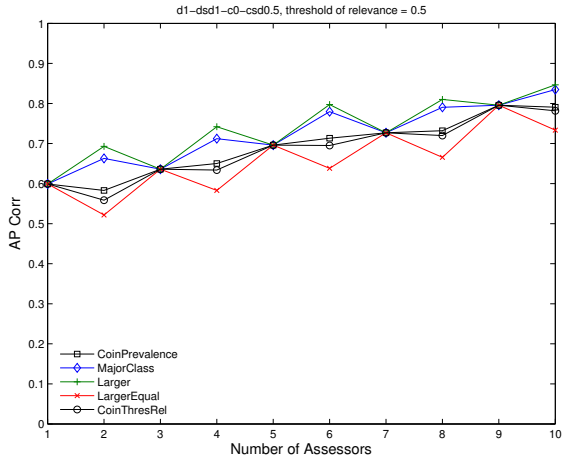
Table 4.1: The best threshold of relevance for assessors with $d' = 1$

| Assessors | Robust2005 | Trec8 |
|---|---|---|
| $d' = 1, c = 0.5$, conservative | 0.5 | 0.5 |
| $d' = 1, c = 0$, neutral | 0.7 | 0.7 |
| $d' = 1, c = -0.5$, liberal | 0.8 | 0.8 |

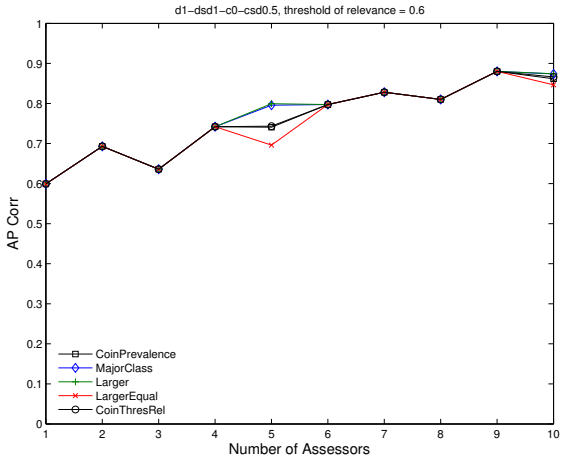
don't see the benefits of involving more assessors to judge when all assessors are liberal and the threshold of relevance is 0.5. It matches results in Chapter 3 that liberal consensus labels hurt the correlation. However, if we keep increasing the decision rule to 0.8, we can clearly see the improvement in the APCorr. This shows that we can still get a benefit from liberal assessors by raising the threshold of relevance. The APCorr results of liberal assessors on Trec8 can be viewed in Figure B.1 at Appendix B.

If the assessors are already very conservative ($c = 0.5$), then we can see the benefit of keeping the relevance standard liberal (The best threshold of relevance is 0.5 in this case). The APCorr results of conservative assessors on Robust2005 and Trec8 can be viewed in Figure B.2 and B.3, respectively.

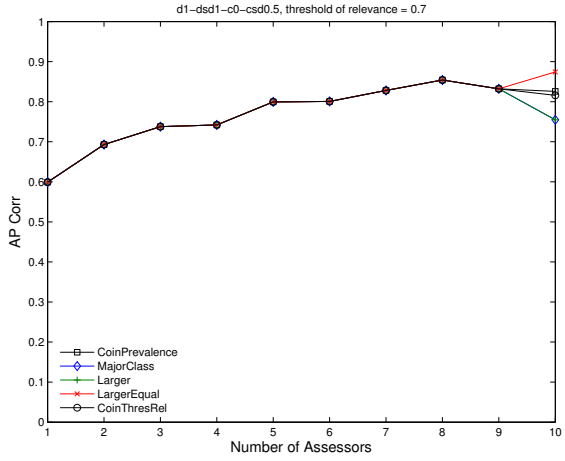
Table 4.1 summaries the best decision rules when we have different types of assessors. In short, if one factor is too conservative (liberal), the other factor should be liberal (conservative).



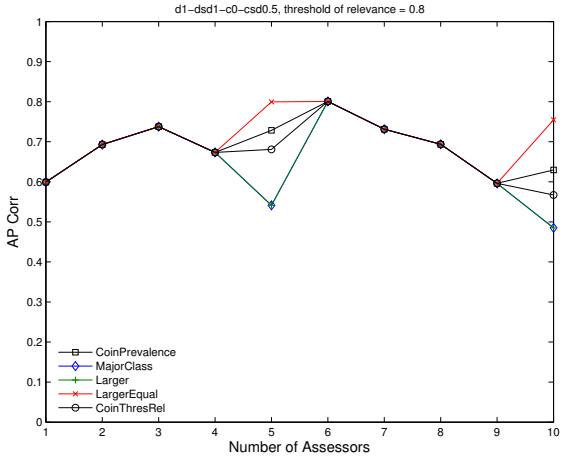

(a) Threshold of Relevance = 0.5



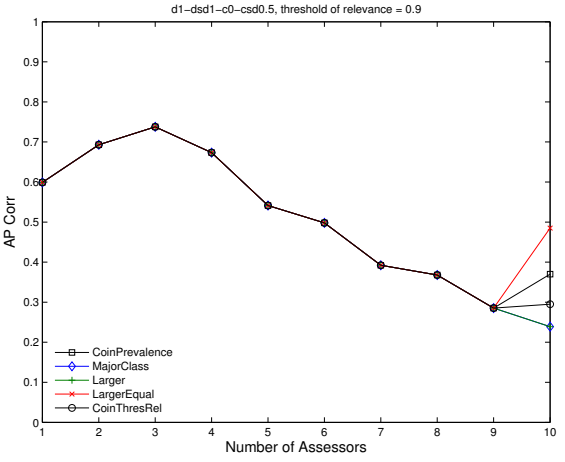

(b) Threshold of Relevance = 0.6



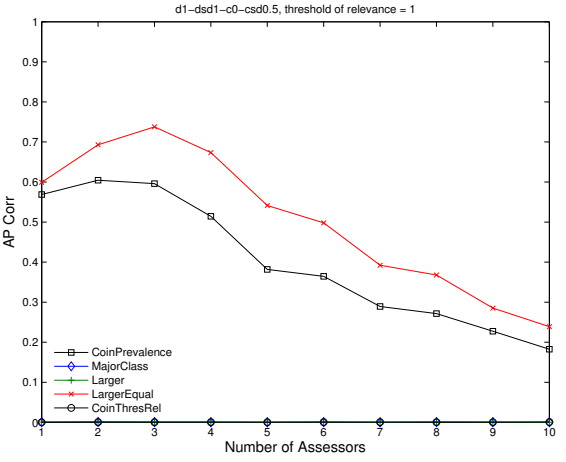

(c) Threshold of Relevance = 0.7




(d) Threshold of Relevance = 0.8

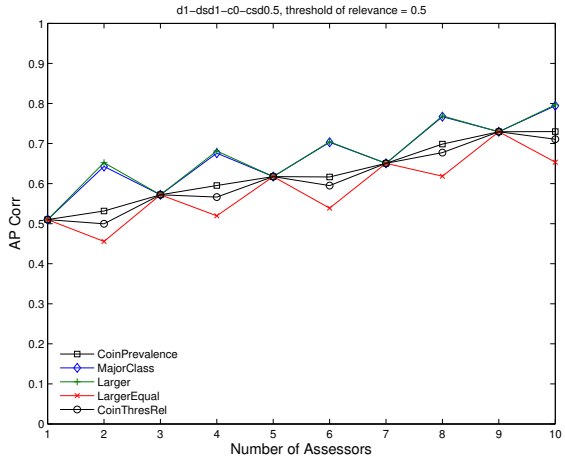



(e) Threshold of Relevance = 0.9

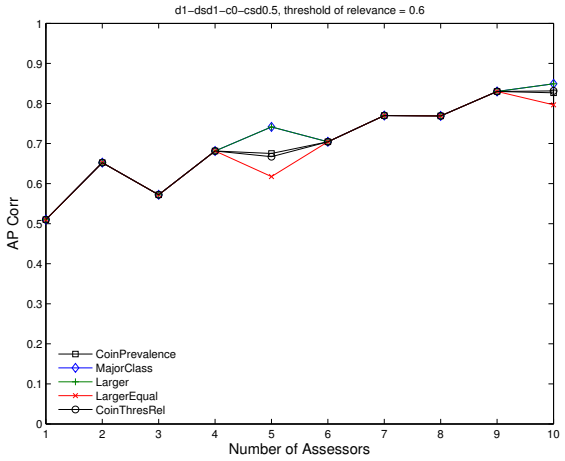



(f) Threshold of Relevance = 1

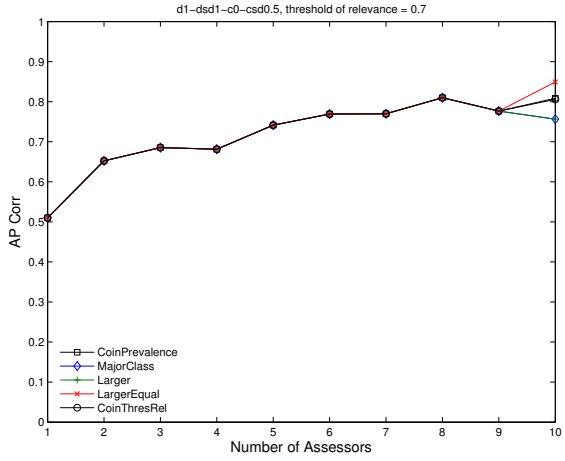
Figure 4.3: The APCorr results of varying threshold of relevance for $d1$–$dsd1$–$c0$–$csd0.5$ on Robust2005



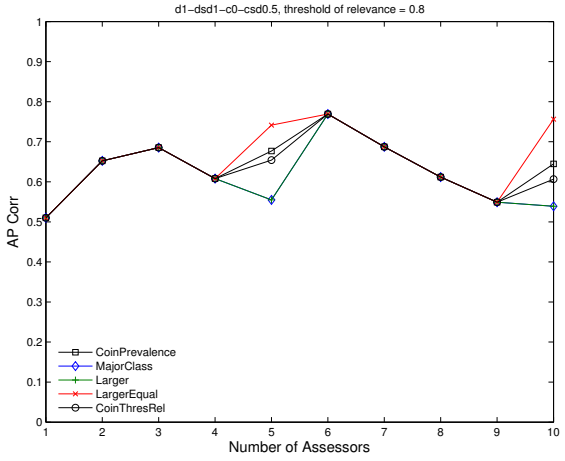

(a) Threshold of Relevance = 0.5



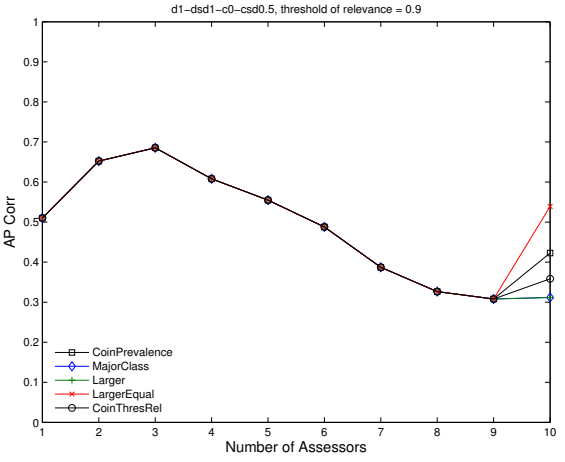

(b) Threshold of Relevance = 0.6



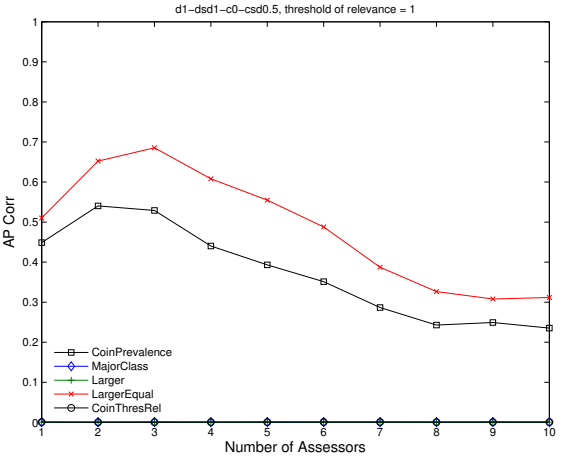

(c) Threshold of Relevance = 0.7




(d) Threshold of Relevance = 0.8

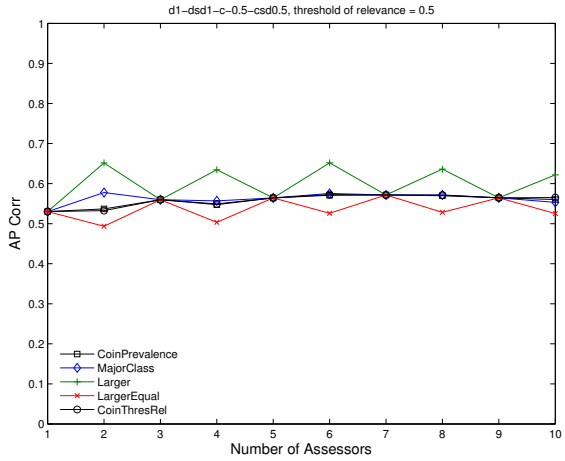



(e) Threshold of Relevance = 0.9

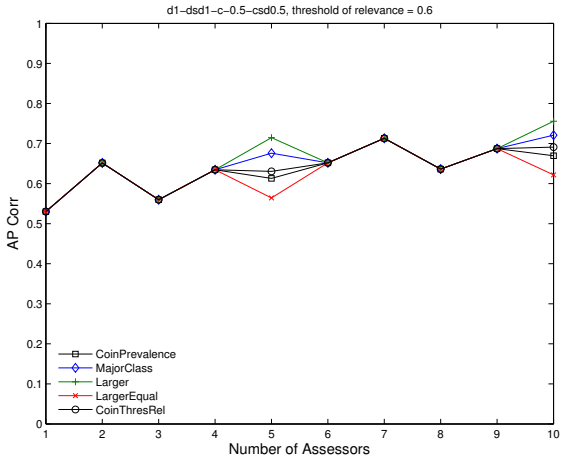



(f) Threshold of Relevance = 1

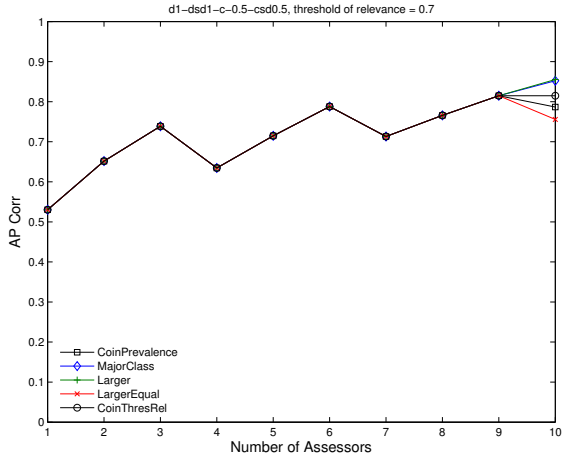
Figure 4.4: The APCorr results of varying threshold of relevance for $d1$–$dsd1$–$c0$–$csd0.5$ on Trec8



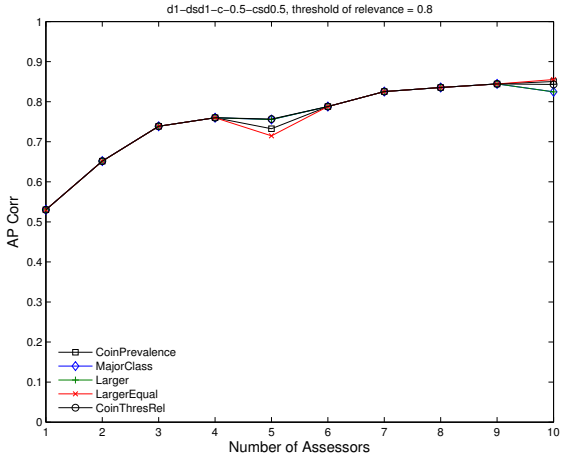

(a) Threshold of Relevance = 0.5



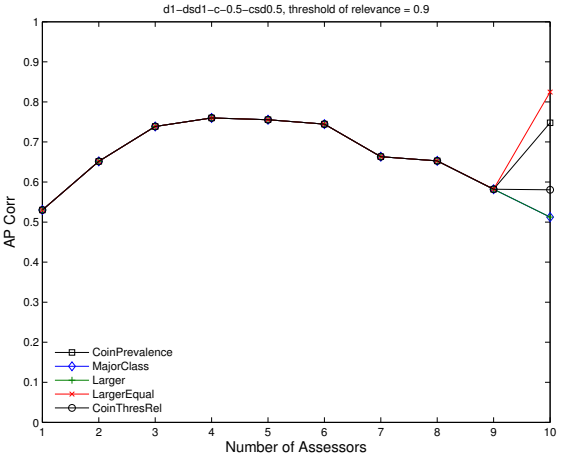

(b) Threshold of Relevance = 0.6



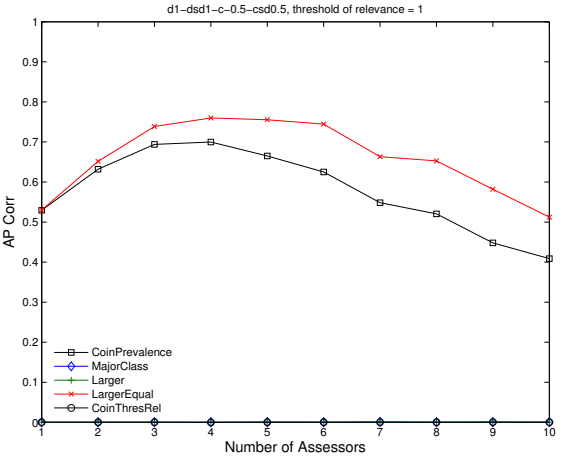

(c) Threshold of Relevance = 0.7




(d) Threshold of Relevance = 0.8




(e) Threshold of Relevance = 0.9




(f) Threshold of Relevance = 1

Figure 4.5: The APCorr results of varying threshold of relevance for $d1$–$dsd1$–$c-0.5$–$csd0.5$ on Robust2005

# Chapter 5

# Content-based Consensus Algorithm

The diverse nature of assessors requires our decision process rely more on those judgments from more trustable assessors. This is the key principle to design the consensus algorithm. Current work on finding consensus (Li et al., 2011, Raykar et al., 2009, 2010, Smucker, 2012, Tang and Lease, 2011, Vuurens et al., 2011, Yan et al., 2011) too focus on the provided noisy judgments to estimate the judging ability of each assessors. Nevertheless, little attention is paid to the content of document in IR field, which is another available and important resource we can leverage in our inference process.

Thus, this chapter investigates the potential benefits of utilizing document content to find consensus. Our implementation is largely based on the content-based method in Raykar et al. (2010) that targets on medical application. The algorithm in Raykar et al. (2010) composes of two key parts: Maximum Likelihood Estimator (MLE) and Logistic Regression (LR). Each algorithm relies on the other one's estimations to train the model, produce the labels, and send labels back. To be more precise, MLE utilizes the weak predictions from LR as the pseudo-true labels, and then estimates the labels based on the noisy judgments. LR mainly predicts based on the document content, after the classification model is trained based on the MLE estimation. This iterative process does not stop until both algorithms reach agreements on the document labels.

The reason why we use this iterative method is based on two considerations. Firstly, this method can be applied to the case when we have zero prior knowledge about the gold standard data. Secondly, this method is especially suitable for those problems with lots of missing data (in our case, zero true labels). This method can iteratively guess the missing labels by their conditional expectations, given noisy labels, document content, and the current estimation for the labels. Essentially, this EM algorithm is very similar to

the co-training theory, as declaimed in Blum and Mitchell (1998). Its basic principle is that we can have two different and complementary "views" of the documents. Based on these two conditionally independent and sufficient views of data, we can build two types of classifiers. So that the weakly labels from one classifier can be used as input to the other classifier. This iterative process keeps running until it reaches convergence. We can interpret convergence as the case that both classifiers agree on the probability of relevance for all or most documents. In our case, we regard those noisy judgments as one view to help us understand the document. At the same time, the content can be another vital view.

However, the method in Raykar et al. (2010) is not applicable to the situation when the dimension of feature vector is high, since their LR involves computationally-intensive calculation of Hessian matrix. So, we substitute their LR part with stochastic gradient descent algorithm in Cormack et al. (2011). We compare this content-based algorithm against other consensus algorithms in both relevance evaluation and ranking system evaluation.

