The University Carlos III of Madrid at TREC 2011 Crowdsourcing Track: Notebook Paper

Julián Urbano, Mónica Marrero, Diego Martín, Jorge Morato, Karina Robles and Juan Lloréns

University Carlos III of Madrid · Department of Computer Science
{jurbano, mmarrero, dmandres, jmorato, krobles, llorens}@inf.uc3m.es

Abstract. This notebook paper describes our participation in both tasks of the TREC 2011 Crowdsourcing Track. For the first one we submitted three runs that used Amazon Mechanical Turk: one where workers made relevance judgments based on a 3-point scale, and two similar runs where workers provided an explicit ranking of documents. All three runs implemented a quality control mechanism at the task level, which was based on a simple reading comprehension test. For the second task we submitted another three runs: one with a stepwise execution of the GetAnotherLabel algorithm by Ipeirotis et al., and two others with a rule-based and a SVM-based model. We also comment on several topics regarding the Track design and evaluation methods.

1. Introduction

The TREC 2011 Crowdsourcing Track was designed to investigate how to better use crowdsourcing platforms to evaluate Information Retrieval systems. The Track was divided in two tasks: obtaining topical relevance judgments from individual workers and computing consensus judgments from several workers. The Knowledge Reuse research group at the University Carlos III of Madrid put together a team of six people to participate in both tasks. We submitted three runs for each task, although most of our work was devoted to the first one. We focused on designing a practical and effective task template for gathering unconventional relevance judgments through Amazon Mechanical Turk (AMT), while studying quality control mechanisms at the task level for such heterogeneous documents as arbitrary HTML pages from the Web. In the second task, two of our runs used a rule-based and an SVM Machine Learning model, while the other one followed a stepwise execution of the GetAnotherLabel algorithm (GAL) [Ipeirotis et al., 2010].

The rest of the paper is organized as follows. Section 2 describes our submissions for the first task, detailing the HIT design, document processing and quality control. Section 3 describes our submissions for the second task, and Section 4 concludes with some observations on the design and evaluation of the Track itself.

2. Task I: Crowdsourcing Individual Judgments

For the first task we submitted three runs (see Table 1), all of which used AMT as the crowdsourcing platform. The task was implemented with external HITs, that is, we hosted the templates and data in our own server, communicating with AMT via the API. This allowed us to have more control over the whole process, besides the possibility of gathering some additional data such as knowing when workers previewed our HITs or where they came from.

	uc3m.graded	uc3m.slider	uc3m.hterms
Uploaded	Sep 12th, 19:22 CEST	Sep 13th, 17:48 CEST	Sep 14th, 18:44 CEST
Hours to complete	8.5	38	20.5
HITs submitted (overhead)	438 (+1%)	535 (+23%)	448 (+3%)
Submitted workers (just previewers)	29 (102)	83 (383)	30 (163)
Average documents per worker	76	32	75
Total cost (including fees)[1]	$95.7	$95.7	$95.7

Table 1. Summary of the submitted runs for Task I.

[1] This is the total cost if rejected work were not paid. See Section 2.4.4.