## 5.1 Algorithms

### 5.1.1 Maximum Likelihood Estimator

We use the way in Raykar et al. (2010) to construct MLE. Assuming we have an external classifier/hypothesis, $h_\theta(x_i)$, which can output the predicted label for the $i^{th}$ document with feature vector $x_i$. Given N documents, we can construct the input feature space $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$. Similarly, we can represent all judgments from assessors as $\mathcal{Y} = \{y_i^1, y_i^2, \ldots, y_i^R\}_{i=1}^N$. Hence, the likelihood function is:

$$\begin{aligned} L(\theta) &= Pr(\mathcal{Y}|\mathcal{X}, \theta, \alpha, \beta) \\ &= \prod_{i=1}^{N} Pr(y_i^1, y_i^2, \ldots, y_i^R \ |x_i, \theta, \alpha, \beta) \\ &= \prod_{i=1}^{N} \{Pr(y_i^1, y_i^2, \ldots, y_i^R \ |y_i = 1, \alpha) Pr(y_i = 1|x_i, \theta) \\ &+ Pr(y_i^1, y_i^2, \ldots, y_i^N \ |y_i = 0, \beta) Pr(y_i = 0|x_i, \theta)\} \end{aligned} \tag{5.1}$$

In above equation,

$$\begin{aligned} Pr(y_i = 1|x_i, \theta) &= h_\theta(x_i) \\ Pr(y_i = 0|x_i, \theta) &= 1 - h_\theta(x_i) \end{aligned} \tag{5.2}$$

And

$$\begin{aligned} Pr(y_i^1, y_i^2, \ldots, y_i^R \ |y_i = 1, \alpha) &= \prod_{i=1}^{N} Pr(y_i^j|y_i = 1, \alpha^j) := a_i \\ Pr(y_i^1, y_i^2, \ldots, y_i^R \ |y_i = 0, \beta) &= \prod_{i=1}^{N} Pr(y_i^j|y_i = 0, \beta^j) := b_i \end{aligned} \tag{5.3}$$

So, we can rewrite $L(\theta) = \prod_{i=1}^{N}(a_i h_\theta(x_i) + b_i(1 - h_\theta(x_i)))$.

Generally, we are trying to maximize the log-likelihood function, $\mathcal{L}(\theta) = log(L(\theta))$, to avoid potential overflow. If we know the true labels $\mathbf{y} = (y_1, y_2, \ldots, y_N)$, then the log-likelihood function is:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} [y_i log(a_i h_\theta(x_i)) + (1 - y_i) log(b_i(1 - h_\theta(x_i)))] \tag{5.4}$$

However, since the true labels are missing, we use $\mu_i$, the predicted label from MLE, to replace $y_i$ in above formula. So we get

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} [\mu_i log(a_i h_\theta(x_i)) + (1 - \mu_i) log(b_i(1 - h_\theta(x_i)))] \tag{5.5}$$

Where

$$\mu_i = Pr(y_i = 1|y_i^1, y_i^2, \ldots, y_i^R, x_i, \theta) = \frac{a_i h_\theta(x_i)}{a_i h_\theta(x_i) + b_i(1 - h_\theta(x_i))} \tag{5.6}$$

Again, the $h_\theta(x_i)$ is the weak prediction from logistic regression.

Since zero ground-truth labels are provided, we use the results from majority voting as the initial value of $\mu$ to start the MLE.

### 5.1.2 Logistic Regression

LR is a gradient descent method that gradually moves from current position towards to the local optimum by steps proportional to the negative of the gradient. In Raykar et al. (2010), the authors update the LR model by Newton-Raphson formula that calculates the gradient vector and the Hessian matrix. Theoretically, it works for low-dimension matrices (Raykar et al. (2010) proposes a 27-dimension feature vector for breast MRI data). We use the stochastic gradient descent method in Cormack et al. (2011), as shown in Equation 5.7, for its computational efficiency.

$$w = w + \delta * (y_i - p_i) * x_i \tag{5.7}$$

$$p_i = h_\theta(x_i) = \frac{1}{1 + e^{-w * x_i}} \tag{5.8}$$

$p_i$, in the range [0,1], is the score from logistic regression classifier to indicate the probability that $i^{th}$ document is relevant.

Since the true label $y_i$ is missing, we actually use the weakly predicted score from MLE, $\mu_i$, to replace $y_i$. So, Equation 5.7 can be rewritten as:

$$w = w + \delta * (\mu_i - p_i) * x_i \tag{5.9}$$

The initial values of LR ($w$) and the step length ($\delta$) are two essential parameters to start the algorithm. We need to select the proper $\delta$ and the initial value of model to avoid the algorithm may reach a not-so-promised local optimum. The suggested step size in Cormack et al. (2011) is 0.002. We use a slightly smaller value, 0.001, as step size. As to the initial value of model, if a subset of documents' true labels are given, we can use these documents to train the logistic regression classifier first, and then use the trained model to initialize $w$. Since no true labels are available, we directly set $w = \vec{0}$. After initialization, the weight vector $w$ keeps updated every time the algorithm processes one document.

## 5.2 Experiment Settings

### 5.2.1 Algorithms to Compare

Here, we compare the content-based consensus algorithm (referred as EM-content) against other three algorithms: majority voting with *MajorClass* (still referred as majority voting

afterwards in this chapter), the EM algorithm without considering content (referred as "EM-no-content" afterwards) from Raykar et al. (2010), and the iterative algorithm with signal detection measurement (Smucker, 2012) (referred as "SDCon" afterwards). We choose *MajorClass* as the strategy to break ties in majority voting since it works well and does not need any prior knowledge about the statistics of dataset, as discussed in Chapter 4. The SDCon algorithm achieves very good performance in the TREC2011 Crowdsourcing Track. The content-based algorithm is consistent with EM-no-content algorithm in the way that they both use sensitivity and specificity to model each assessor's quality. So, we are interested in how much improvement we can gain by adding content.

As mentioned before, EM-content algorithm has two perspectives of measuring the relevance of document: one from MLE, denoted as mu; another one is from LR, denoted as pi. Which prediction should we trust? We believe that both mu and pi are good measurements of relevance. The explanation is that the iterative process stops when convergence reaches, which means, theoretically, both types of classifiers reach an agreement on the documents' labels. To fully understand how these two behave in practice, we use both as outputs for EM-content algorithm. So we have two variants of EM-content algorithm: EM-content-mu and EM-content-pi.

In sum, we have 5 algorithms to test: majority voting, EM-no-content, SDCon, EM-content-mu, and EM-content-pi.

We use the same simulated judgments in Chapter 4. Only 0.5 is used as the threshold of relevance since the effects of varying decision rule have already been investigated in Chapter 4.

### 5.2.2 Feature Extraction

The feature extraction method is the same as the 4-grams in Cormack et al. (2011). We simply treat overlapping sequences of 4 characters as tokens. For example, the string "ab cde" would be split into the following 4-grams: "ab c", "b cd", and " cde". To achieve it, we use a sliding window with the width of 4 characters to scan the document. Each document is treated as flat text. Document exceeding 35000 bytes in length is arbitrarily truncated to this length. Each feature is represented as a binary number indicating its presence or absence in this document. Finally, the feature space is reduced from $4 * 10^9$ to $10^6$ dimensions using hashing and ignores collision. All features are saved in sparse format. Essentially, 4-gram is language-independent, no language-specific tokenization, parsing, or link analysis is done.

## 5.3 Evaluation

We conduct both relevance evaluation and ranking system evaluation in this chapter. Relevance evaluation metrics include Logistic Average Misclassification Rate (LAM) (Cormack and Lynam, 2005), Area Under Curve (AUC). APCorr is used for IR evaluation.

LAM is defined as:

$$LAM = logit^{-1}(\frac{logit(fpr) + logit(fnr)}{2}) \tag{5.10}$$

Where $fpr$ is false positive rate and $fnr$ is false negative rate, and

$$\begin{aligned} logit(p) &= \log \frac{p}{1-p} \\ logit^{-1}(x) &= \frac{e^x}{1+e^x} \end{aligned} \tag{5.11}$$

Meanwhile, we use the same way in Smucker et al. (2013) to smooth the $fpr$ and $fnr$ based on the prevalence:

$$\begin{aligned} fnr &= \frac{fn + 0.5 * numRels/numDocs}{fn + tp + numRels/numDocs} \\ fpr &= \frac{fp + 0.5 * numRels/numDocs}{fp + tn + numRels/numDocs} \end{aligned} \tag{5.12}$$

Where $numRels$ means the number of relevant documents in this topic and $numDocs$ is the total number of documents. As we can see, LAM value is in the range [0, 1]. The lower value of LAM, the better the classifier is, and vice versa.

Besides LAM, the AUC (Area Under Curve) of each ROC curve is adopted as another metric for models comparison. One advantage of AUC against accuracy is that we don't need to set the threshold of relevance since choosing the theoretically optimal threshold value is subjective and the threshold may vary from different datasets and topics. Instead, we can bypass this issue by generating ROC and AUC based on score, a value between 0 and 1, since no predefined threshold is needed.

## 5.4 Implementation Details

### 5.4.1 Avoid Overflow

Overflows may happen when calculating $a_i$ and $b_i$ in Equation 5.3 since it involves the multiplication of multiple probabilities, though we have already assumed $0 \log 0 = 0$. When at least one of $a_i$ and $b_i$ equals to 0 or 1 due to undesirable overflow, the log-likelihood value equals to negative infinity. We cannot just simply convert it into log-scale and then substitute the inverse-log of $a_i$ and $b_i$ to log-likelihood function. Instead, we introduce a penalty method to relieve the overflow issue. Every time overflow happens, we add (or minus) a very small value (penalty), if $a_i$ or $b_i$ equals to 0 (or 1).

Basically, any reasonable small number should be OK for penalty. In our experiment, penalty equals to 0.000001. After the penalty incorporated into our implementation, the log-likelihood keeps non-decreasing from iteration to iteration. This matches the theoretical analysis of EM algorithm.

### 5.4.2 Convergence Condition of EM Algorithms

If the difference between two iterations log-likelihood values is smaller than the predefined threshold, we think EM algorithm reaches convergence and should output this iteration results. However, since it's hard to know the optimal threshold, we vary the threshold of convergence for EM-no-content and EM-content to check how it affects the evaluation. We don't vary the threshold of convergence for SDCon since SDCon converges very quickly in less than 10 iterations in our experiment.

The candidate thresholds for convergence are: {1, 10, 20, 30, 40, 50}.

### 5.4.3 Aggregate The Results

The way to average the results is the same as in Chapter 3 and 4. We calculate each simulated run's results first then average across the runs.

## 5.5 Results and Discussions

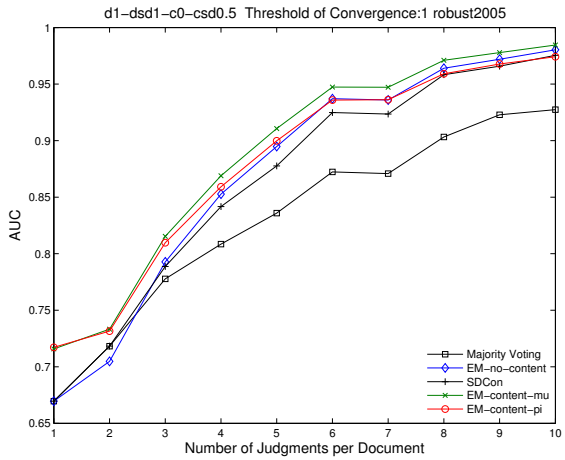
### 5.5.1 Relevance Evaluation

#### Varying The Convergence Condition

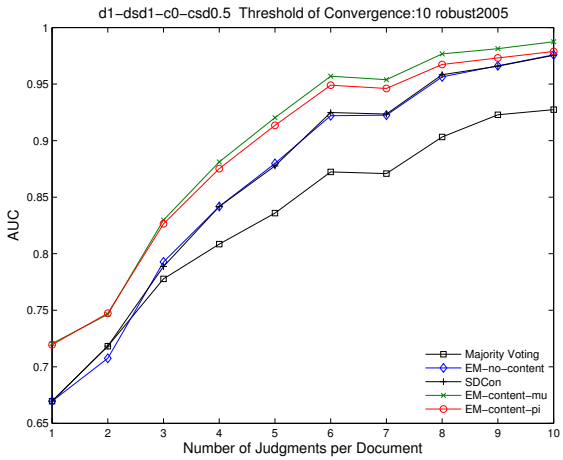
We use the AUC and LAM results of $d1$–$dsd1$–$c0$–$csd0.5$ on Robust2005 as example to show how the convergence condition affects the relevance evaluation results, as shown in Figures 5.1 and 5.2.

As we can see, the proposed content-based EM algorithm outperforms the other 3 algorithms in all convergence conditions. It shows the benefit of leveraging the contents of documents in relevance evaluation on Robust2005. Each subfigure in Figure 5.3 summaries individual algorithm's performance when varying convergence condition. The best threshold of likelihood difference for EM-content algorithm and EM-no-content algorithm are 10 and 1, respectively. Essentially, the larger the threshold of log-likelihood difference, the more relaxed convergence condition is. This is because we immediately output the results once the convergence condition is satisfied. In other word, the EM algorithm stops earlier, if the threshold is larger. Based on the results, as the convergence condition becomes looser (the threshold increases), both AUC and LAM results of EM algorithms become worse. So, we need to make sure both EM algorithms run reasonably sufficient number of iterations to achieve satisfied relevance evaluation results.

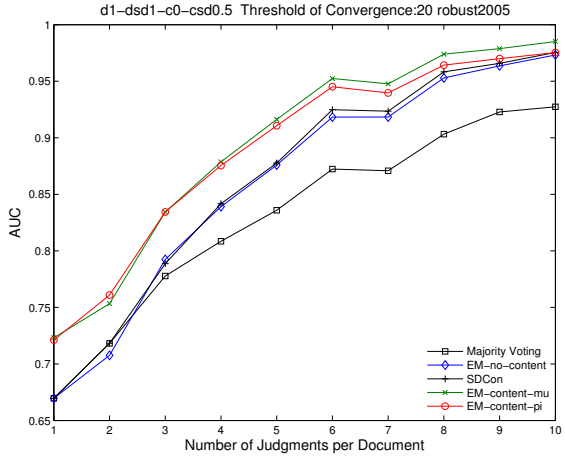
If we compare the performance between EM-content-mu and EM-content-pi, using mu as output is more preferable for relevance evaluation, though these two's performances are very close to each other.



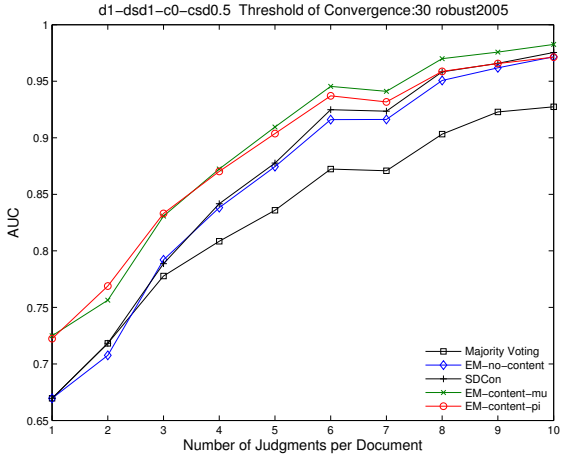

(a) Threshold of Convergence = 1



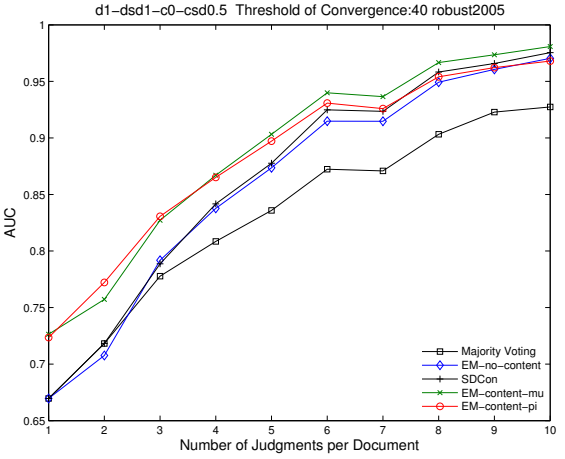

(b) Threshold of Convergence = 10



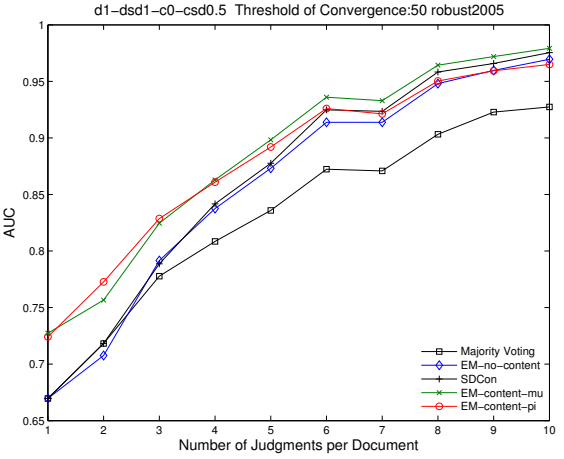

(c) Threshold of Convergence = 20




(d) Threshold of Convergence = 30

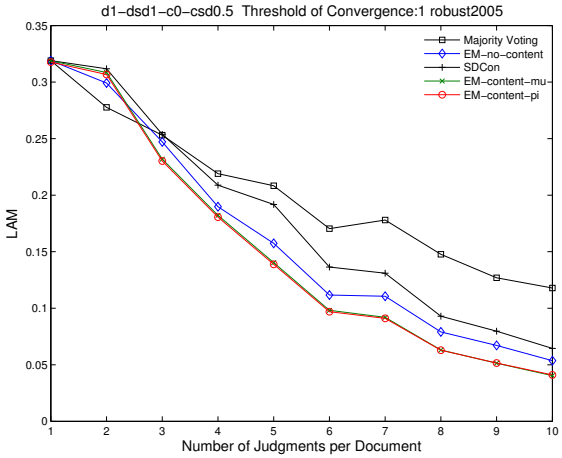



(e) Threshold of Convergence = 40

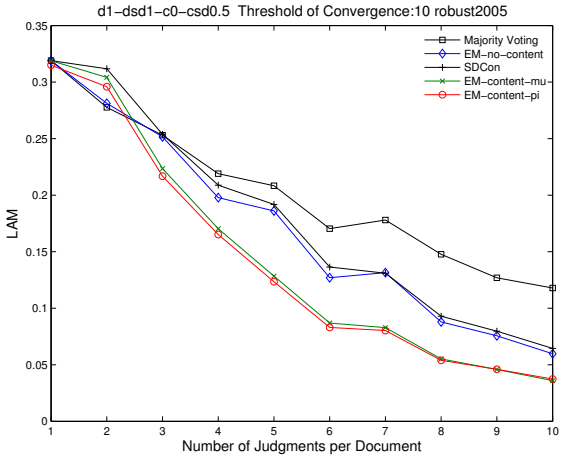



(f) Threshold of Convergence = 50

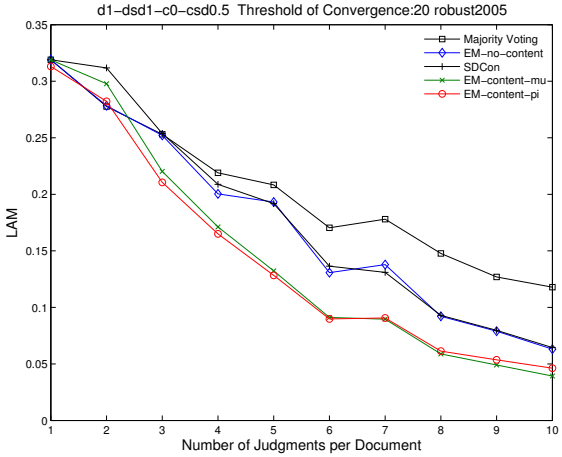
Figure 5.1: The AUC results of varying the convergence condition of EM algorithms on Robust2005, using simulation setting $d1$–$dsd1$–$c0$–$csd0.5$