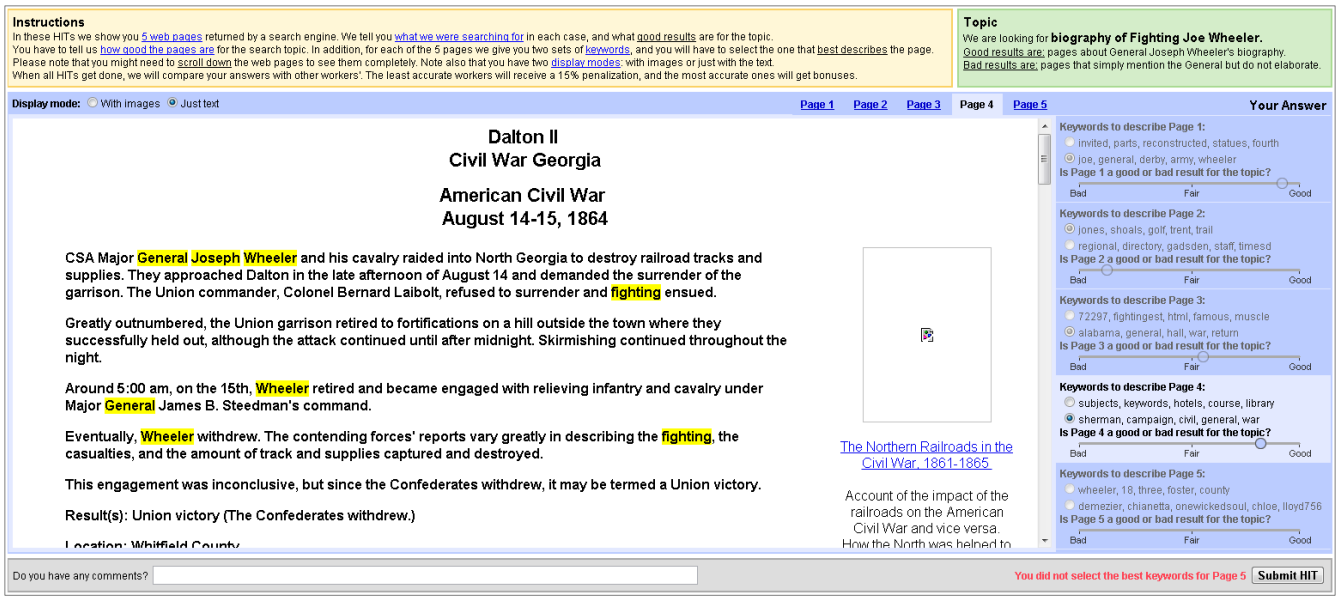


Figure 1. HIT design for run uc3m.hterms.

2.1. HIT Design[2]

We worked on the HIT template to make it as simple, functional and self-contained as possible (see Figure 1). Given that all five documents in a set had to be judged by the same worker, we decided to include them all in a single HIT to make the assignment process easier and allow workers to have all five documents at once to make the ranking judgments easier. Thus, we had a total of 435 HITs for a total of 2175 judgments. We paid $0.20 plus fees per HIT, which makes it $0.04 per document. It was in our plans to try paying a little less, but given the tight deadlines we decided to stick with the $0.04 wage: workers seemed to get interest in our task, and in previous trials we asked them if they thought the payment was fair, and they mostly thought so.

All three runs shared the same template except for the HIT description, instructions and the part where the actual relevance questions were asked. The top left box contained the task instructions. For the first and second runs the focus of the task was on the quality control questions (see Section 2.4.3), making the relevance question secondary (see Section 2.3). For the third run, we made it the other way around. We informed that the most accurate workers would get a bonus, while the least accurate would receive a 15% penalization [Shaw et al., 2011]. However, due to mere programming difficulties, we decided to pay bad workers anyway to avoid unnecessary conflicts for the time being. Some fragments of the instructions were rendered as hyperlinks (see Figure 1), and when clicked upon they triggered a highlighting effect on the part of the template they referred to, so that workers could immediately follow up while reading. For instance, when clicking on "what we were searching for" the box with the topic description would be highlighted blinking three times.

The top right box contained the topic description, with the title in a bigger bold face. Underneath, two lines briefly described what good and bad results were considered for the topic. These descriptions were not directly taken as in the files distributed for the track; we made a simpler version, explaining concepts when appropriate.

The middle right area of the template contained five stacked boxes where workers had to give their answers, one per document. First, we asked for the answer to the quality control question (see Section 2.4), which consisted in selecting one correct answer out of two options. Next, we asked the actual relevance question for the document selected. Unlike the quality control question, the relevance question was different in all three runs (see Section 2.3).

The five documents in the set were displayed below the instructions and topic description using a tabbed design where workers could display one document or another by clicking on the corresponding tab header. This way, there is no need to follow external links or do long scrolls throughout the HIT to go from document to document. The tab and answer boxes corresponding to the document currently displayed were rendered with a lighter color, while the others' where darker and with all options disabled, so that workers could easily know which of the five answer boxes belonged to the document and

[2] All HIT templates and gathered data can be downloaded from http://www.kr.inf.uc3m.es/trec2011/.

make no mistake. Tab headers were located in the upper right corner, while the upper left corner offered two different options to display documents:

- With images. This mode kept CSS styles, layout, images, etc. However, we removed all scripts, embedded objects and all HTML elements not related with the page rendering (see Section 2.2). Therefore, workers saw the documents nearly as they would if surfing the Web themselves.
- Just text. This mode rendered documents as in the images mode, but removing all images, colors, custom formatting and all other elements not related with the page layout. This results in a simple black and white view, while still maintaining the headers and layout scheme of the original page.

In early trials we asked workers which of the two display modes they preferred. Most of the times they chose the mode with images, so we made it the default one. However, some workers always selected the just text mode, so we decided to keep it anyway and let workers decide. The bottom of the HIT contained a simple box with a textbox to provide optional feedback (they rarely did), and in the bottom right corner we placed the submission button. When clicking this button, a script checked that workers answered all five relevance questions and all five quality control questions. If not, they were informed with a message displayed in red font right before the submission button, keeping the browser in the same page.