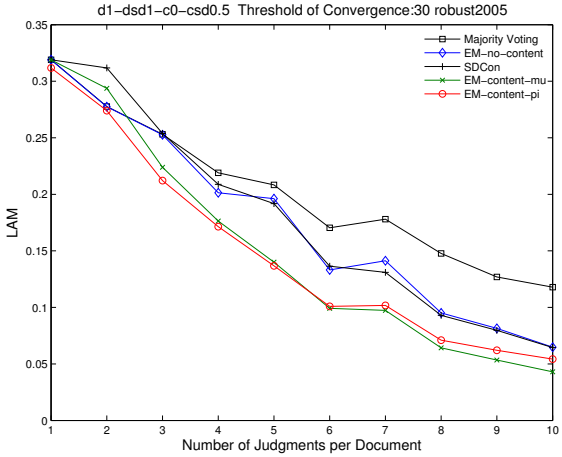

(a) Threshold of Convergence = 1



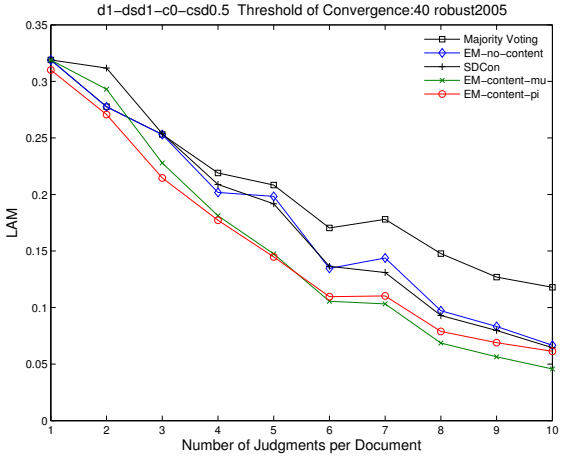

(b) Threshold of Convergence = 10



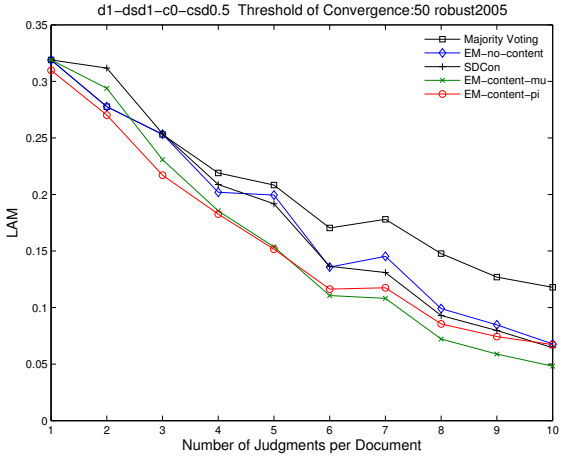

(c) Threshold of Convergence = 20




(d) Threshold of Convergence = 30

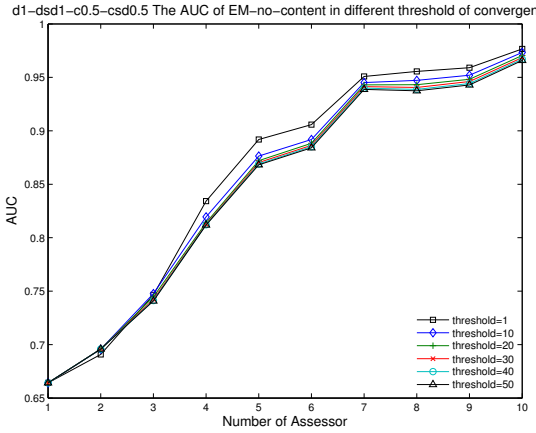



(e) Threshold of Convergence = 40

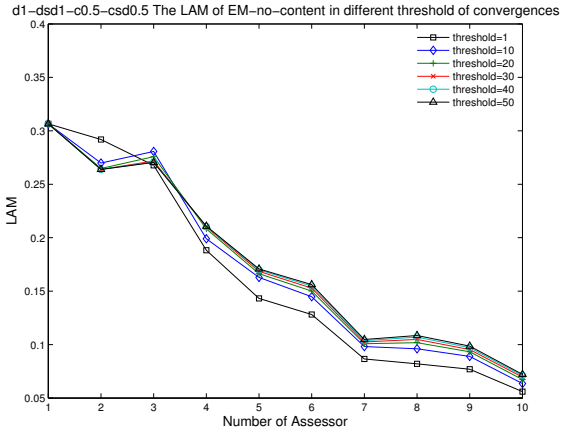



(f) Threshold of Convergence = 50

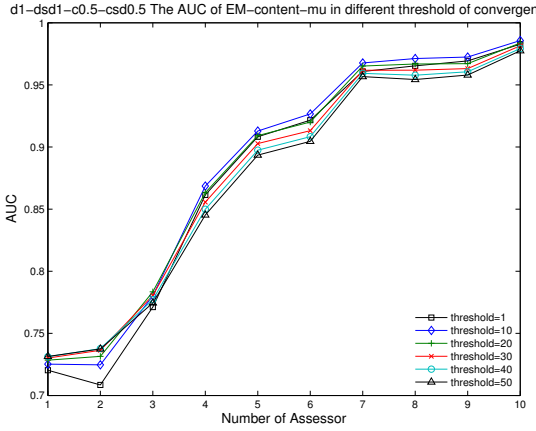
Figure 5.2: The LAM results of varying the convergence condition of EM algorithms on Robust2005, using simulation setting $d1$–$dsd1$–$c0$–$csd0.5$



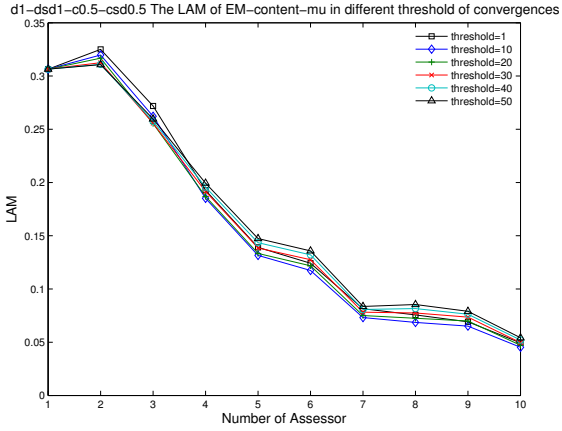

(a) AUC of EM-no-Content Algorithm



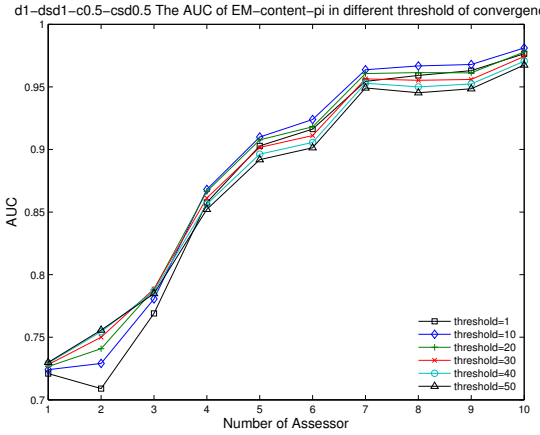

(b) LAM of EM-no-Content Algorithm



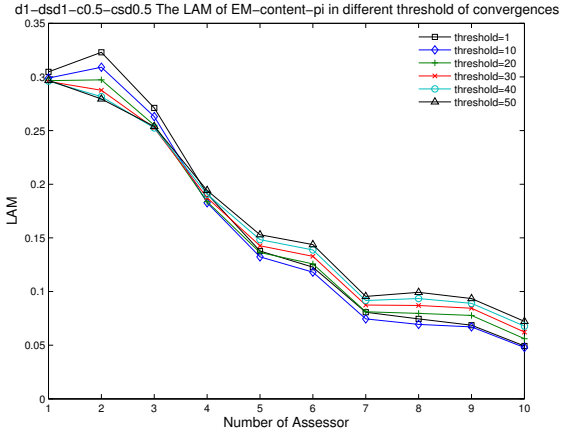

(c) AUC of EM-Content-mu Algorithm




(d) LAM of EM-Content-mu Algorithm




(e) AUC of EM-Content-pi Algorithm




(f) LAM of EM-Content-pi Algorithm

Figure 5.3: Each plot shows the AUC and LAM results for same EM algorithm when varying convergence condition on Robust2005, using simulation setting $d1$–$dsd1$–$c0$–$csd0.5$

### Comparison Between Consensus Algorithms

In this section, we compare consensus algorithms when each one is at its best performance, given assessors with multiple simulation settings. In this way, we can understand whether the EM-content algorithm consistently performs well under various cases.

Similar to previous section, we show the results of $d = \{1, 2\}$, $dsd = 0.5$, $c = \{-0.5, 0, 0.5\}$, $csd = 0.5$. These simulation settings cover all types of assessors we need as well as keep the variations among assessors. The AUC and LAM results are shown in Figures 5.4 and 5.5, respectively. In each subfigure, each EM algorithm's optimal threshold is shown in the legend. For example, EM-content-mu-10 means the EM-content-mu algorithm achieves its best performance when the threshold of convergence equals to 10.

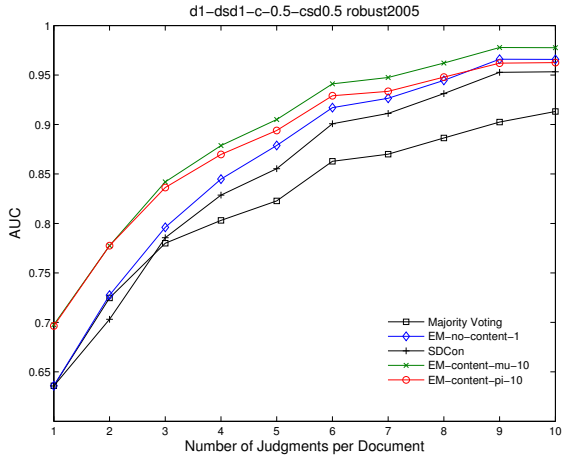
As to each algorithm of EM-no-content and EM-content algorithms, the optimal threshold is the one with the largest sum of AUC/APCorr, or smallest sum of LAM.

We can have the following observations.

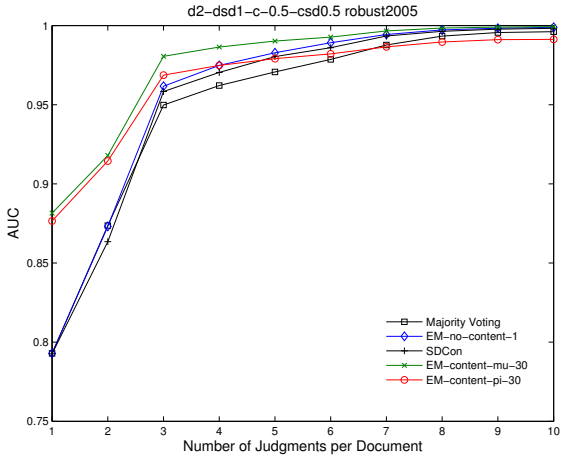
In all simulation cases, EM-content algorithm outperforms the rest algorithm. Meanwhile, mu is still more suitable than pi as output for relevance evaluation.

Secondlly, even though occasionally more assessors don't improve both AUC and LAM results, we can still conclude that almost all algorithms' performances tend to be better as the number of assessors increases. Intuitively, more assessors reduce the chances that spammers' judgments dominate the decision-making process and relieve the effects of the potential errors made by ethical assessors. So, if budget permitted, it's better hiring more assessors even though each assessor's judging ability is weak.

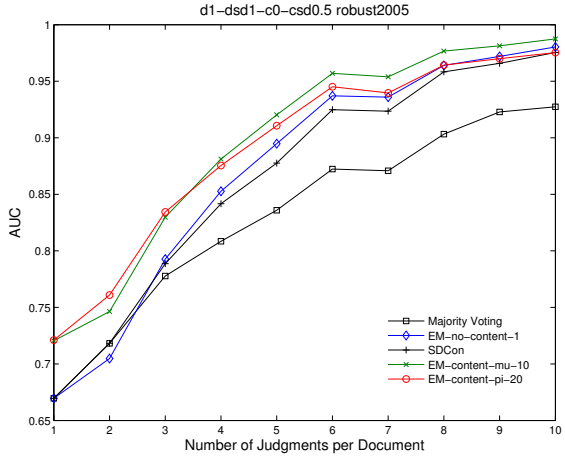
EM-content algorithm has rapid performance improvement when the number of assessor is less than 5. When $n \geq 6$, the increment rate tends to be low. When we have a few assessors, it's strongly suggested to use content of the documents. Essentially, we can treat the content as another hidden and useful "assessor" in the way that this kind of information is very objective.



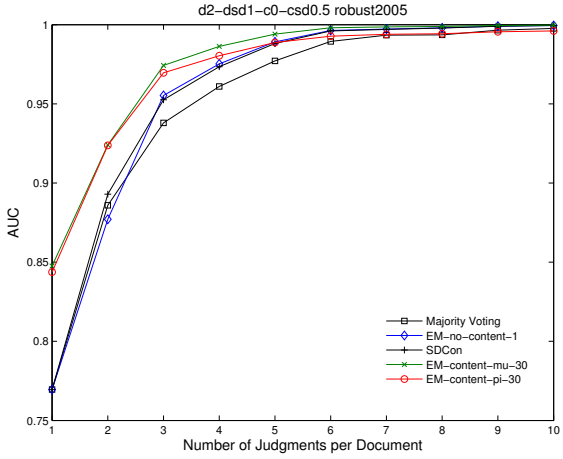

(a) $d' = 1, c = -0.5$



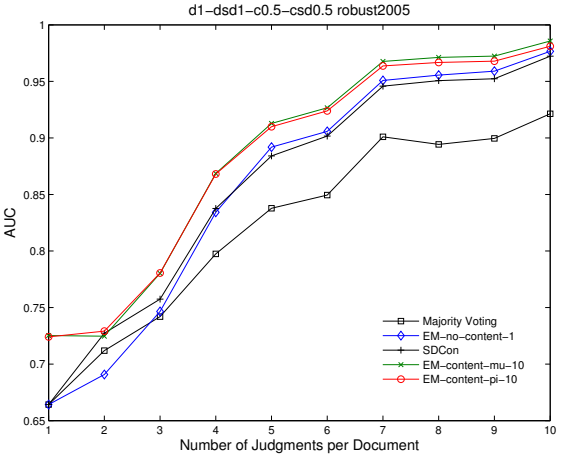

(b) $d' = 2, c = -0.5$



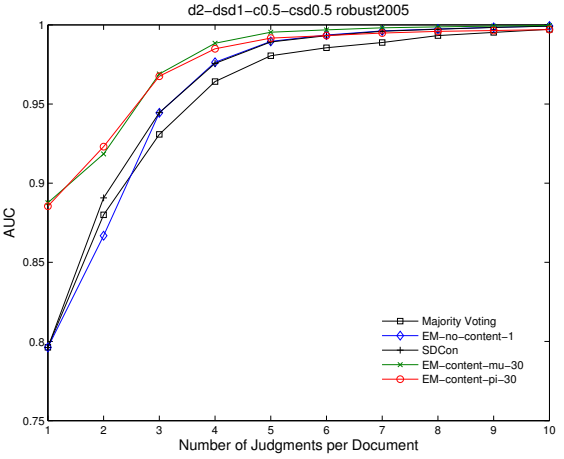

(c) $d' = 1, c = 0$




(d) $d' = 2, c = 0$

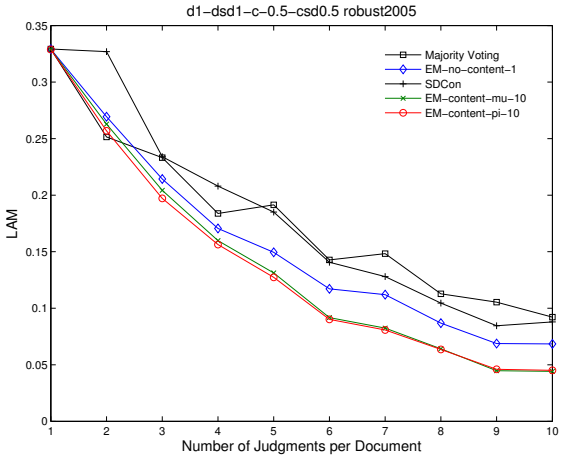



(e) $d' = 1, c = 0.5$

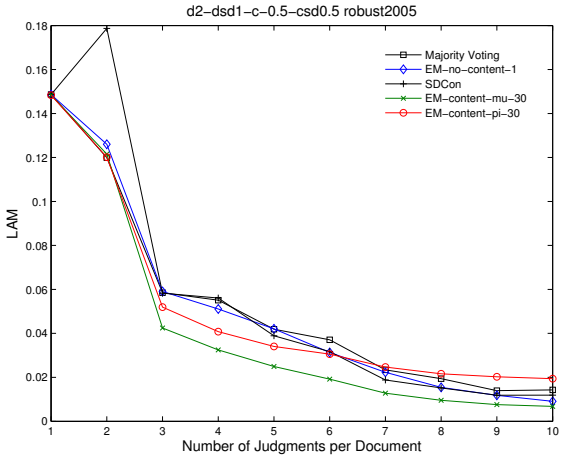



(f) $d' = 2, c = 0.5$

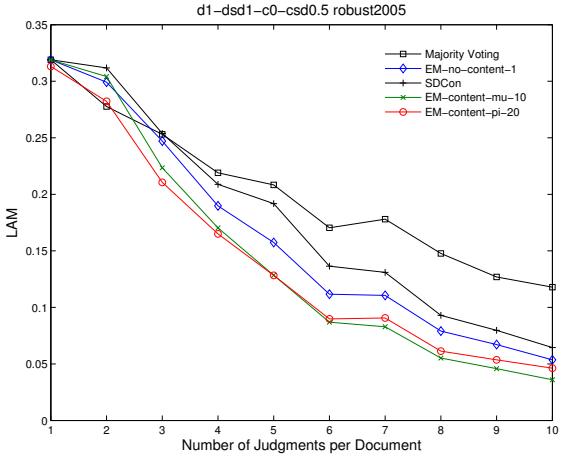
Figure 5.4: The AUC results of comparing consensus algorithms under different simulation settings on Robust2005



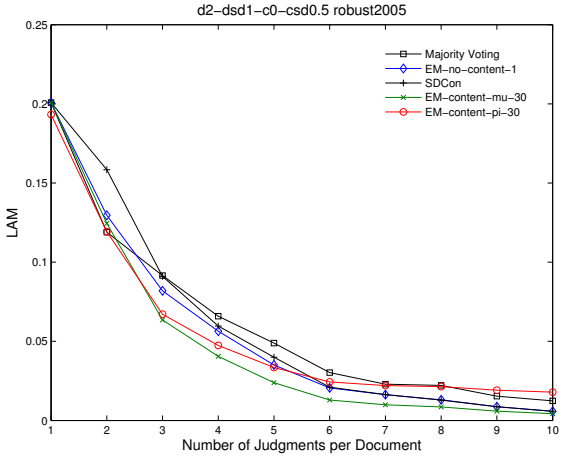

(a) $d' = 1, c = -0.5$



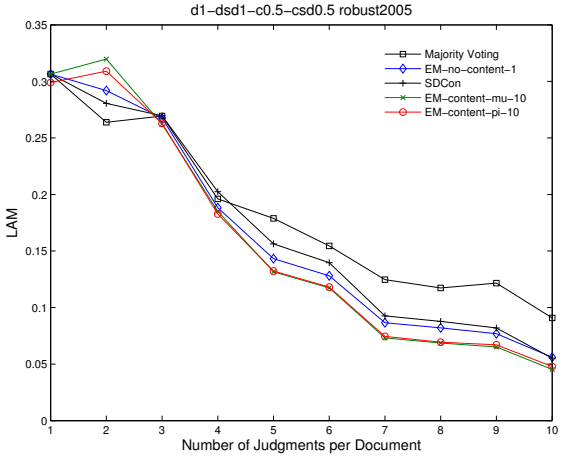

(b) $d' = 2, c = -0.5$



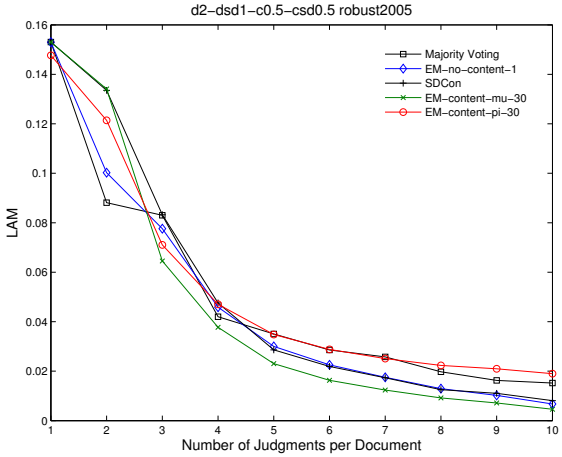

(c) $d' = 1, c = 0$




(d) $d' = 2, c = 0$




(e) $d' = 1, c = 0.5$




(f) $d' = 2, c = 0.5$

Figure 5.5: The LAM results of comparing consensus algorithms under different simulation settings on Robust2005

### 5.5.2 Ranking System Evaluation

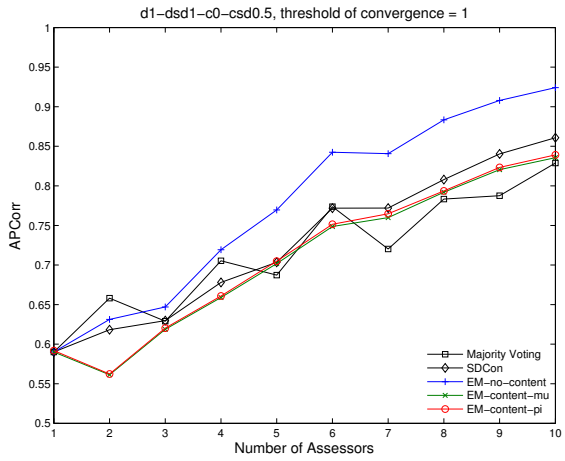
#### Varying The Threshold of Likelihood

Figure 5.6 shows how different convergence conditions affect the IR evaluation on Robust2005, using $d1$–$dsd1$–$c0$–$csd0.5$. Now, the behavior of EM content algorithm is different from that in relevance evaluation. The larger threshold, the better APCorr EM-content algorithm can achieve. EM-no-content algorithm still achieves its best correlation when it has a very strict convergence condition.