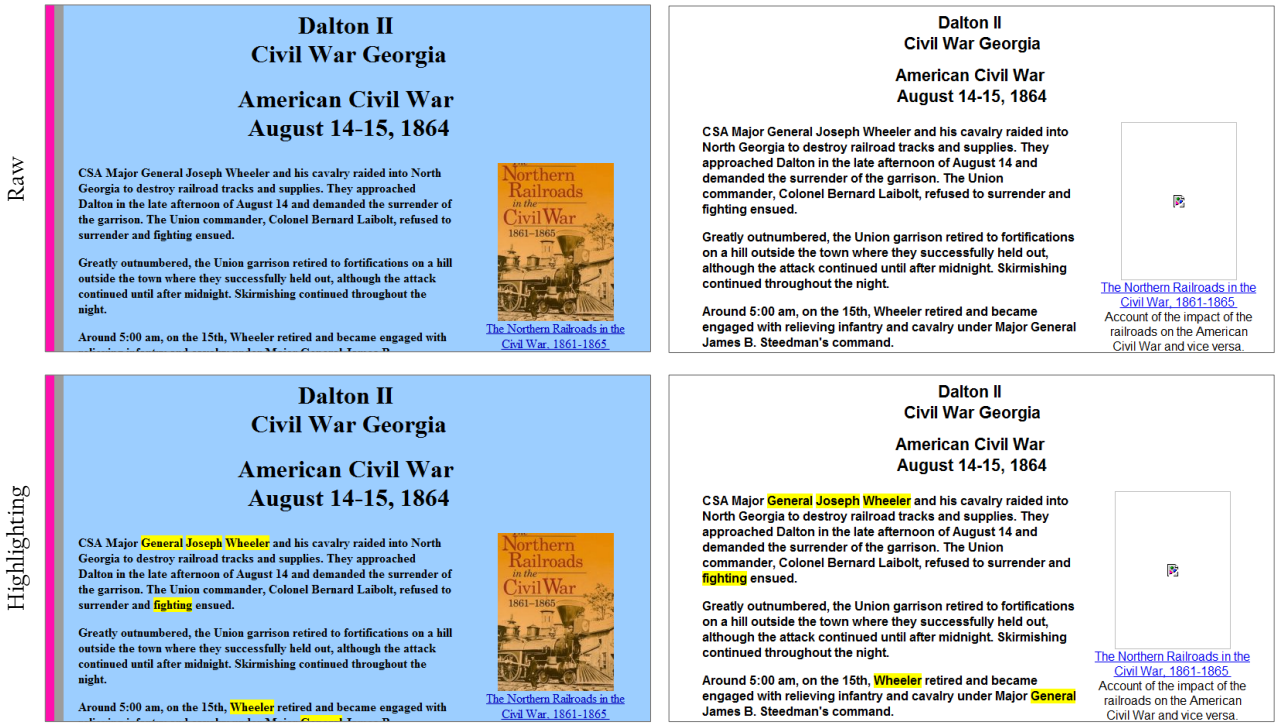


Figure 2. Document display modes for runs one and two (top) and run three (bottom).

2.2. Document Preprocessing

The documents displayed to workers were the result of a previous cleanup process to ensure smooth loading and safe rendering, embedding all contents within a single HTML file so no additional requests were made to any server. Documents displayed in the mode with images were processed as follows:

1. All hyperlinks were modified to point to a null URL, so that clicking on them would not trigger any page loading.
2. All CSS rules in external stylesheets were joined and put together in a single `style` tag within the document.
3. All CSS rules unrelated to style or layout were removed.
4. Unsafe or irrelevant HTML elements, such as scripts and applets, were removed too.
5. All HTML attributes unrelated to style or layout were removed.
6. All images were loaded and embedded within their `a` tag, using base64 encoding.

Documents displayed in the mode with just text underwent three more processing steps:

1. The source of all images was removed. That is, workers could see where images appeared in the document, but no image was shown at all (see Figure 2, right).
2. HTML attributes related to style were removed, keeping the ones related to layout.
3. Background colors were all set to white, font colors set to black, and the font family set to Sans Serif of size 11px.

Both document versions were modified for the third run to have key terms highlighted. We removed stopwords from each topic description, and obtained the stems of the remaining words. Then, we compiled all documents to judge for that topic, and for every word with a matching stem we set the background color to yellow and font color to black, wrapping the word with an HTML `span` tag (see Figure 2, bottom).