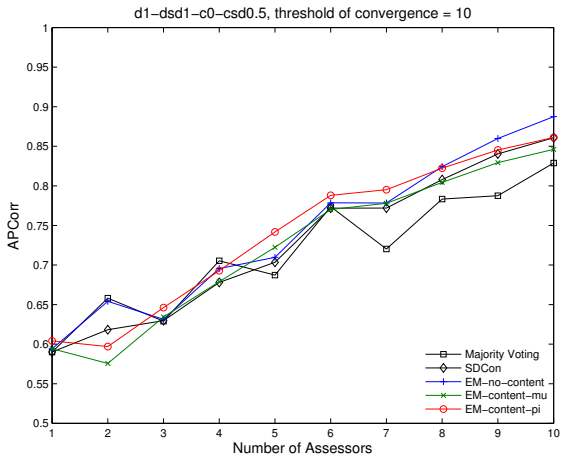
To figure out the reason for that, we calculate the $d'$ and $c$ of each algorithm under different convergence conditions, as shown in Figure 5.7 and 5.8. Essentially, from the criterion figures we can see that EM-content algorithm becomes more conservative as the convergence threshold increases. However, EM-no-content algorithm becomes more liberal. Even though both algorithm's $d'$ decreases as thresholds increases, when the $d'$ is at reasonable range, like $> 1.5$, algorithms' criterion plays a more important role to the correlation than $d'$. So, EM-content algorithm is more preferable if the iterative process stops earlier.

Majority voting seems to be more conservative compared with other algorithms since *MajorClass* is used to deal with ties. However, majority voting's $d'$ is very low compared with others. It makes the majority voting does not perform that well in all cases.

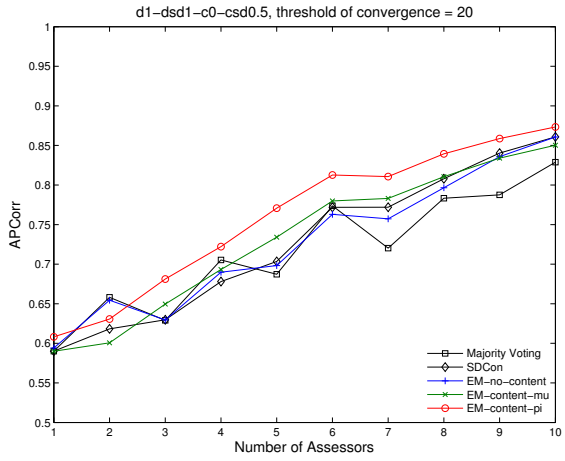
The best correlation we can achieve now is the EM-no-content algorithm when the threshold of convergence equals to 1. This conclusion can be also observed in Figure 5.9(a), where all consensus algorithms at their performances are compared. It means even though we leverage the document contents, and it works when thresholds of convergence are larger than 1, the best performance we can achive is EM-no-content algorithm when threshold equals to 1.



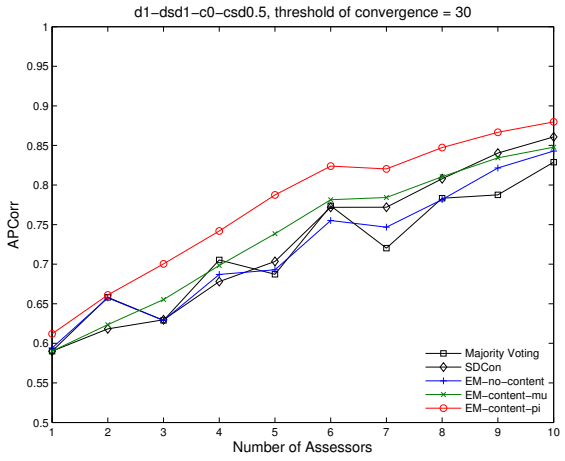

(a) Threshold of Convergence = 1



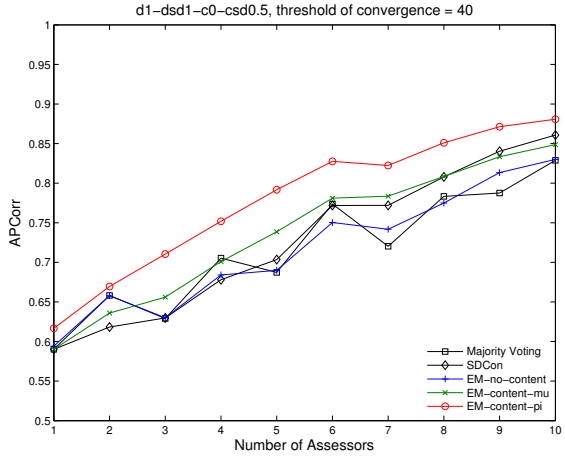

(b) Threshold of Convergence = 10



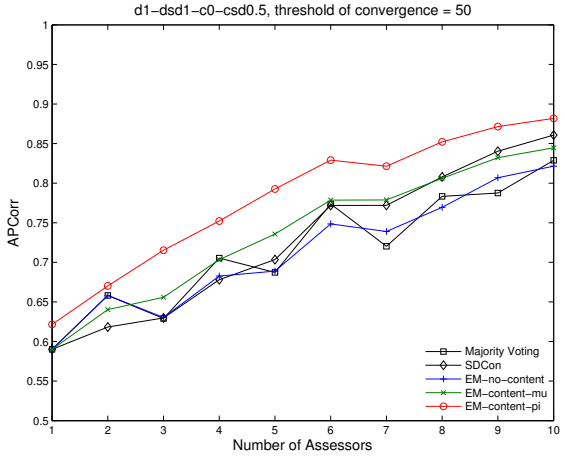

(c) Threshold of Convergence = 20




(d) Threshold of Convergence = 30

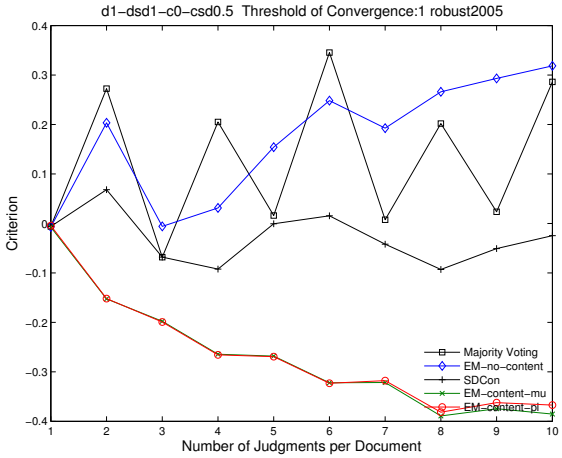



(e) Threshold of Convergence = 40

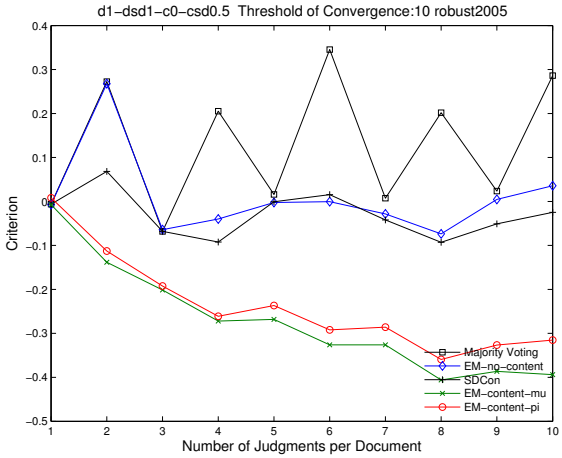



(f) Threshold of Convergence = 50

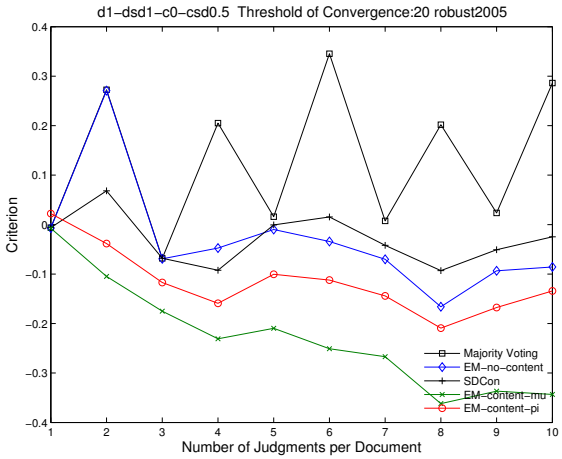
Figure 5.6: The APCorr results of varying the convergence condition of EM algorithms on Robust2005



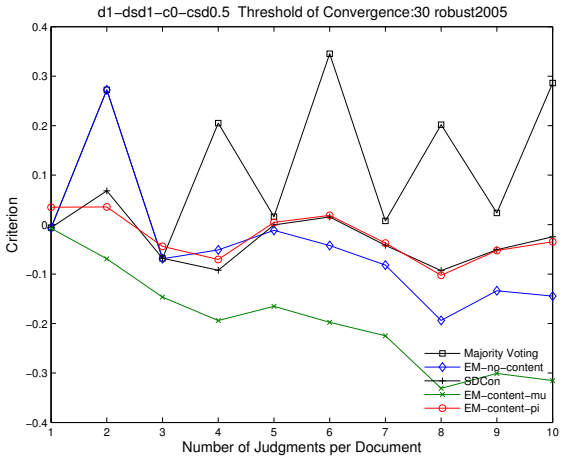

(a) Threshold of Convergence = 1



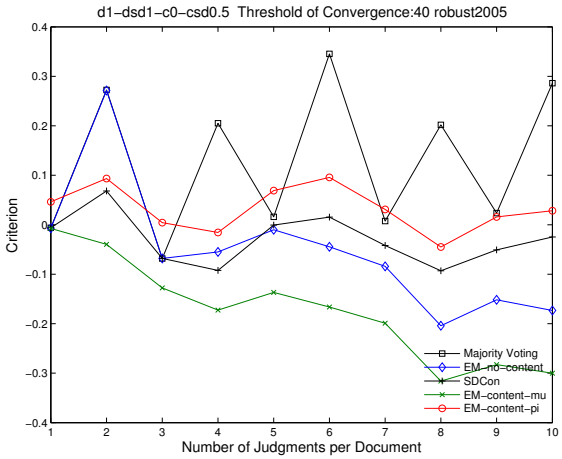

(b) Threshold of Convergence = 10



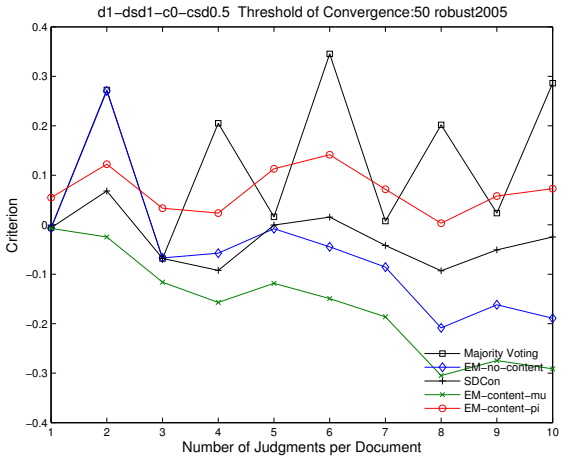

(c) Threshold of Convergence = 20




(d) Threshold of Convergence = 30

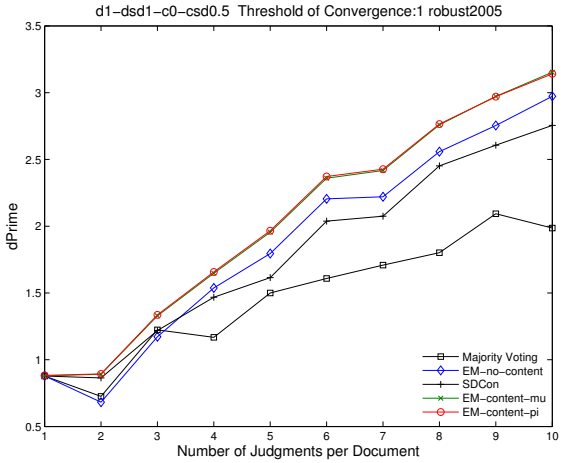



(e) Threshold of Convergence = 40

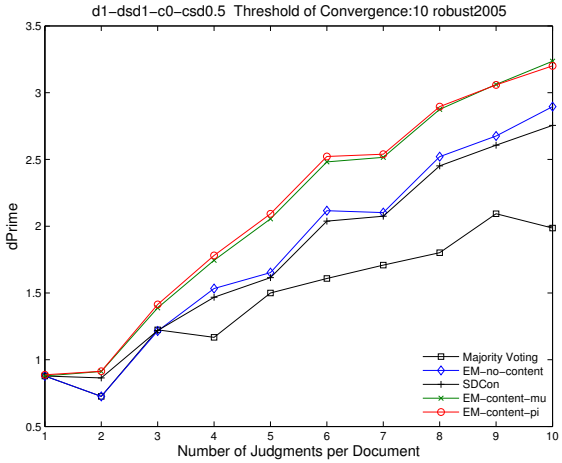



(f) Threshold of Convergence = 50

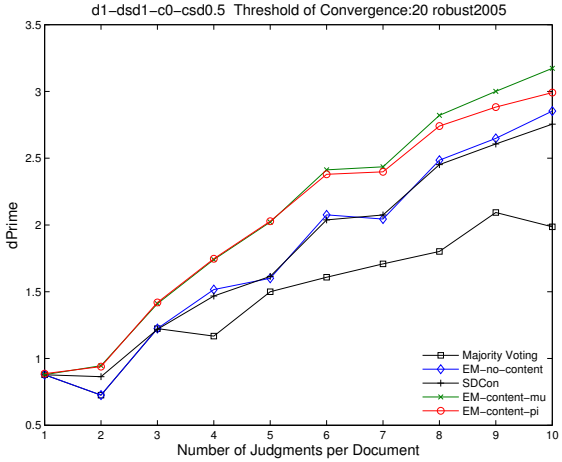
Figure 5.7: The criterion of varying the convergence condition of EM algorithms on Robust2005



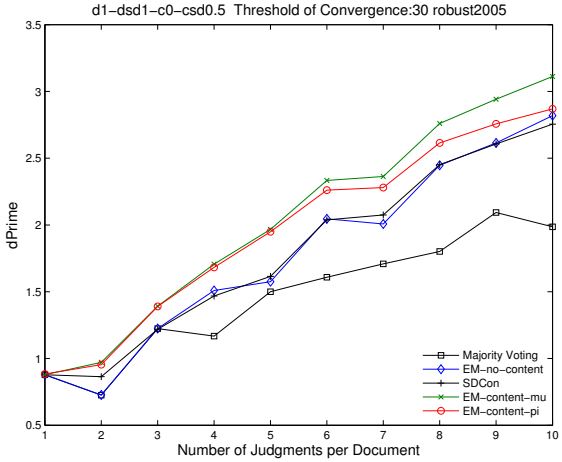

(a) Threshold of Convergence = 1



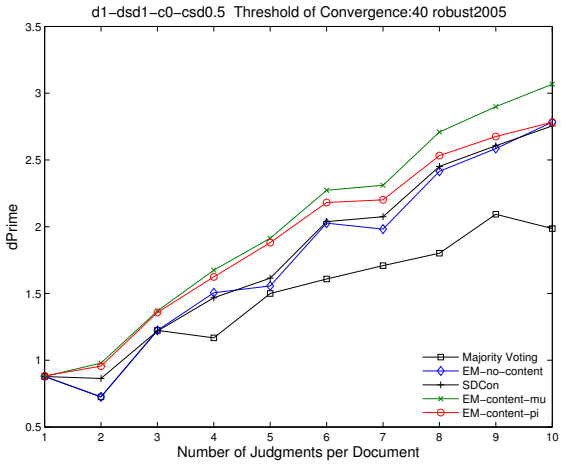

(b) Threshold of Convergence = 10



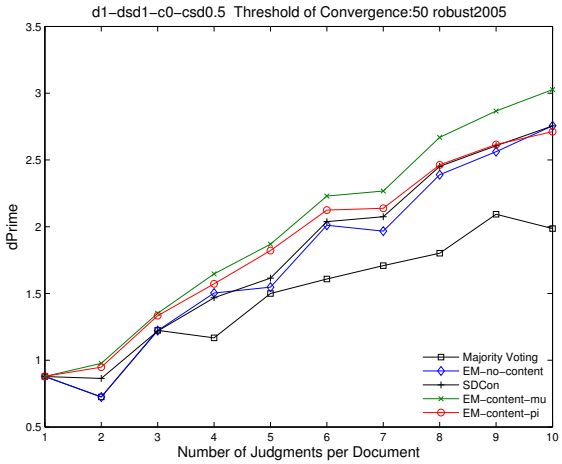

(c) Threshold of Convergence = 20




(d) Threshold of Convergence = 30




(e) Threshold of Convergence = 40




(f) Threshold of Convergence = 50

Figure 5.8: The $d'$ of varying the convergence condition of EM algorithms on Robust2005

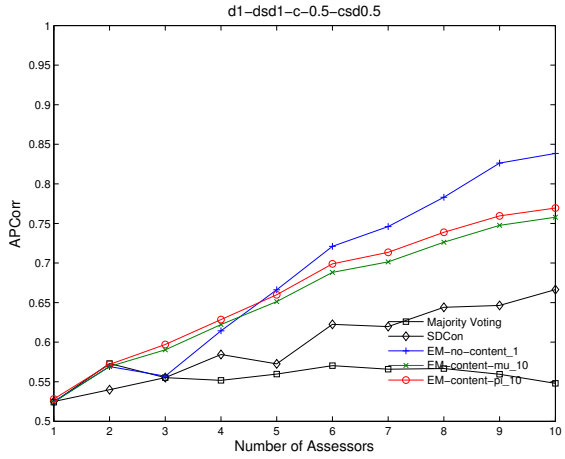
### Comparison Between Consensus Algorithms

Figure 5.9 shows the comparison results between consensus algorithms in terms of APCorr on Robust2005. Each algorithm is at their optimal performance across the thresholds of convergence.

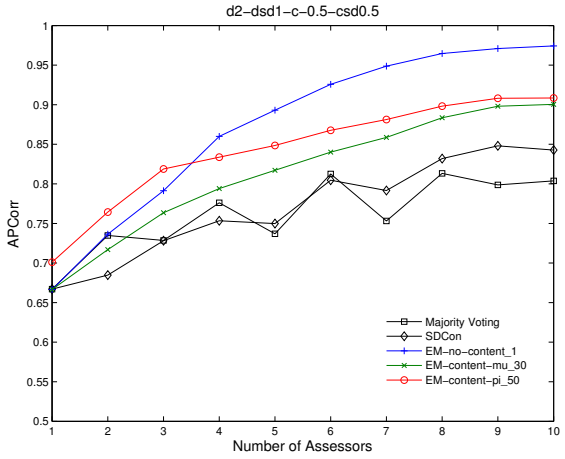
Firstly, we can see that if the number of assessors is larger than or equal to 5, EM-no-content with threshold of 1 outperforms the rest algorithms at their optimal performances. However, when $n$ is smaller than 5, using content is a good idea. However, when the $n$ is less than 5, it's very hard to achieve good correlation with the experts' judgments. Essentially, if we want to have over 0.9 correlation, the recommended strategy is to hire at least 8 assessors, plus EM-no-content algorithm to find consensus.

Moreover, we see that all figures with $d' = 2$ tends to have better correlation than those figures with $d' = 1$. This is simply because $d' = 2$ contains more accurate labels than $d' = 1$. Meanwhile, EM-content algorithm almost achieves the best performance when the noisy judgments are neutral or liberal. Essentially, EM-content algorithm tends to behave more conservative than other algorithms when given these two types of data.

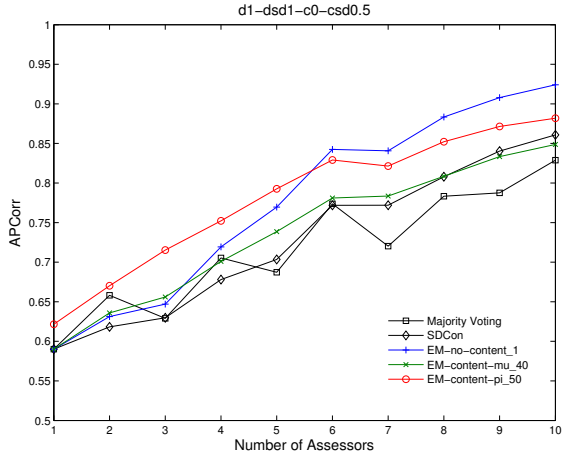
However, when the noisy labels are already very conservative ($c = 0.5$), in this case, we don't see too much improvement by using the content. However, when the number of assessors is smaller than 5, we can still see the content boosts the performance. Actually, when $c = 0.5$, we don't need very fancy or complicated consensus methods. Simple algorithms, like SDCon and EM-no-content, even Majority voting at $d' = 2$, can result in very good correlation. If we fix $d'$ and compare how the algorithms perform when varying the criterion, we can have a clear clue that the higher criterion is, the better correlation all algorithms can get. Now, it's hard to decide whether we should choose mu or pi as output for EM-content algorithm. pi tends to be better when the collected judgments are not that conservative but worse when the judgments are already conservative enough.