All these document versions were stored in our server and sent on demand whenever a worker clicked on a document tab header. Runs one and two allowed workers to see documents with and without images, and run three used those same two versions but with the topic key terms highlighted. Many workers preferred to use the just text version: 7 (24%) in the first run, 21 (25%) in the second one, and 12 (40%) in the third run. Apparently, a larger proportion in the third run preferred this version, because the highlighted terms can be more easily glimpsed with a black and white document.

2.3. Relevance Question

We explored two lines of work for obtaining the actual relevance judgments (see Figure 3). In the first run we tried to optimize for the binary relevance label, while for runs two and three we tried to optimize for the ranking labels.

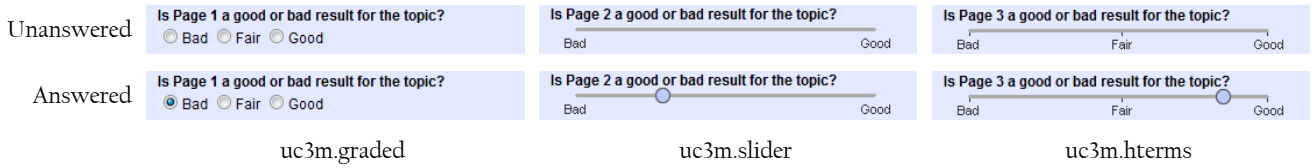


Figure 3. Relevance question widgets.

2.3.1 uc3m.graded

For the first run we used a traditional 3-point graded relevance scale where documents could be judged as *bad*, *fair* or *good* for the search topic (see Figure 3, left). Even though the Track asks for binary labels, using three levels provides more information to adjust for a probability-based binary label as well as for the ranking labels.

For the probabilistic binary labels we set the *bad* documents as 0, the *good* documents as 1, and then studied different probabilities for the *fair* documents. We tried values from 0 to 1, with increments of 0.1, and the results with the known judgments supported the choice of 1 for the *fair* documents too. For the ranking labels, we ordered documents by their relevance level, as reported by workers. We broke ties ordering again by the number of failures in quality control (see Section 2.4) and then by the time spent on each document (descending). We thus assumed that, given two documents with the same relevance judgment, the one for which workers failed the quality question or spent little time is less likely to be relevant. We tried other attributes and orderings, but they were marginally worse.

2.3.2 uc3m.slider

For the second run we investigated how to have workers provide an explicit ranking of the five documents in the set without much trouble. Preference judgments would make more sense for a relatively large number of documents, but for just five documents we did not consider it a viable option because of the additional cost. We also thought about using a 5-point relevance scale without ties, but such an option would not be very practical for little over just five documents, and still we would not be able to obtain a clear boundary between the relevant and not relevant documents for the binary scale.

Instead, we gave workers a slider widget they could use to give each document a certain amount of relevance (see Figure 3, middle), from 0 to 100, though workers could not see the actual value set. This would give us a complete ranking of the documents and something like a relevance distance between them: documents similarly relevant to the topic would have their slider handles close together, and documents too different in terms of relevance would have them far apart. Intuitively, workers would move the slider handle to the right for the relevant documents and to the left for nonrelevant ones, even form little clusters with the slider handles for documents of similar relevance. Using a simple clustering algorithm we could infer the binary relevance level out of these clusters.

In early trials the slider handle was shown in the middle from the beginning, and workers had to move it in either direction, even if they wanted it to be exactly in the middle. This had a clear advantage for binary labels: if the slider handle

were moved to the right side, even if just a little, we could assign a binary label of 1 because the right end of the slider was labeled as *good*. Similarly, if moved to the left side we could assign a label of 0. The downside was that some workers moved the slider handle just 5-10 pixels, being hardly informative for the ranking labels. Most probably, these were careless workers trying to make easy money, but they could as well be honest workers that just wanted to indicate mid-relevant documents. In contrast, some others tended to place the slider handles at the the end points, providing little information for the ranking labels too. Probably they just did not understand the slider widget and just clicked on the end labels. We finally decided to hide the slider handle at the beginning, as an indication that no value was set (see Figure 3, top), avoiding any initial bias too. Workers had to click on the slider bar to set an initial value.

For the ranking label we just used the slider positions provided by workers, and broke ties by the number of failures in quality control and then by the time spent on each document (descending), just like in the uc3m.graded run. For the binary label we studied several options:

- Threshold. All documents with a slider value larger than a threshold t would be assigned a binary label of 1 and 0 otherwise. We tried threshold values from 10 to 90, with increments of 10.
- Normalized. The probabilistic binary label is just the slider value divided by 100.
- Worker-Normalized. We compute the range of values given per worker for all documents (i.e. global minimum and maximum), and then normalize the slider values within that range. This option could correct for possible biases per worker, although it makes the unjustified assumption that all workers judge at least one highly relevant document and one highly nonrelevant. In addition, it could suffer from learning effects.
- Set-Normalized. The slider value is normalized between the minimum and maximum judgments in the set of five documents. Similarly, this option could correct for worker bias, although it might suffer learning effects and it assumes there is a highly relevant document and a highly nonrelevant document in the set.
- Set-Normalized Threshold. The Set-Normalized value is computed, and if it is larger than a threshold t we assign a binary label of 1 and 0 otherwise. We tried threshold values from 0.1 to 0.9 with increments of 0.1.
- Cluster. Given the slider values for all five documents in the set, we ran a k-means clustering algorithm for 2 clusters: documents in the left cluster are assigned a binary label of 0, and those in the right side are assigned a label of 1. Again, this option assumes there are both relevant and nonrelevant documents in the set.

These options were evaluated with the known labels of the test set provided for the Track, and we chose the Set-Normalized Threshold labels with threshold t=0.4.

2.3.3 uc3m.hterms

For the third run we followed the same line as in the second run, but we decided to include another label in the slider widget marking the middle point as *fair* (see Figure 3). Our hope was that workers would better understand that they had a wide range of possible values to set, not just the *bad* and *good* extremes. However, thy just set more values around the middle label as well. Again, we tried all options to generate the binary labels, and the Set-Normalized Threshold method with t=0.4 yielded again the best results.

2.4. Quality Control

There is no warranty that the answers provided by workers are correct (to the extent there is an objective correct answer at all in this task). We thus decided to implement some quality control mechanisms, which work at different levels:

- Worker level, where only workers that meet certain demographic criteria are allowed to work in the task. In the case of Amazon Mechanical Turk, they can be filtered by their percent of work accepted, geographic location, etc.
- Task level, where the task template is modified to incorporate some additional information about the individual worker response that could indicate misbehavior. This information can be implicit, such as the time needed to complete the task [Zhu and Carterette, 2010][Urbano et al., 2010] or behavioral patterns such as mouse clicks, scrolling and key strokes [Rzeszotarski and Kittur, 2011]. It can also be gathered explicitly, actually asking additional secondary questions with quantitative, unbiased and automatically computable answers [Kittur et al., 2008]. If giving correct answers to these questions required as much effort as answering the main question, we could assume that if the secondary answer is correct then we can trust the answer to the main question.

- Process level, where the task assignment process is modified so that workers receive examples for which the correct answer to the main question is known beforehand (i.e. trap questions), allowing us to assess how well the worker responses fit the known answers [Sanderson et al., 2010] and follow a learning process accordingly [Le et al., 2010].

The subset of known labels provided in the test set suggests the use of trap questions as the quality control mechanism at the Process level. However, we decided not to pursue this line for several reasons. First, these known labels would need to be generated in real evaluation settings, with the problem of choosing what topics (if not all of them) and what documents to judge, increasing the workload in a real evaluation experiment via crowdsourcing. Second, the labels would not necessarily be balanced across relevance levels like in the data used in the Track, so that many judgments could be needed beforehand to get a meaningful gold set to use as trap questions. Third, using these questions involve an overhead cost, whether adding just one document per set or having workers judge all five documents in a known set. Fourth, trap questions may ensure that workers understood and paid attention to those particular topics and documents, but not necessarily to all others.

Instead, we decided to investigate quality control mechanisms at the Task level. As an implicit measure we used the work time per document, taking into account both the HIT previewing and working times; and as an explicit measure we used a very simple reading comprehension test based on a selection of the keywords that best describe the documents. We also used the Worker level control provided by AMT.

2.4.1 Worker Level

We used some of the worker filters provided by Amazon Mechanical Turk. At first, we considered submitting different runs filtering by country, but given the tight deadlines we discarded this option in favor of having more workers. We did require all them to have a minimum of 100 total approved HITs, and a minimum percentage of approved HITs of 95% for the first and second runs, 98% for the third run. In addition, we limited workers to contribute a maximum of 50 sets (250 judgments). For the third run we also requested workers to have Amazon's Categorization Master qualification, but these workers were not attracted at all by our task, so we immediately removed this filter and started over. There is no way of knowing if any of the workers in the three runs does indeed have this qualification though.

2.4.2 Implicit Task Level: Work Time

Out HIT templates measured how much time workers spent on each document, rather than on all five altogether, summing the time spent with both display modes. As an implicit quality measure we counted the number of time failures, that is, the number of documents in the set for which workers spent less than 4.5 seconds. Previous trials with real AMT workers and colleagues indicated 4.5 seconds as a suitable threshold, even for documents that can be quickly identifiable as spam.