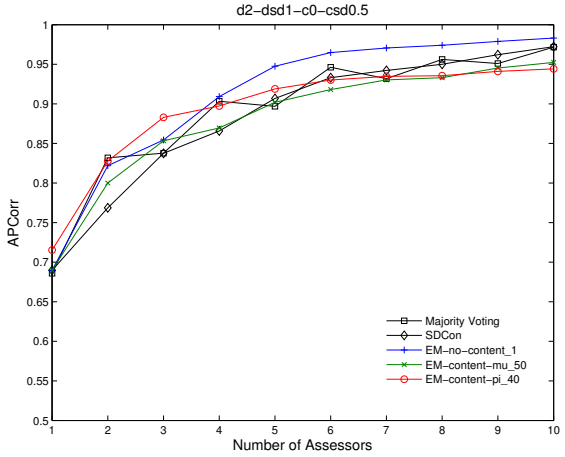

(a) $d' = 1, c = -0.5$



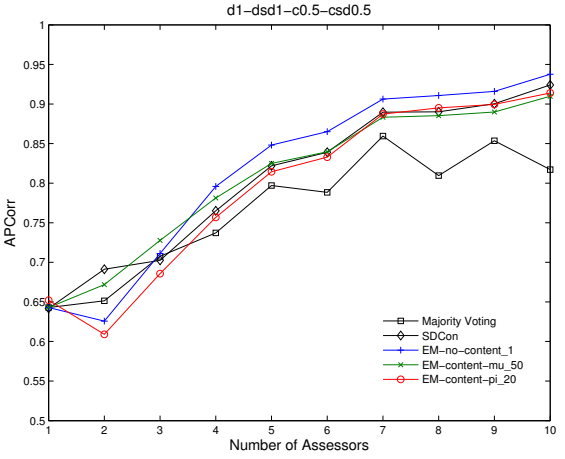

(b) $d' = 2, c = -0.5$



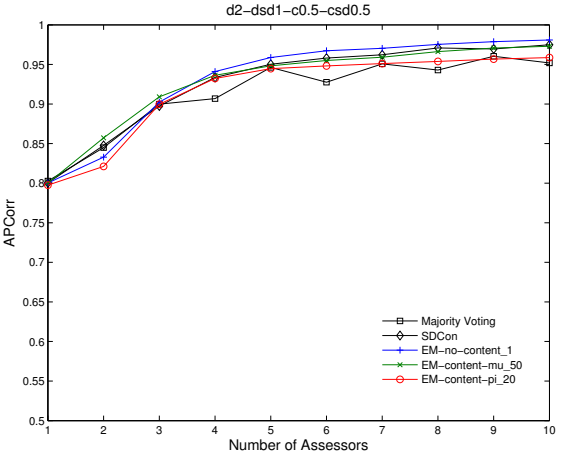

(c) $d' = 1, c = 0$




(d) $d' = 2, c = 0$




(e) $d' = 1, c = 0.5$




(f) $d' = 2, c = 0.5$

Figure 5.9: The APCorr results of comparing consensus algorithms under different simulation settings on Robust2005

### 5.5.3 Limitations of Content-based Consensus Algorithm

When running all algorithms on Trec8 dataset, we see the limitations of the content-based algorithm.

Figure 5.10 and 5.11 show the AUC and LAM results of varying convergence conditions with $d1$–$dsd1$–$c0$–$csd0.5$ on Trec8 dataset. Figure 5.12 and 5.13 show the comparison based on the various simulation settings in terms of AUC and LAM. If the threshold of convergence is not equal to 1, the EM-content-mu still outperforms the rest algorithms. However, EM-content-pi algorithm has the worst performance, sometimes even worse than Majority Voting.

When it comes to the correlation test, as shown in Figures 5.14 and 5.15, both variants of EM-content algorithms perform poorly.

There are two reasons for it.

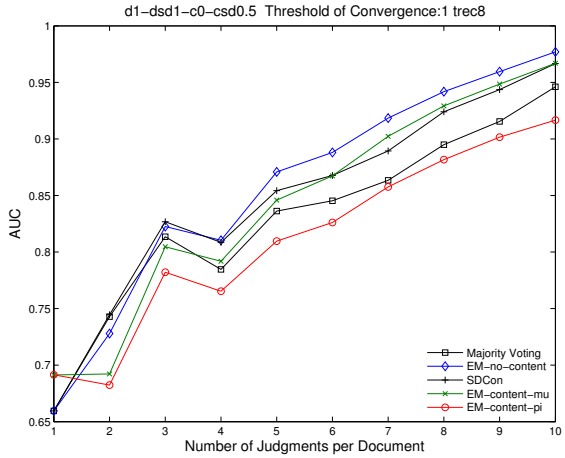
Firstly, Trec8 is more difficult to judge than Robust2005. We have two observations to support this point:

- Trec8 is essentially an imbalanced dataset and the relevant documents only occupy a small portion in each topic. Table 3.1 shows the statistics of Robust2005 and Trec8. The average percentage of relevant document per topic in Trec8 is less than half of Robust2005. However, the average number of documents per topic is at least as double as Robust2005. A false positive in Trec8 results in worse impact on the correlation than that on Robust2005. The unequal class distribution makes many classification algorithms biased towards the majority class (non-relevant) of documents, which increases the difficulty of finding out the correct labels.

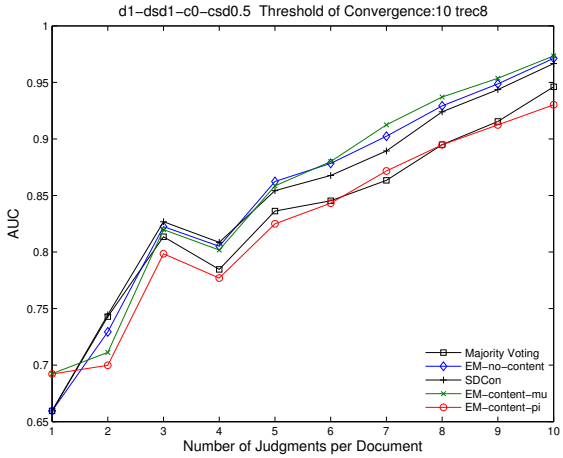
- If we only look at those consensus algorithms without utilizing document contents, i.e. majority voting, SDCon and EM-no-content, their APCorr results on Trec8 are worse than Robust2005. This happens in both varying threshold of convergence experiment (Figures 5.6 and 5.14), and the experiment with various simulation settings (Figures 5.9 and 5.15).

Secondly, the way we do logistics regression in this thesis is improper for imbalanced dataset, as it assumes same cost of errors. Essentially, a false positive is more harmful than a false negative in IR evaluation. So, without proper ways of handling imbalanced dataset, the labels generated from LR is inaccurate and untrustable. These incorrect labels meanwhile downgrade the performance of MLE. Figure 5.16 and 5.17 show the

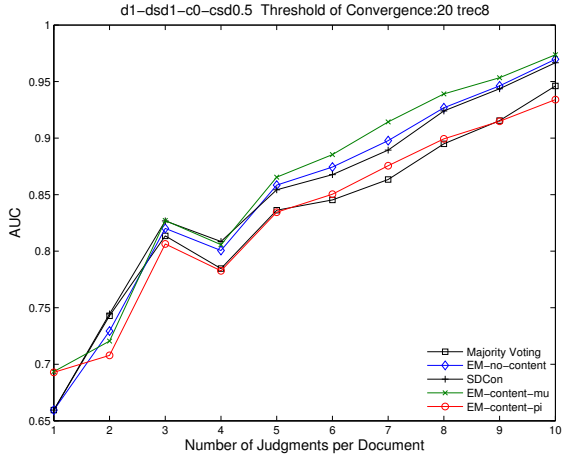
criterion and $d'$ of EM-content algorithm on Trec8. The discriminative ability of EM-content algorithm is even worse than majority voting. Extracting features without any pre-processing for the documents, e.g. parsing and tokenization could be another issue. So, we need some special techniques, like cost-sensitive learning, to deal with the asymmetric costs of misclassification in unbalanced dataset. Essentially, the co-training framework requires both views of data have reasonable accuracies. If one view generates many wrong labels, those wrong labels essentially downgrade the accuracy of the other view, as each one relies on the outputs of the other side as the pseudo truth. In our case, the logistic regression seems to be the main cause of the issue. It is recommended to make sure each view of co-training works on the dataset individually, before we put them together to find consensus.



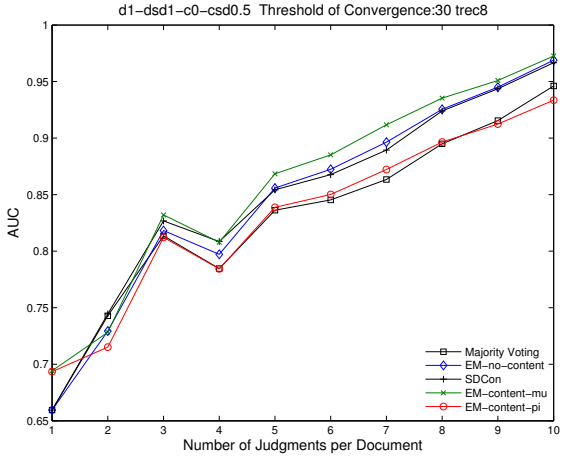

(a) Threshold of Convergence = 1



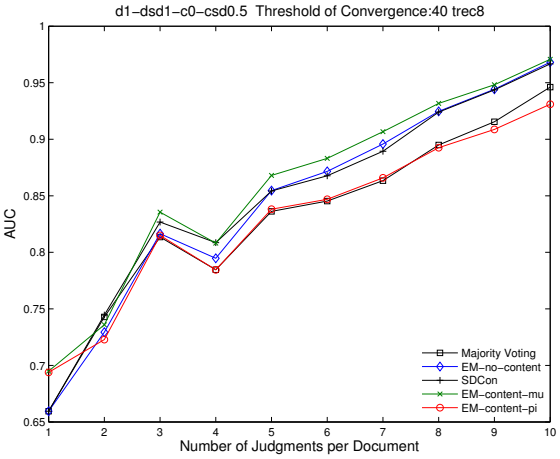

(b) Threshold of Convergence = 10



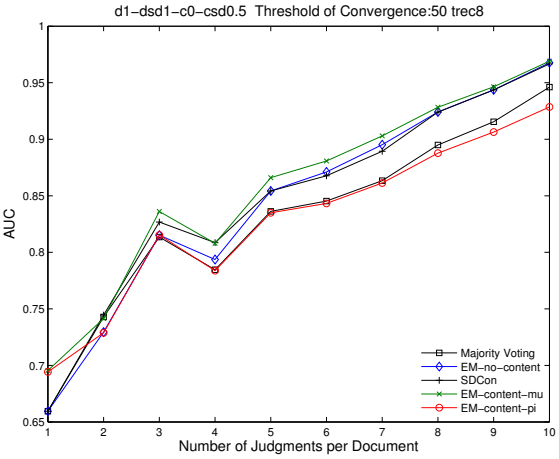

(c) Threshold of Convergence = 20




(d) Threshold of Convergence = 30

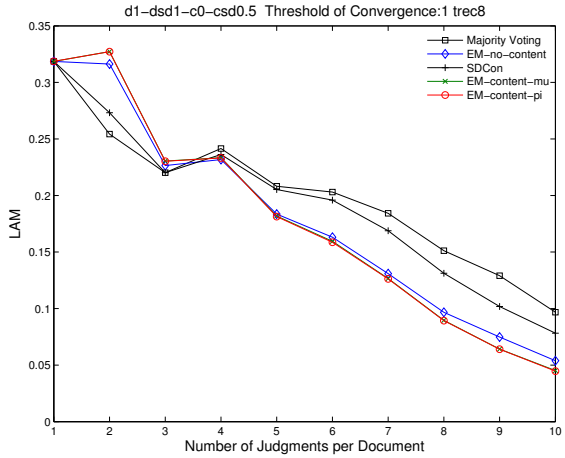



(e) Threshold of Convergence = 40

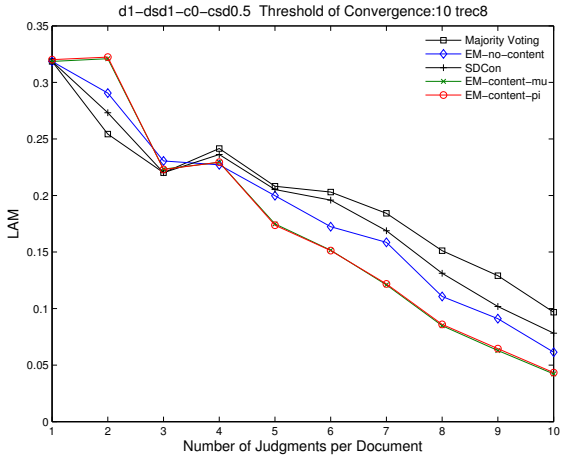



(f) Threshold of Convergence = 50

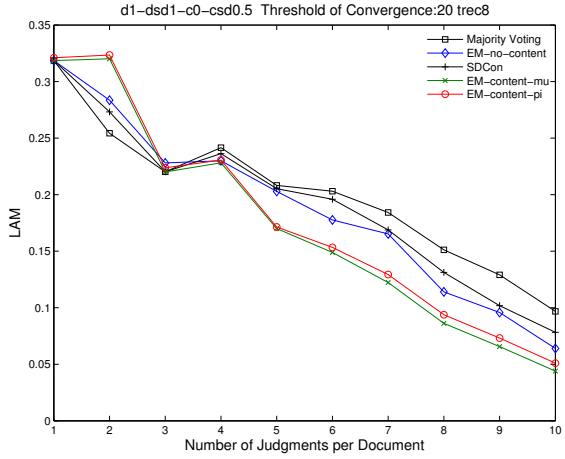
Figure 5.10: The AUC results of varying the convergence condition of EM algorithms on Trec8, using simulation setting $d1$–$dsd1$–$c0$–$csd0.5$



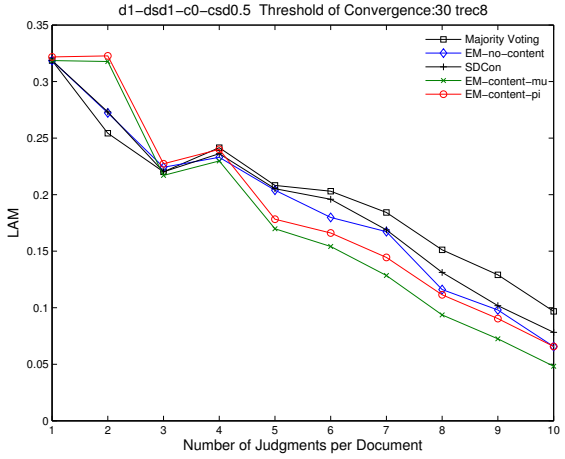

(a) Threshold of Convergence = 1



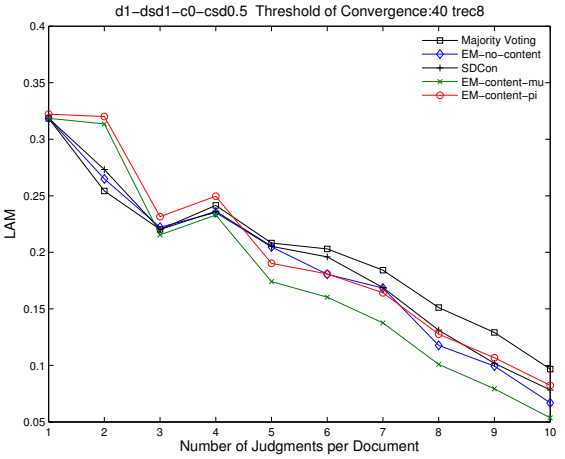

(b) Threshold of Convergence = 10



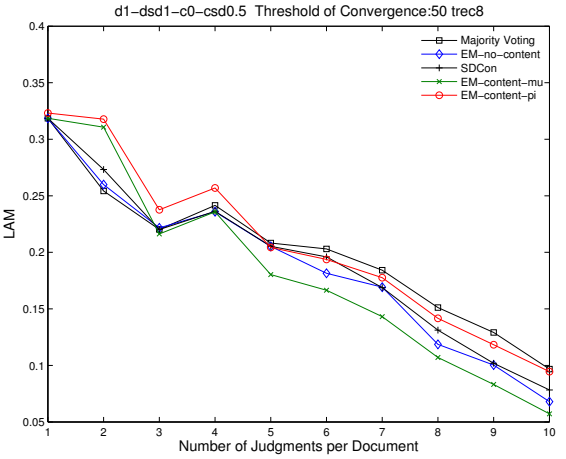

(c) Threshold of Convergence = 20




(d) Threshold of Convergence = 30

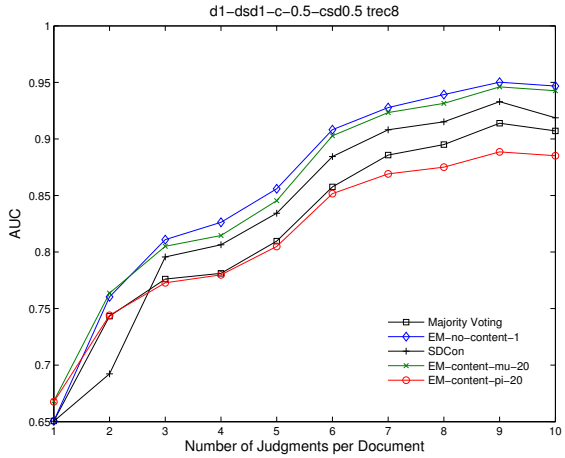



(e) Threshold of Convergence = 40

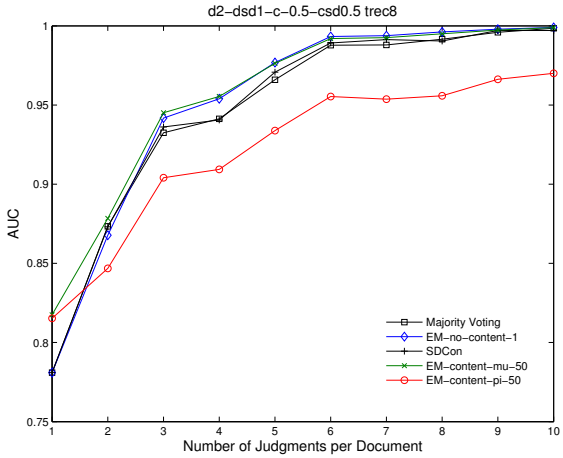



(f) Threshold of Convergence = 50

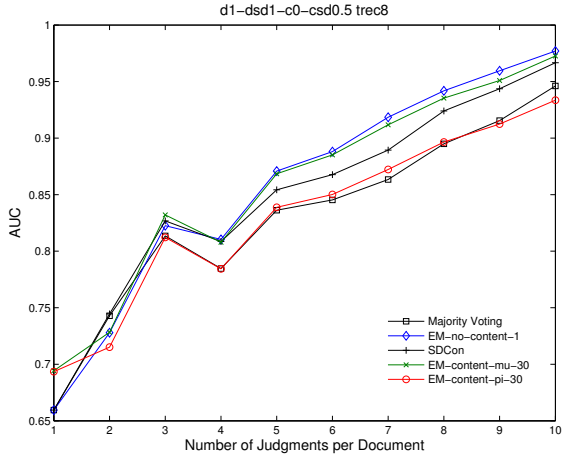
Figure 5.11: The LAM results of varying the convergence condition of EM algorithms on Trec8, using simulation setting $d1$–$dsd1$–$c0$–$csd0.5$