We did not use the work time in seconds reported by Amazon. In our experience, many workers like to preview HITs, and spend some time getting used to the particular contents of each one. Some workers solve HITs while previewing, and the time they spend after accepting the HIT is used just to select their answers and submit. As such, using only the work time can be very misleading, and some workers get very offended if their work is rejected on the basis of work time alone [Urbano et al., 2010]. Our HITs were implemented as external, that is, they were hosted in our own server. As such, we knew when workers previewed HITs. We included a script in the HIT template to report back to our server, every 10 seconds, how much time was spent on each document so far. When computing the number of time failures, we added this preview time to the work time to have a better estimation of actual time.

2.4.3 Explicit Task Level: Reading Comprehension

Our first approach for explicit quality measures was to modify documents by inserting a paragraph with syntactically correct yet nonsensical sentences, asking workers to detect where they are. However, some documents can be very easily identifiable as relevant or nonrelevant, so asking workers to spot the nonsensical paragraph would just be a waste of time and it would deteriorate worker performance (for example in a large Wikipedia article). In addition, automatically finding a suitable location to embed these paragraphs is certainly not an easy task. [Kittur et al., 2008] asked for the number of references, sections and images in Wikipedia articles, but in our case documents are very heterogeneous HTML documents that seldom have a common structure, so we discarded these questions as well.

Our choice was to implement a very simple reading comprehension test where workers had to tell which one of two sets of keywords better described the document (call this the positive keyword set). We designed a previous experiment where subjects were given documents and they had to provide a list with 5 to 10 keywords that they thought best described the document. We ran four trials with real AMT workers: one with no qualification required, with more than

95% approval, with more than 1000 total HITS approved and 95% approval, and another one with the Categorization Master qualification required. Another trial was run with four colleagues with varying expertise on Information Retrieval. We found that virtually all subjects in all five trials provided the most and/or the second most frequent terms in the document (excluding stopwords and considering two terms as equal if they had the same stem), and those who did not were clearly spammers.

Given that workers recognized the most frequent terms, we included in the positive keyword set the 3 most frequent terms plus 2 random terms from the next 5 most frequent ones; and for the negative keyword set we randomly picked 5 terms out of the 25 least frequent ones (see Figure 1, right). In addition, terms within keyword sets were shuffled. We decided to pick terms randomly to some degree because many document sets had duplicate documents or documents for which the most frequent terms were the same, which would have made the task very easy. As an explicit quality measure we counted the number of keyword failures, that is, the number of documents in the set for which workers did not select the positive keyword set.

2.4.4 Rejecting HITs and Blocking Workers

We used the number of time and keyword failures to decide whether to accept all five judgments in the HIT, and in case of rejecting it whether to block the worker or not. If a HIT was to be rejected, we extended it so that another worker could answer it, and when blocking a worker we did not reject all his previous work to be sure we met the deadline. Moreover, some documents might not have clearly important terms, so simply failing the question would not be that rare.

Action	Failure	uc3m.graded	uc3m.slider	uc3m.hterms
Reject HIT	Keyword	1	0	1
	Time	2	1	1
Block Worker	Keyword	1	1	1
	Time	2	1	1
Total HITs rejected		3 (1%)	100 (23%)	13 (3%)
Total Workers blocked		0 (0%)	40 (48%)	4 (13%)

Table 2. Maximum number of keyword and time failures allowed per HIT and worker.

As Table 2 shows, the first run was the most permissive of all three, and run two was the most restrictive. For instance, in the first run we rejected a HIT if there were more than 1 keyword failures or more than 2 time failures, and if a worker accumulated more than 1 HIT rejected because of keyword failures or more than 2 HITs rejected because of time failures, we would block him. One of the consequences was that the second run took much more time to complete, as much more work was being rejected. In fact, workers seemed to keep failing in 7 of the HITs from the second run, so we decided to stop extending those HITs when three workers failed in them. The labels for those HITs were computed by simply averaging the labels of their three workers.

3. Task II: Aggregating Multiple Judgments

The second task of the Track promoted research on quality control at the fourth level: Aggregation. These mechanisms modify the task assignment process so that questions are answered by multiple workers, resulting in redundant answers from which a consensus response can be computed [Snow et al., 2008]. There are simple mechanisms such as using the vote of the majority [Alonso et al., 2008] or comparing the distributions of possible answers [Urbano et al., 2010]; and more complex alternatives such as weighting answers by the worker trustworthiness [Le et al., 2010] or using statistical models that try to maximize the effectiveness of workers [Snow et al., 2008][Ipeirotis et al., 2010]. We followed the last line.