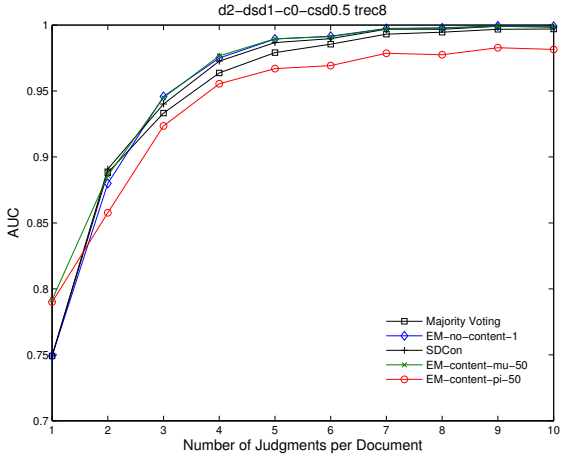

(a) $d' = 1, c = -0.5$



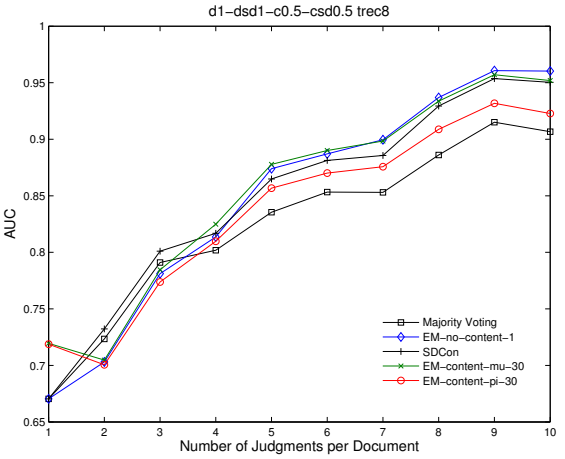

(b) $d' = 2, c = -0.5$



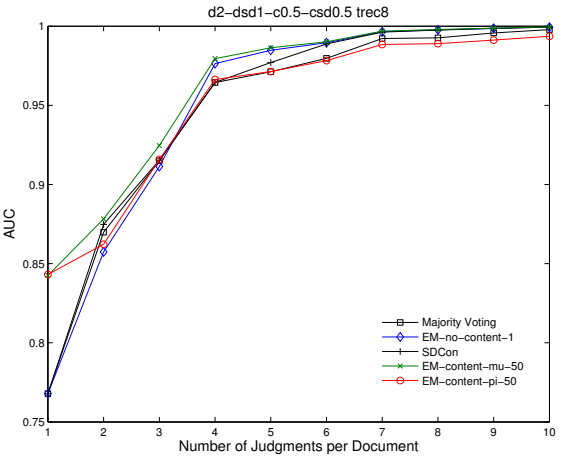

(c) $d' = 1, c = 0$




(d) $d' = 2, c = 0$

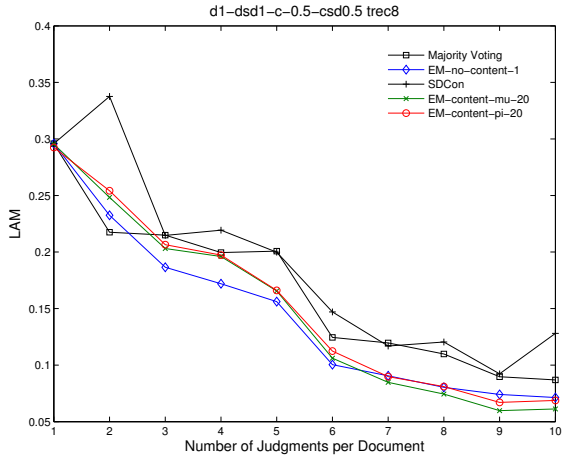



(e) $d' = 1, c = 0.5$

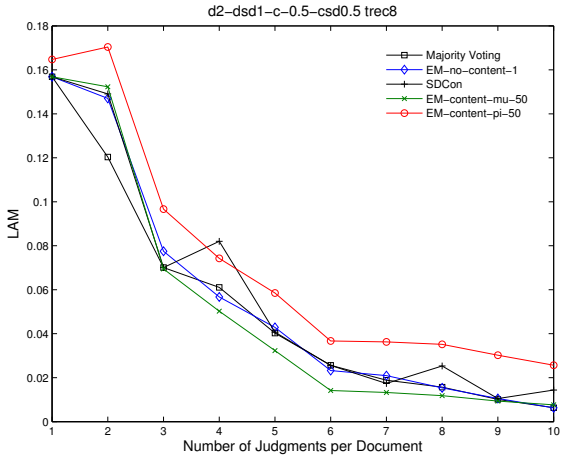



(f) $d' = 2, c = 0.5$

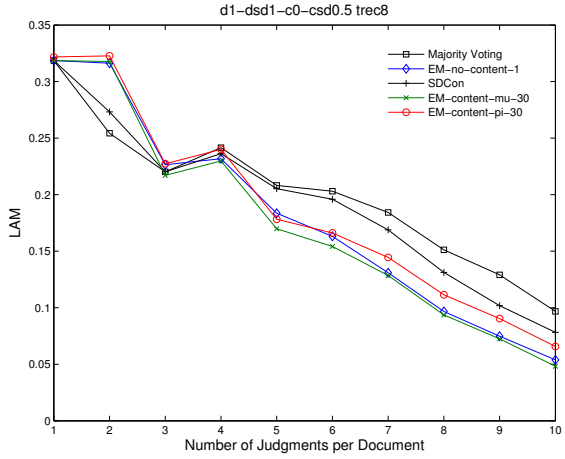
Figure 5.12: The AUC results of comparing consensus algorithms under different simulation settings on Trec8



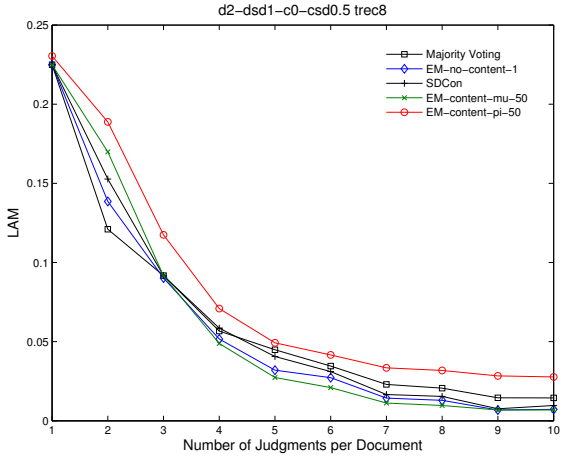

(a) $d' = 1, c = -0.5$



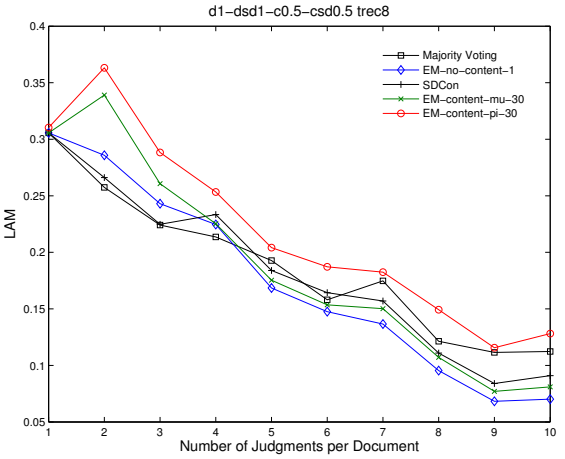

(b) $d' = 2, c = -0.5$



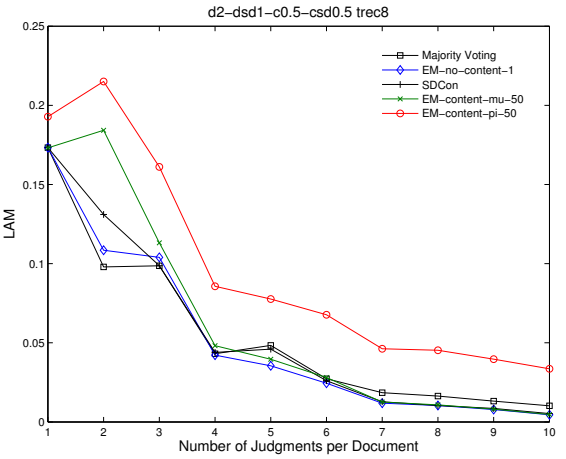

(c) $d' = 1, c = 0$




(d) $d' = 2, c = 0$

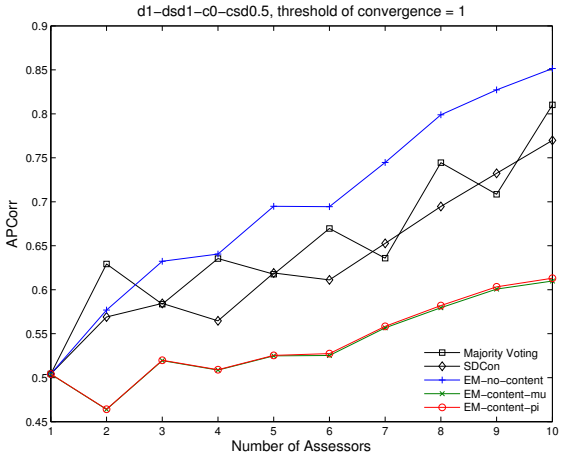



(e) $d' = 1, c = 0.5$

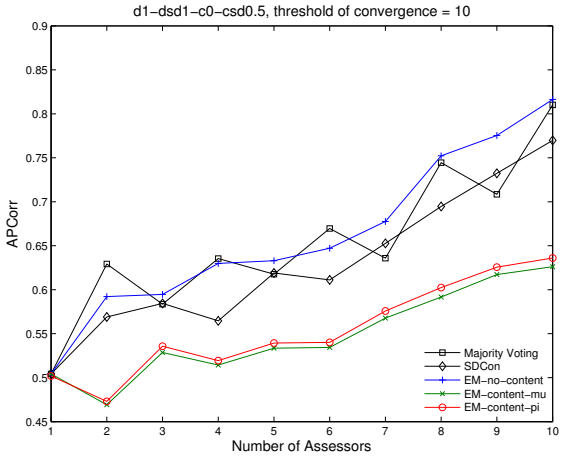



(f) $d' = 2, c = 0.5$

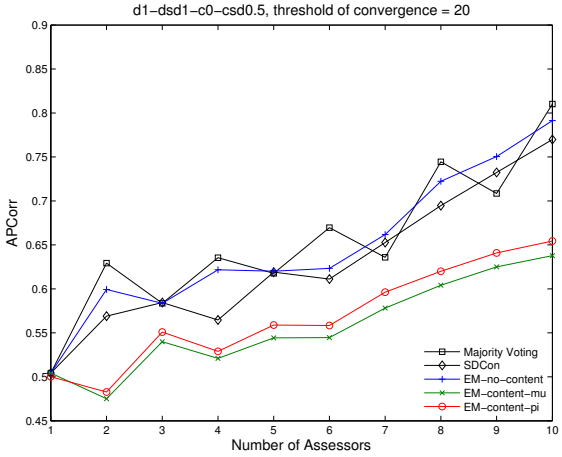
Figure 5.13: The LAM results of comparing consensus algorithms under different simulation settings on Trec8



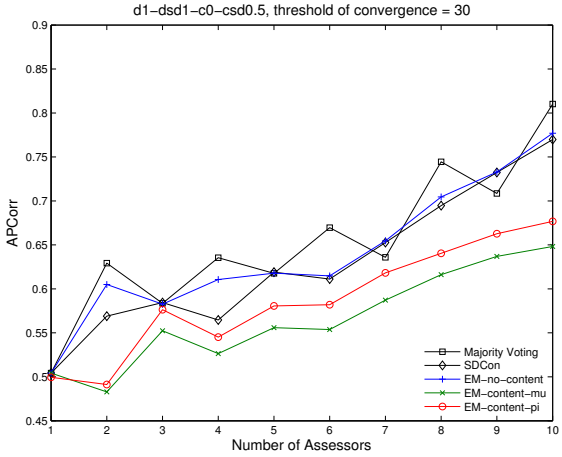

(a) Threshold of Convergence = 1



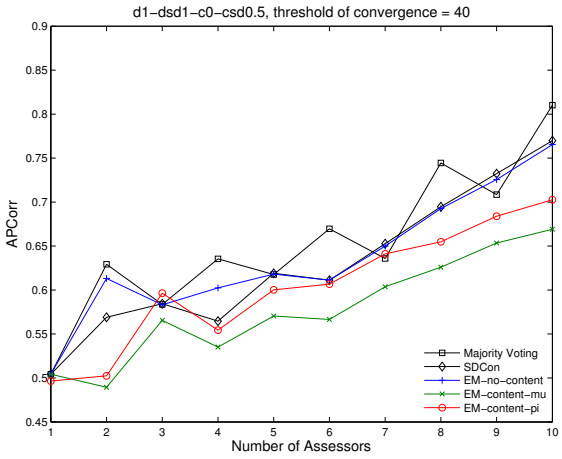

(b) Threshold of Convergence = 10



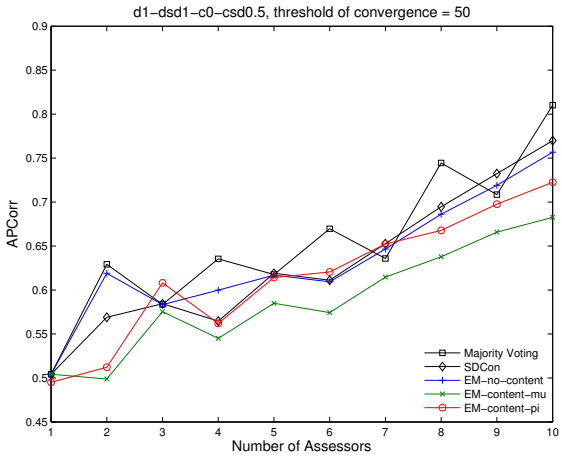

(c) Threshold of Convergence = 20




(d) Threshold of Convergence = 30

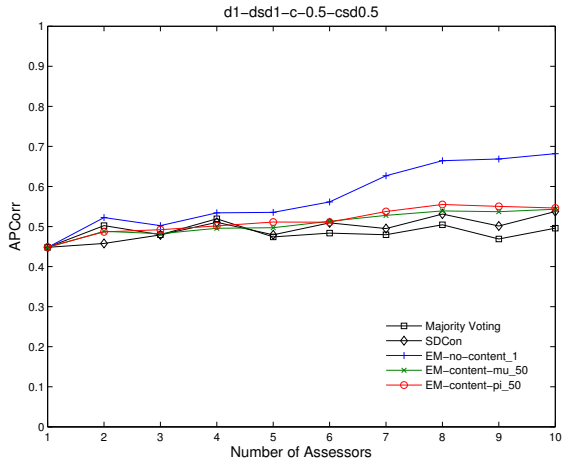



(e) Threshold of Convergence = 40

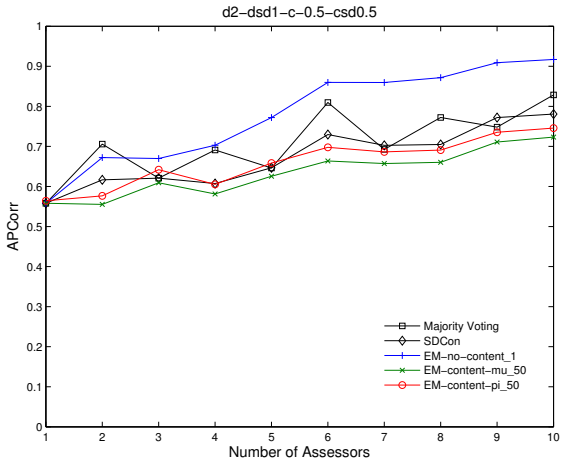



(f) Threshold of Convergence = 50

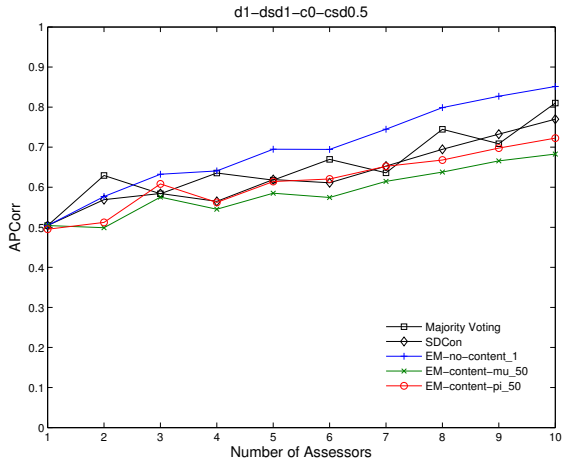
Figure 5.14: The APCorr results of varying the convergence condition of EM algorithms on Trec8



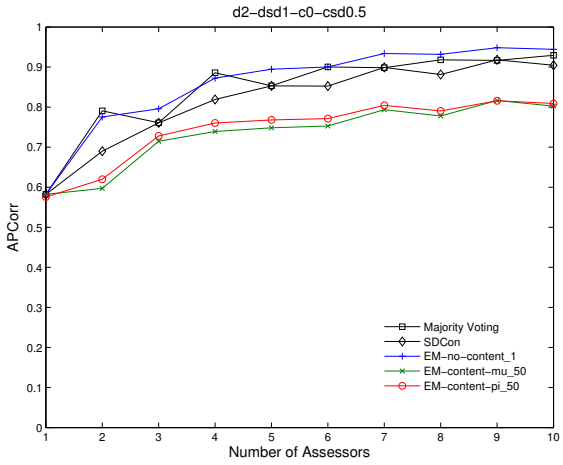

(a) $d' = 1, c = -0.5$



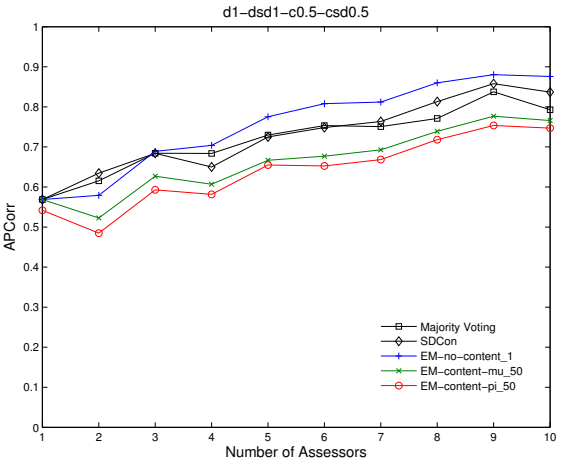

(b) $d' = 2, c = -0.5$



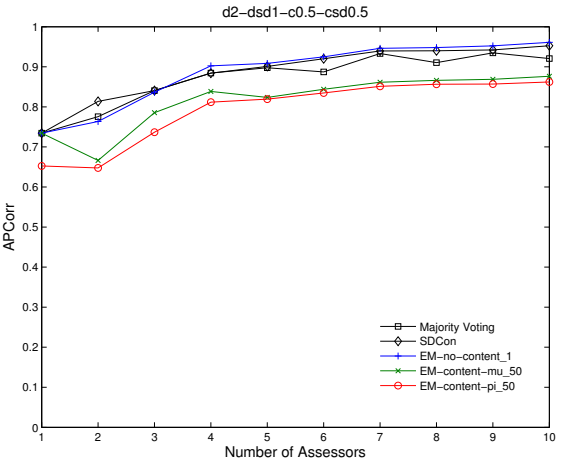

(c) $d' = 1, c = 0$




(d) $d' = 2, c = 0$

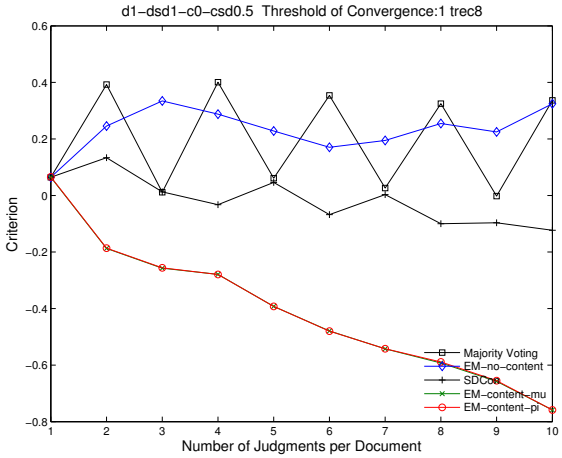



(e) $d' = 1, c = 0.5$

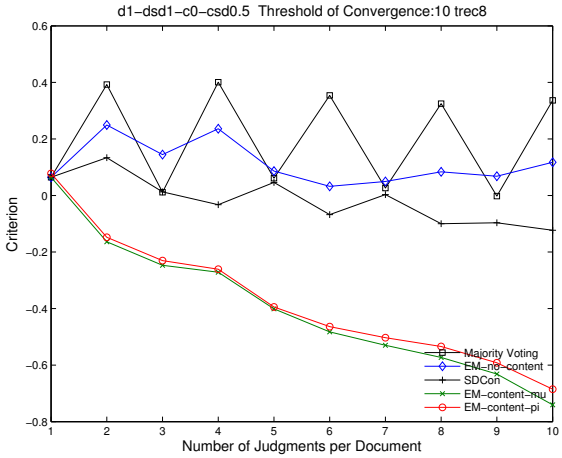



(f) $d' = 2, c = 0.5$

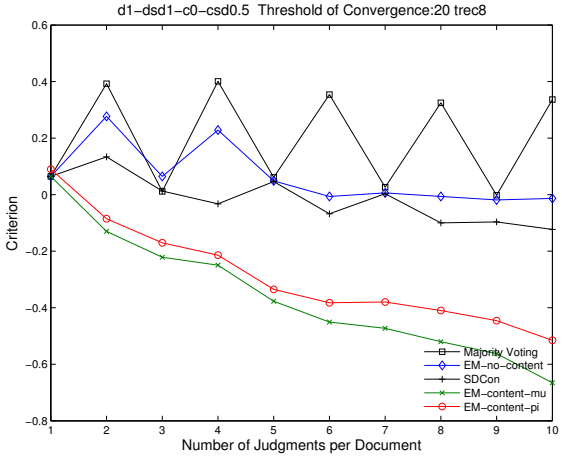
Figure 5.15: The APCorr results of comparing consensus algorithms under different simulation settings on Trec8



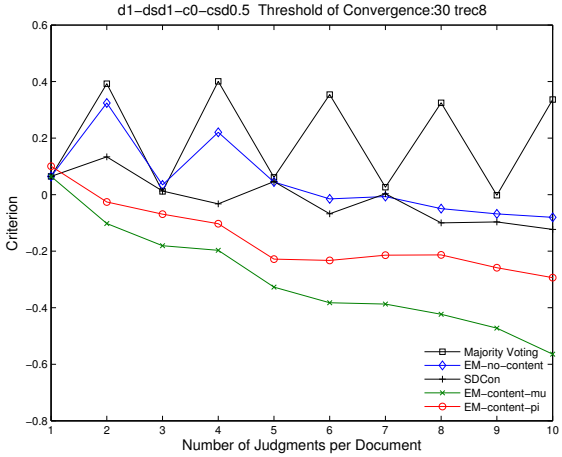

(a) Threshold of Convergence = 1



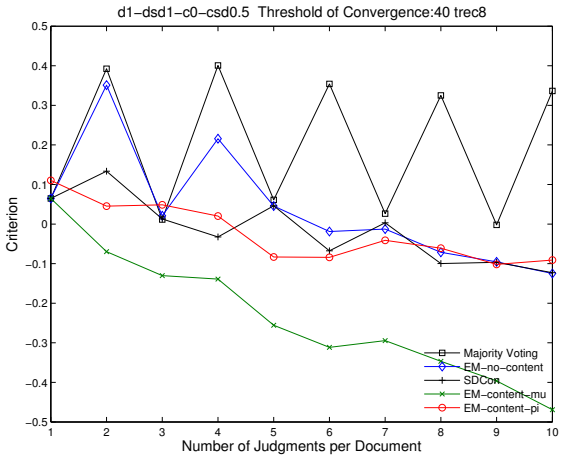

(b) Threshold of Convergence = 10



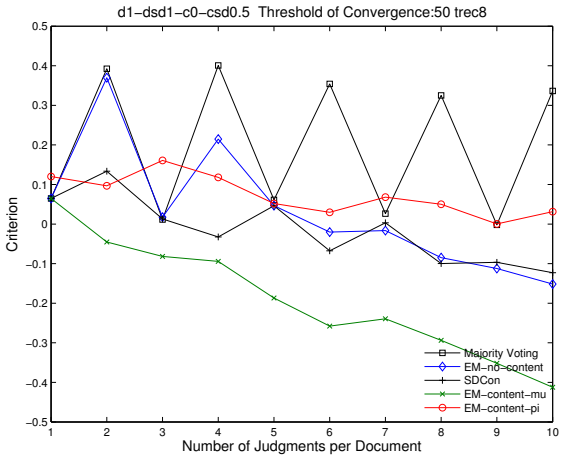

(c) Threshold of Convergence = 20




(d) Threshold of Convergence = 30

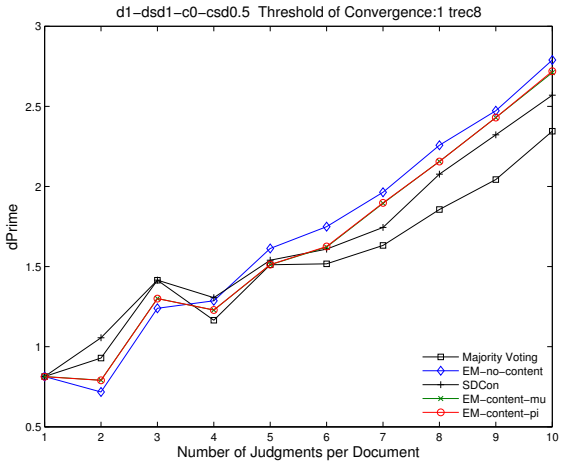



(e) Threshold of Convergence = 40

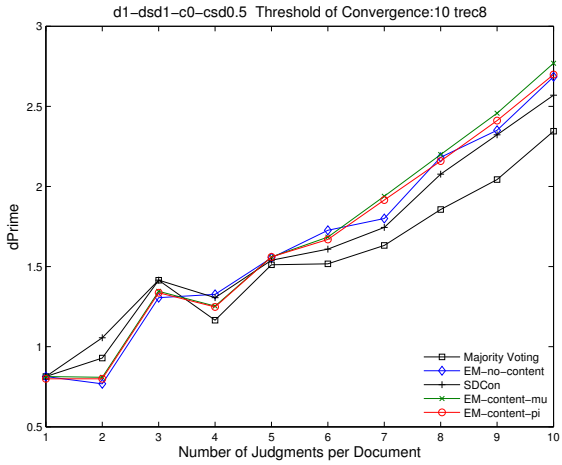



(f) Threshold of Convergence = 50

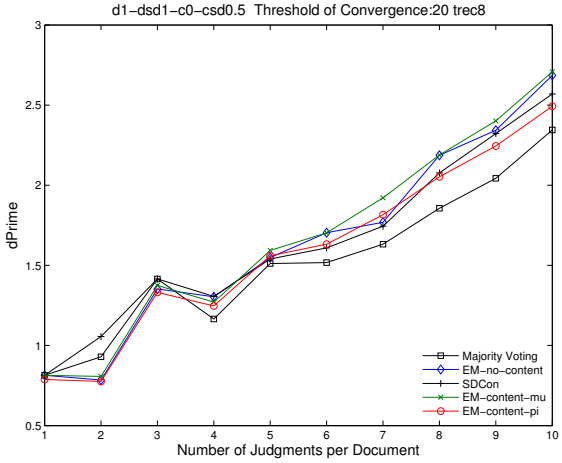
Figure 5.16: The criterion of varying the convergence condition of EM algorithms on Trec8, using simulation setting $d1$–$dsd1$–$c0$–$csd0.5$



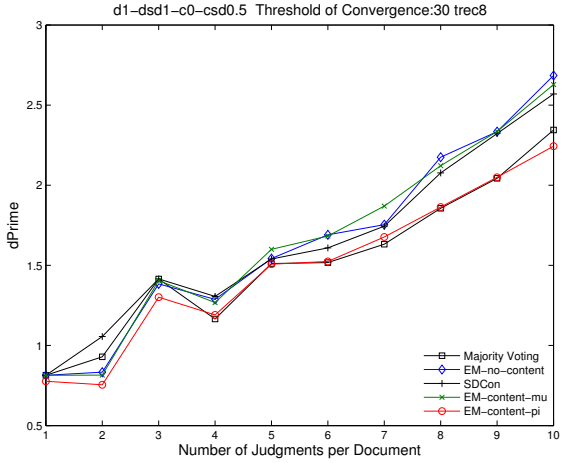

(a) Threshold of Convergence = 1



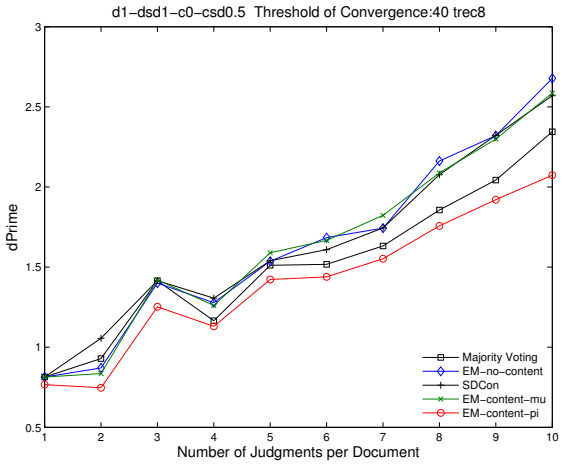

(b) Threshold of Convergence = 10



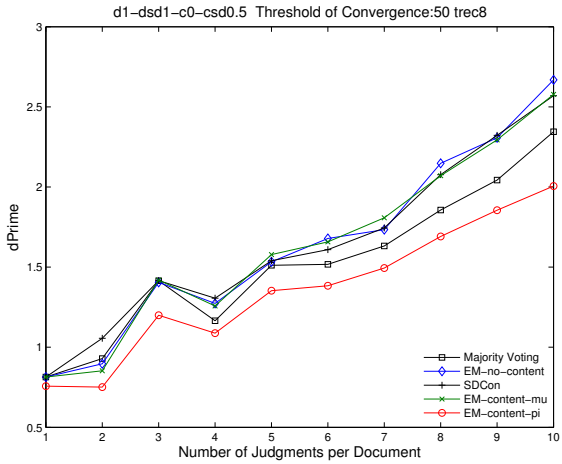

(c) Threshold of Convergence = 20




(d) Threshold of Convergence = 30




(e) Threshold of Convergence = 40




(f) Threshold of Convergence = 50

Figure 5.17: The $d'$ of varying the convergence condition of EM algorithms on Trec8, using simulation setting $d1$–$dsd1$–$c0$–$csd0.5$

# Chapter 6

# Conclusion and Future Work

In this thesis, we mainly investigate whether we can replace the relevance judgments from experts with those from multiple non-expert assessors in the ranking system evaluation.

To simulate different types of errors made by assessors, we rely on signal detection theory that models each assessor with two factors: the discriminative ability to distinguish between relevant and non-relevant document, and the criterion of relevance. Based on the simulation results, we find that the reasonably conservative consensus labels are preferable in order to keep high correlation with experts' judgments. Meanwhile, we investigate two factors that affect the conservativeness of the consensus labels: the assessors' criterion, and the threshold of relevance. We also find that same assessor produces worse correlation results on the dataset that is harder to judge.

Moreover, as to those uncertain documents with equal confidence to be either relevant or non-relevant, we propose and investigate five strategies to break the ties. The results on IR evaluation suggest that if we have zero prior knowledge about the dataset, *MajorClass* is the recommended; otherwise, we should use *Larger* or *LargerEq* based on the prevalence information.

Finally, we investigate the content-based consensus algorithm from Raykar et al. (2010), which incorporates both document contents and noisy labels into the decision-making process. The experiment results show this content-based algorithm may or may not help the results, depending on the features of dataset. If the dataset is difficult to judge, like Trec8, we need to make sure the classifier work on this dataset before we use it into co-training method to find consensus.

There are a few potential places needed to be improved.

The current simulation method only captures the random errors made by the assessors. However, many other factors, like misunderstandings of the task, fatigue and etc., also potentially cause the judgment errors. We leave it for future work.

Simulation is good since it helps us understand the effects of various relevance judgment errors on the ranking system evaluation. In the next step, experiments on a real dataset are needed to examine the level to which the simulation results agree with reality.

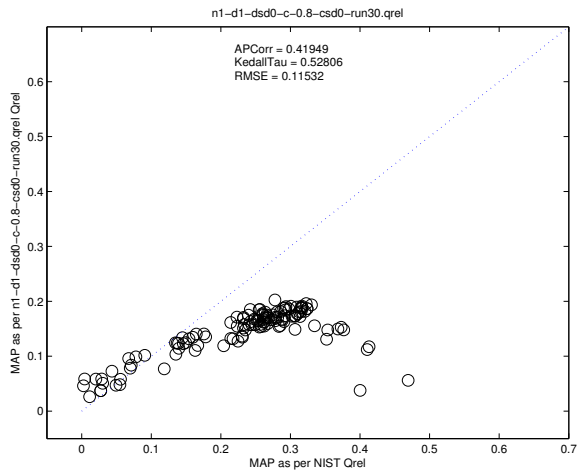
Moreover, increasing the standard of relevance can make the overall judging system more conservative. But, should we keep using same relevance threshold for the documents with different number of judgments? Essentially, it becomes harder to reach agreement when the number of assessors increases. So, we may need to come out an optimal strategy that varies the threshold of relevance based on the number of assessors we have, and relaxes the relevance standard when more assessors judge same documents.

In the end, we need to make sure the both parts of EM-content algorithms are robust, so that each part receives accurate labels for estimation. In our case, if we are given an extremely imbalanced dataset, we need to think about a way to handle the asymmetric costs of misclassification.

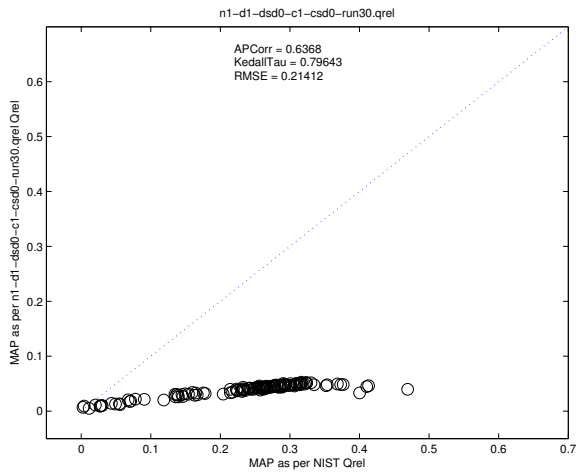
# APPENDICES

# Appendix A

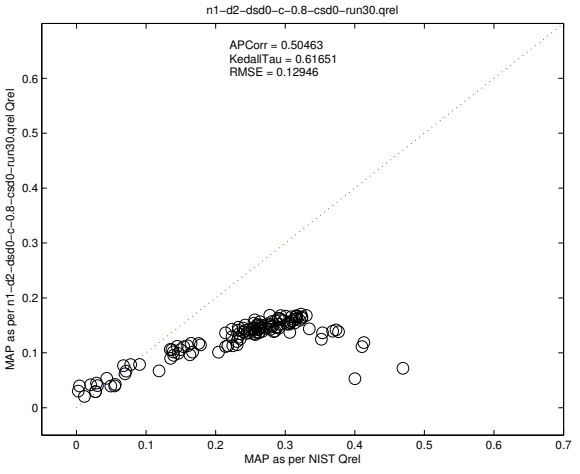
# Extra Plots For The Effects of Relevance Judgment Errros on IR evaluation



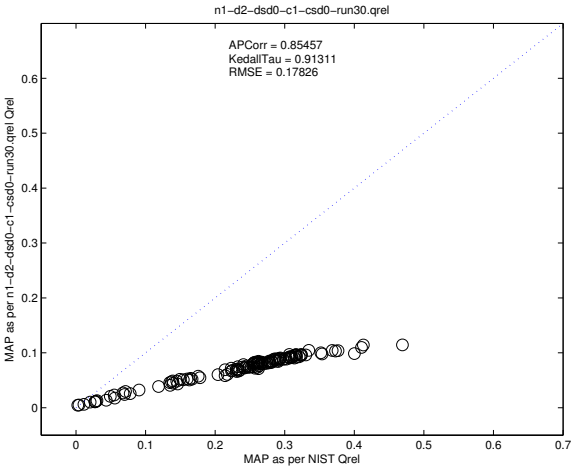

(a) $d' = 1, c = -0.8$, Simulated Run 30




(b) $d' = 1, c = co1$, Simulated Run 30




(c) $d' = 2, c = -0.8$, Simulated Run 30

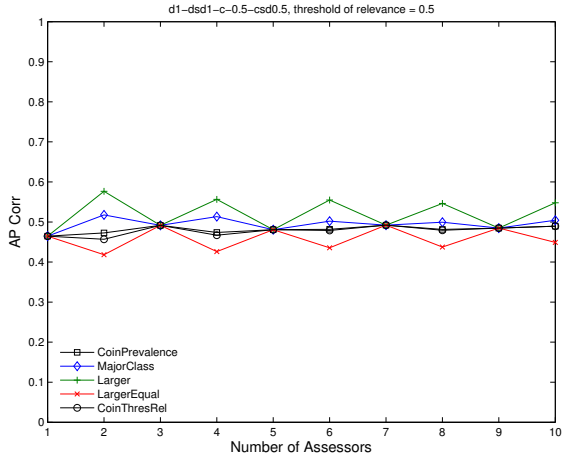



(d) $d' = 2, c = -0.8$, Simulated Run 30

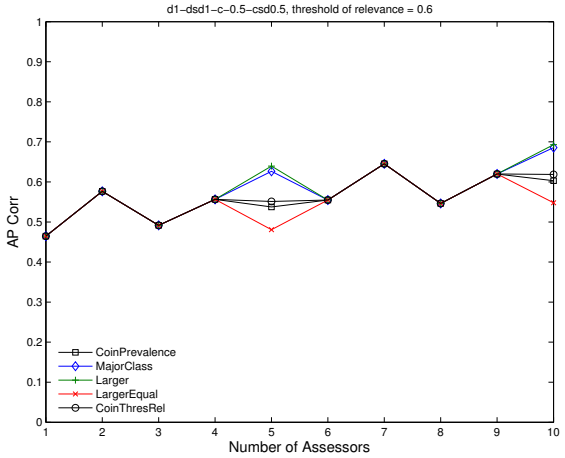
Figure A.1: MAPs distribution against NIST qrels and pseudo qrels on Trec8

# Appendix B

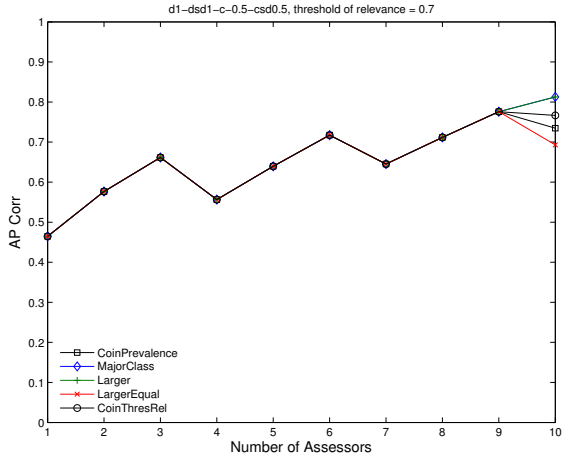
# Extra Plots For Break Ties In Consensus Decision-Making



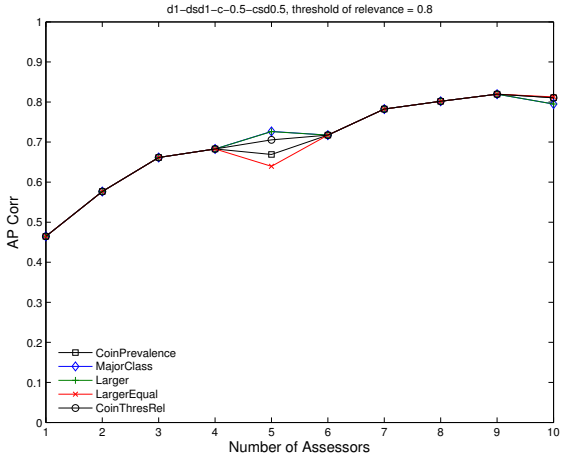

(a) Threshold of Relevance = 0.5



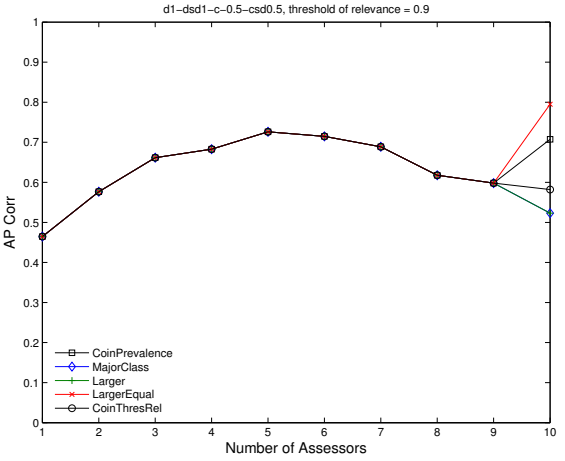

(b) Threshold of Relevance = 0.6



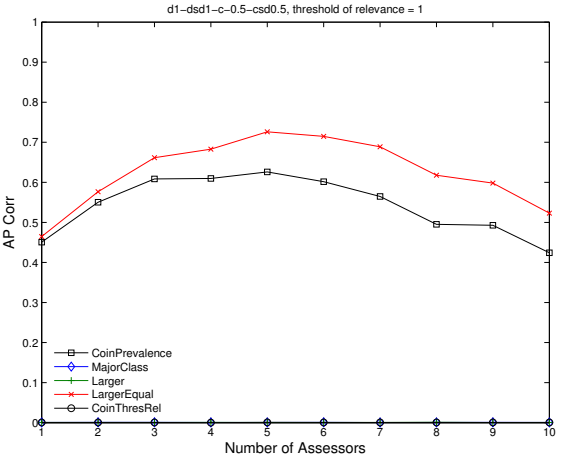

(c) Threshold of Relevance = 0.7




(d) Threshold of Relevance = 0.8

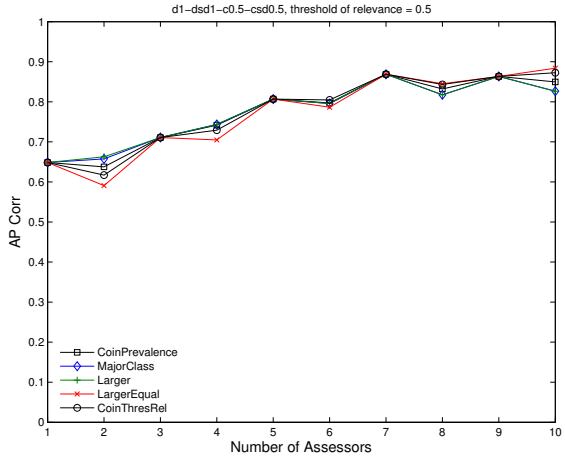



(e) Threshold of Relevance = 0.9

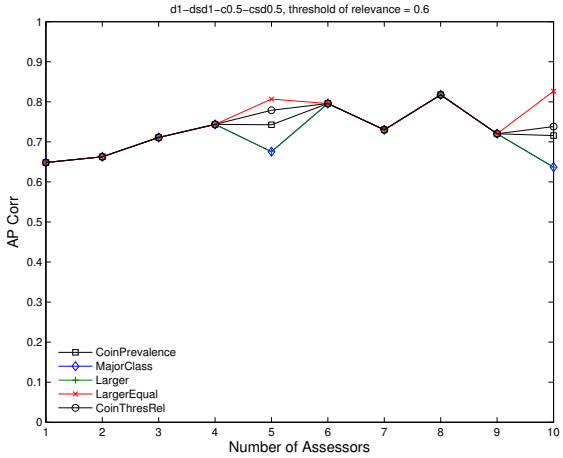



(f) Threshold of Relevance = 1

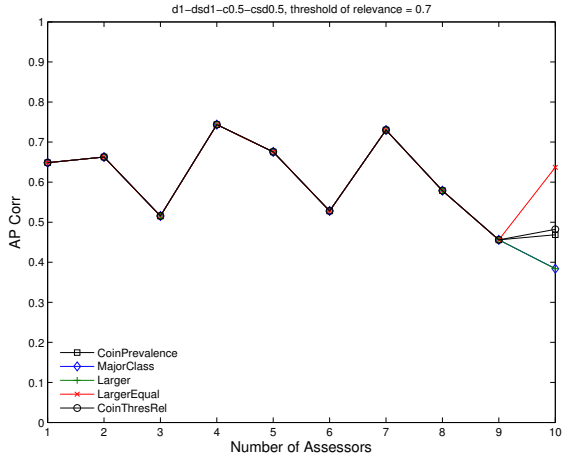
Figure B.1: The APCorr results of varying threshold of relevance for $d1$–$dsd1$–$c$−0.5–$csd0.5$ on Trec8