3.1. uc3m.wordnet

For this submission we used the GetAnotherLabel algorithm[3] to manage redundant answers [Ipeirotis et al., 2010]. Based on a set of examples with previously known answers and the labels given by workers to those and other examples, it can compute a certain probability for each unknown example to have a particular label based on the worker responses. In addition, it computes a confusion matrix and an overall expected quality for each worker. In early trials we found this algorithm to perform exceptionally well, but we did not make a submission just with it alone because it is not our work.

Instead, we decided to explore stepwise executions of the algorithm using various features. Let us assume a feature that can be used to categorize the topic-document pairs. Our idea is that workers could be more reliable in documents from one

[3] Actually, we implemented a C# port of the original Java software, available at http://code.google.com/p/get-another-label-dotnet/

category than another, so we ran the algorithm once per category, using only documents pertaining to it. We then compared the worker expected quality in each category with the overall expected quality using all documents, and if it was smaller we ignored that worker's judgments and executed the algorithm again. That is, we used only the best workers per category, and ran the GAL algorithm with them. The features we tried were:

- Topic category, as indicated in TREC's topic descriptions: closed or limited, advice, navigational, open-ended, etc.
- Subject of the information need: politics, shopping, people, geography, etc.
- Rareness of the topic. We looked up in Wordnet the terms in the topic descriptions, and if a topic had terms that did not appear in any glossary we categorized it as rare. That is, we assumed that some workers might have more difficulty with rare topics.

The binary labels are directly provided by the algorithm as the probability of having the relevant label, and the ranking labels can be assigned just by ordering documents according to these probabilistic binary labels. We tried the three features, and the topic rareness was marginally better than the others, so that is the one we used for our submission.

3.2. uc3m.rule

For the second run we trained a rule-based model using several features regarding worker quality, with each topic-document as a different example. Besides other features, we used the output of the GAL algorithm using all topic-document pairs, in particular, the confusion matrix for each worker. These matrixes are defined with several variables of the form π_{ij}^{w}, that is, the probability that worker *w* assigned the label *j* to an example whose actual label is *i*. The complete set of features is as follows:

- Number of relevant labels the topic-document received, divided by the total number of labels received. Using only this feature would be like using majority voting, possibly with bias correction.
- For all workers that labeled the example, the average correct to incorrect ratio when assigning a relevant label.
- For all workers that labeled example, the average correct to incorrect ratio when assigning a nonrelevant label.
- For all workers that labeled the topic-document, the average posterior probability of the document being relevant, that is, π_{11}^{w} if they said relevant (they were right) and π_{10}^{w} if they said nonrelevant (they were wrong).
- For all workers that labeled the topic-document, the average posterior probability of the document being nonrelevant, that is, π_{00}^{w} if they said nonrelevant (they were right) and π_{01}^{w} if they said relevant (they were wrong).

The model provided the binary labels out of the box, and the ranking labels were computed by the branch of the decision tree the topic-document fell under: the larger the positive to negative ratio of examples falling under that branch, the higher the ranking.

3.3. uc3m.svm

For the third run we trained a SVM model using the same features as in the second run. The model allowed us to easily rank documents and assign binary labels: positive scores indicated relevant documents (negative scores for nonrelevant), and the larger the score the more relevant the document.

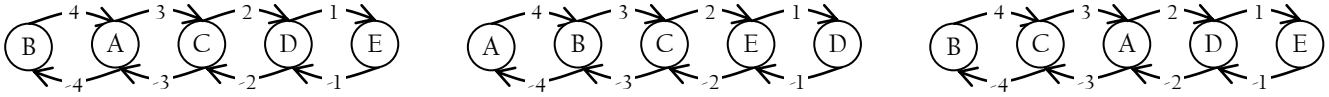
4. The TREC Crowdsourcing Track

We would like to propose an extension of the first task to incorporate the second one too. While the first task alone does make sense to better design task templates and the second task to further study aggregation methods, the truth is that in a real scenario both tasks are rarely run alone: we need individual judgments from task one to compute consensus in task two. The current Track design does not allow teams to include any information other than worker and label from task one into task two, hindering proper research on aggregation methods. For instance, we could use valuable information such as the time spent or demographic data, besides changing the assignment process based on information from other workers. In addition, providing just binary labels for task two seems unrealistic given that in the first task we could use many other types of judgments. In summary, it is our believe that the current Track design, while well promoting research on different parts of the crowdsourcing process, does not allow us to investigate full methodologies from task design to worker response aggregation. Therefore, we propose to keep task two and have task one produce not only individual judgments, but also aggregated ones with all the information available. To this end, the number of examples to judge could be reduced from ~2000 to ~500 to allow redundancy in task one but with a similar cost for participants.