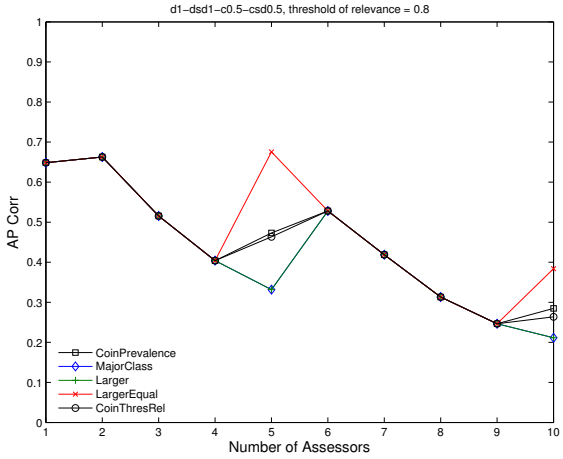

(a) Threshold of Relevance = 0.5



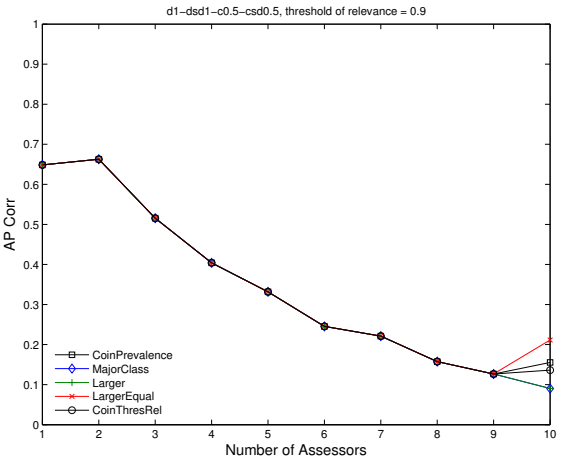

(b) Threshold of Relevance = 0.6



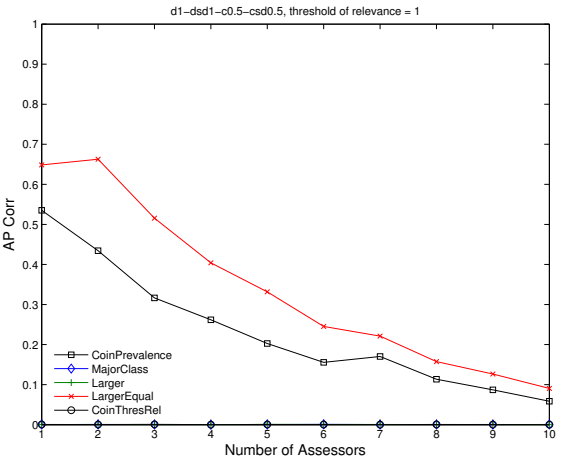

(c) Threshold of Relevance = 0.7




(d) Threshold of Relevance = 0.8

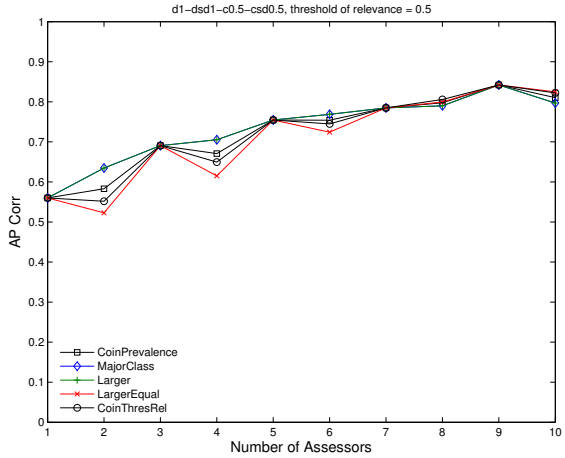



(e) Threshold of Relevance = 0.9

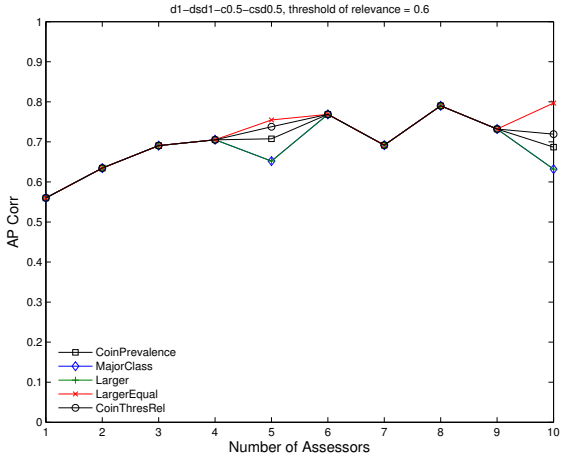



(f) Threshold of Relevance = 1

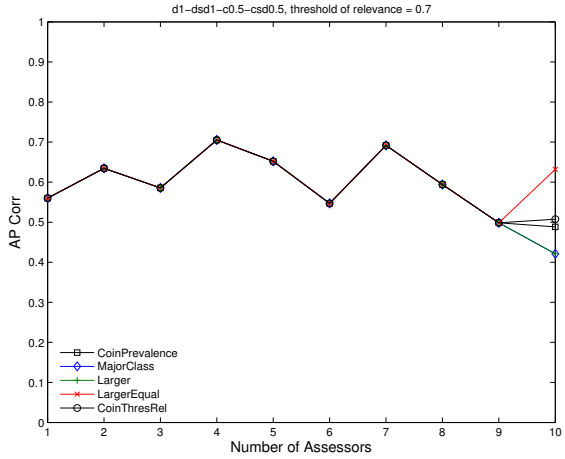
Figure B.2: The APCorr results of varying threshold of relevance for $d1$–$dsd1$–$c0.5$–$csd0.5$ on Robust2005



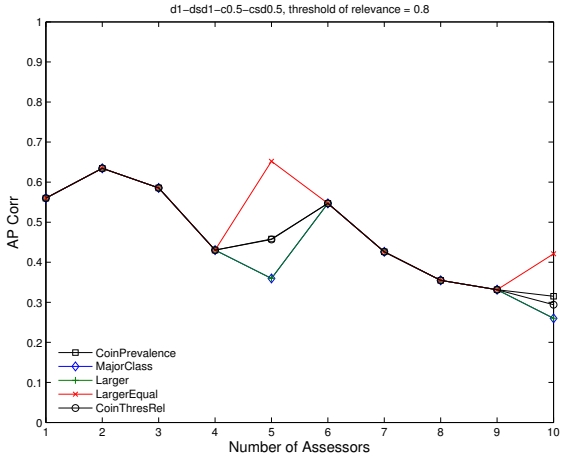

(a) Threshold of Relevance = 0.5



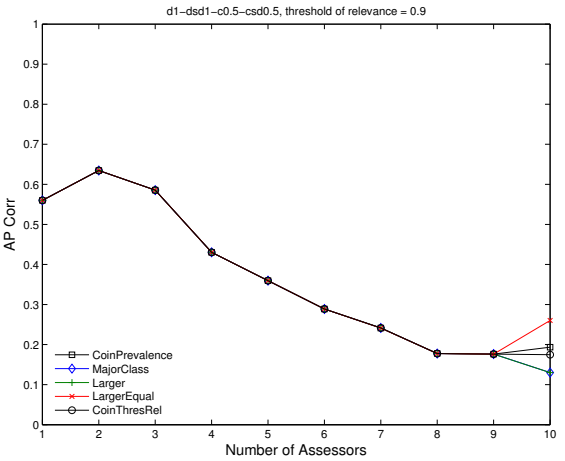

(b) Threshold of Relevance = 0.6



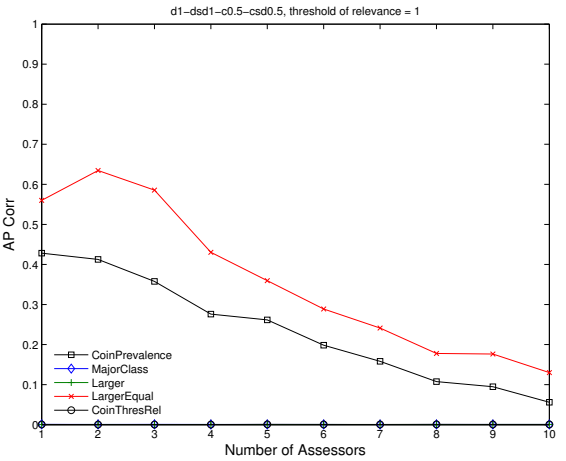

(c) Threshold of Relevance = 0.7




(d) Threshold of Relevance = 0.8




(e) Threshold of Relevance = 0.9




(f) Threshold of Relevance = 1

Figure B.3: The APCorr results of varying threshold of relevance for $d1$–$dsd1$–$c0.5$–$csd0.5$ on Trec8

# References


O. Alonso. Implementing crowdsourcing-based relevance experimentation: an industrial perspective. *Information retrieval*, pages 1–20, 2012.

O. Alonso and R. Baeza-Yates. Design and implementation of relevance assessments using crowdsourcing. *Advances in information retrieval*, pages 153–164, 2011.

O. Alonso and S. Mizzaro. Can we get rid of TREC assessors? Using Mechanical Turk for relevance assessment. In *Proceedings of the SIGIR 2009 workshop on the future of IR evaluation*, pages 15–16, 2009.

O. Alonso, D.E. Rose, and B. Stewart. Crowdsourcing for relevance evaluation. In *ACM SIGIR Forum*, volume 42, pages 9–15. ACM, 2008.

J.A. Aslam, V. Pavlu, and E. Yilmaz. A statistical method for system evaluation using incomplete judgments. In *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, pages 541–548. ACM, 2006.

P. Bailey, N. Craswell, I. Soboroff, P. Thomas, A.P. de Vries, and E. Yilmaz. Relevance assessment: are judges exchangeable and does it matter. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval*, pages 667–674. ACM, 2008.

R. Blanco, H. Halpin, D.M. Herzig, P. Mika, J. Pound, H.S. Thompson, and T. Tran Duc. Repeatable and reliable search system evaluation using crowdsourcing. In *Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval*, pages 923–932. ACM, 2011.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on computational learning theory*, pages 92–100. ACM, 1998.

Chris Buckley, Darrin Dimmick, Ian Soboroff, and Ellen Voorhees. Bias and the limits of pooling for large collections. *Information retrieval*, 10(6):491–508, 2007.

Chris Callison-Burch. Fast, cheap, and creative: evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 conference on empirical methods in natural language processing: Volume 1-Volume 1*, pages 286–295. Association for Computational Linguistics, 2009.

B. Carterette and J. Allan. Incremental test collections. In *Proceedings of the 14th ACM international conference on information and knowledge management*, pages 680–687. ACM, 2005.

B. Carterette and I. Soboroff. The effect of assessor error on IR system evaluation. In *Proceeding of the 33rd international ACM SIGIR conference on research and development in information retrieval*, pages 539–546. ACM, 2010.

Cyril Cleverdon. The cranfield tests on index language devices. In *Aslib proceedings*, volume 19, pages 173–194. MCB UP Ltd, 1967.

Gordon V Cormack and Thomas R Lynam. TREC 2005 spam track overview. In *TREC*, 2005.

Gordon V Cormack, Mark D Smucker, and Charles LA Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information retrieval*, 14(5):441–465, 2011.

G.V. Cormack, C.R. Palmer, and C.L.A. Clarke. Efficient construction of large test collections. In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval*, pages 282–289. ACM, 1998.

Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

C. Eickhoff and A.P. de Vries. Increasing cheat robustness of crowdsourcing tasks. *Information retrieval*, pages 1–17, 2012.

C. Grady and M. Lease. Crowdsourcing document relevance assessment with Mechanical Turk. In *Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon's Mechanical Turk*, pages 172–179. Association for Computational Linguistics, 2010.

M. Hosseini, I. Cox, and N. Milic-Frayling. Optimizing the cost of information retrieval test collections. In *Proceedings of the 4th workshop on workshop for Ph. D. students in information & knowledge management*, pages 79–82. ACM, 2011a.

M. Hosseini, I.J. Cox, N. Milic-Frayling, T. Sweeting, and V. Vinay. Prioritizing relevance judgments to improve the construction of IR test collections. In *Proceedings of the 20th ACM international conference on information and knowledge management*, pages 641–646. ACM, 2011b.

M. Hosseini, I. Cox, N. Milić-Frayling, G. Kazai, and V. Vinay. On aggregating labels from multiple crowd workers to infer relevance of documents. *Advances in information retrieval*, pages 182–194, 2012.

Panagiotis Ipeirotis. Demographics of Mechanical Turk. 2010a.

Panagiotis G Ipeirotis. Analyzing the Amazon Mechanical Turk marketplace. *XRDS: Crossroads, The ACM magazine for students*, 17(2):16–21, 2010b.

Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67. ACM, 2010.

H.J. Jung and M. Lease. Evaluating classifiers without expert labels. *arXiv preprint arXiv:1212.0960*, 2012.

Hyun Joon Jung and Matthew Lease. Improving consensus accuracy via z-score and weighted voting. In *AAAI workshop on human computation (HComp)*, 2011.

G. Kazai. An exploration of the influence that task parameters have on the performance of crowds. *Proceedings of the CrowdConf 2010*, 2010.

G. Kazai and N. Milic-Frayling. On the evaluation of the quality of relevance assessments collected through crowdsourcing. In *SIGIR 2009 workshop on the future of IR evaluation*, page 21. Citeseer, 2009.

G. Kazai, J. Kamps, M. Koolen, and N. Milic-Frayling. Crowdsourcing for book search evaluation: impact of hit design on comparative system ranking. *SIGIR 2011, July 24-28, 2011, Beijing, China*, 2011a.

G. Kazai, J. Kamps, and N. Milic-Frayling. Worker types and personality traits in crowdsourcing relevance labels. In *Proceedings of the 20th ACM international conference on information and knowledge management*, pages 1941–1944. ACM, 2011b.

G. Kazai, N. Craswell, E. Yilmaz, and SMM Tahaghoghi. An analysis of systematic judging errors in information retrieval. In *Proceedings of the 21st ACM international conference on information and knowledge management*, pages 105–114. ACM, 2012a.

G. Kazai, J. Kamps, and N. Milic-Frayling. An analysis of human factors and label accuracy in crowdsourcing relevance judgments. *Information retrieval*, pages 1–41, 2012b.

Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

K.A. Kinney, S.B. Huffman, and J. Zhai. How evaluator domain expertise affects search result relevance judgments. In *Proceedings of the 17th ACM conference on information and knowledge management*, pages 591–598. ACM, 2008.

Abhimanu Kumar and Matthew Lease. Modeling annotator accuracies for supervised learning. In *WSDM workshop on crowdsourcing for search and data mining*, pages 19–22, 2011.

J. Le, A. Edmonds, V. Hester, and L. Biewald. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *SIGIR 2010 workshop on crowdsourcing for search evaluation*, pages 21–26, 2010.

T. Li, C. Zhang, T. Xia, M. Wu, and J. Xie. Quality control of crowdsourcing through workers experience. In *Proceedings of the ACM SIGIR workshop on crowdsourcing for information retrieval*, 2011.

Neil A Macmillan and C Douglas Creelman. *Detection theory: A user's guide*. Psychology press, 2004.

R.M.C. McCreadie, C. Macdonald, and I. Ounis. Crowdsourcing a news query classification dataset. In *Proceedings of the ACM SIGIR 2010 workshop on crowdsourcing for search evaluation (CSE 2010)*, pages 31–38, 2010.

S. Nowak and S. Rüger. How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation. In *Proceedings of the international conference on Multimedia information retrieval*, pages 557–566. ACM, 2010.

D. Oleson, A. Sorokin, G. Laughlin, V. Hester, J. Le, and L. Biewald. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. *Proc. HComp*, 2011.

Vikas C Raykar, Shipeng Yu, Linda H Zhao, Anna Jerebko, Charles Florin, Gerardo Hermosillo Valadez, Luca Bogoni, and Linda Moy. Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *Proceedings of the 26th Annual international conference on machine learning*, pages 889–896. ACM, 2009.

Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *The Journal of machine learning research*, 99:1297–1322, 2010.

Stephen E Robertson. The probability ranking principle in IR. *Journal of documentation*, 33(4):294–304, 1977.

Joel Ross, Lilly Irani, M Silberman, Andrew Zaldivar, and Bill Tomlinson. Who are the crowdworkers?: shifting demographics in Mechanical Turk. In *Proceedings of the 28th of the international conference extended abstracts on human factors in computing systems*, pages 2863–2872. ACM, 2010.

M. Sanderson and H. Joho. Forming test collections with no system pooling. In *Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval*, pages 33–40. ACM, 2004.

M Smucker, Gabriella Kazai, and Matthew Lease. Overview of the TREC 2012 crowdsourcing track. In *Proceedings of the 21st NIST text retrieval conference (TREC)*, 2013.

Mark Smucker. Crowdsourcing with a crowd of one and other TREC 2011 crowdsourcing and web track experiments. In *Proceedings of the Text REtrieval Conference (TREC 2011)*, 2012.

Mark D Smucker and Chandra Prakash Jethani. Measuring assessor accuracy: a comparison of nist assessors and user study participants. In *Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval*, pages 1231–1232. ACM, 2011a.

M.D. Smucker and C.P. Jethani. The crowd vs. the lab: A comparison of crowd-sourced and university laboratory participant behavior. In *Proceedings of the SIGIR 2011 Workshop on crowdsourcing for information retrieval*, 2011b.

I. Soboroff, C. Nicholas, and P. Cahan. Ranking retrieval systems without relevance judgments. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*, pages 66–73. ACM, 2001.

W. Tang and M. Lease. Semi-supervised consensus labeling for crowdsourcing. In *Proceedings of the ACM SIGIR workshop on crowdsourcing for information retrieval*, 2011.

Ellen Voorhees, Donna K Harman, et al. *TREC: Experiment and evaluation in information retrieval*, volume 63. MIT press Cambridge, 2005.

Ellen M Voorhees. Overview of TREC 2005. In *Proceedings of TREC*, 2005.

Ellen M Voorhees and Donna Harman. Overview of the Eighth Text REtrieval Conference (TREC-8). In *Proceedings of TREC*, volume 8, pages 1–24, 1999.

E.M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information processing & management*, 36(5):697–716, 2000.

E.M. Voorhees. Evaluation by highly relevant documents. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*, pages 74–82. ACM, 2001.

J. Vuurens, A.P. de Vries, and C. Eickhoff. How much spam can you take? An analysis of crowdsourcing results to increase accuracy. In *Proc. ACM SIGIR Workshop on crowdsourcing for information retrieval (CIR11)*, pages 21–26, 2011.

W. Webber, P. Chandar, and B. Carterette. Alternative assessor disagreement and retrieval depth. 2012.

Yan Yan, Romer Rosales, Glenn Fung, and Jennifer Dy. Active learning from crowds. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1161–1168, 2011.

E. Yilmaz and J.A. Aslam. Estimating average precision with incomplete and imperfect judgments. In *Conference on information and Knowledge Management: Proceedings of the 15 th ACM international conference on information and knowledge management*, volume 6, pages 102–111, 2006.

Emine Yilmaz, Javed A Aslam, and Stephen Robertson. A new rank correlation coefficient for information retrieval. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval*, pages 587–594. ACM, 2008.

J. Zobel. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval*, pages 307–314. ACM, 1998.