Another point to study is how to evaluate runs. This year, the Track followed a different methodology than usual, because NIST assessors did not provide new relevance judgments to evaluate systems. The alternative used consists on computing the aggregated labels for each topic-document example through the vote of the majority, and use those labels as ground truth. One problem with this method is that the vote of the majority is too simple and thus offers low quality labels when compared to other methods [Sheng et al., 2008]. In addition, it is not clear to us whether the best results will be achieved by the actually best systems or by the average systems. In our experience, workers are biased toward the relevant label, probably just because of the topic terms appearing in the documents. This can also be observed when comparing Mark Smucker's judgments against the consensus for recall and specificity. As a result, if teams do not correct this bias somehow the vote of the majority will be biased too. Indeed, it is striking how much the numbers change when looking at the results against the vote of the majority and the results against the known labels by NIST, though this would also be caused by teams overfitting to the known labels.

Another concern is how to evaluate the ranking labels. The Crowdsourcing Track has the particularity that all documents are eventually "retrieved" by workers, inflating the AP and NDCG scores. Let us assume a document set where only one of the five documents is relevant, and a run that ranks this document at the very bottom. The AP score would still be 0.2, and if there were actually two relevant documents retrieved at the very bottom, the AP score would still be 0.325. That is, not only the lower bound is not zero, but it also varies across queries. This lack of a fixed lower bound could cause stability problems when averaging across workers [Sakai, 2007].

An alternative measure to avoid this problem is Average Dynamic Recall (ADR), used by the Multimedia IR community [Typke et al., 2006]. It basically computes, for each rank k, the proportion of retrieved documents that are in the ideal top k. Another alternative could be a rank correlation coefficient such as Kendall's tau, but it has its own issues [Carterette, 2009][Yilmaz et al., 2008]. The problem is that document swaps are equally penalized whether they occur at the top of the list or at the bottom, when swaps towards the top are worse in terms of user satisfaction. To avoid this, we propose to weight swap penalizations by the rank in which they occur. For instance, given the perfect ranking ⟨A, B, C, D, E⟩, let us define a directed graph where nodes are documents and edges are weighted: weights are larger towards the top of the ranking, positive when moving downwards and negative when moving upwards (see Figure 4). Then, for every ordered pair of documents we compute the length of the shortest path from one node to another. For instance, the distance between documents A and B would be -4 in the left figure and -4-3=-7 in the right figure. Summing up the path distances between all ordered document pairs we obtain an overall similarity score with similar interpretation to a correlation coefficient: positive scores represent similar rankings and negative scores represent reversed rankings (or swaps between highly ranked documents).



Swap towards the top, score=42/50=0.84 | Swap towards the bottom, score=48/50=0.96 | Longer swap towards the top, score=30/50=0.6

Figure 4. Examples of ranking similarity scores.

With this weighting scheme we get the desired measure behavior: swaps between documents far apart are penalized more than documents close together, and swaps involving documents towards the top of the ranking are more penalized than documents towards the bottom. For instance, if the first and second documents were swapped, the path length between them would be -4, but if it were the last two documents their distance would be -1. To normalize between -1 and 1, we just divide the overall score by the possible maximum: $n^2(n^2-1)/12$, where n is the number of documents.

References

O. Alonso, D.E. Rose, and B. Stewart, "Crowdsourcing for Relevance Evaluation," *ACM SIGIR Forum*, vol. 42, no. 2, pp. 9-15, 2008.

B. Carterette, "On Rank Correlation and the Distance Between Rankings," *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 436-443, 2009.

P.G. Ipeirotis, F. Provost, and J. Wang, "Quality Management on Amazon Mechanical Turk," *ACM SIGKDD Workshop on Human Computation*, pp. 64-67, 2010.

A. Kittur, E.H. Chi, and B. Suh, "Crowdsourcing User Studies With Mechanical Turk," *Annual ACM SIGCHI Conference on Human Factors in Computing Systems*, pp. 453-456, 2008.

J. Le, A. Edmonds, V. Hester, and L. Biewald, "Ensuring Quality in Crowdsourced Search Relevance Evaluation: The Effects of Training Question Distribution," *ACM SIGIR Workshop on Crowdsourcing for Search Evaluation*, pp. 17-20, 2010.

J. Rzeszotarski and A. Kittur, "Instrumenting the Crowd: Using Implicit Behavioral Measures to Predict Task Performance," *ACM Symposium on User Interface Software and Technology*, 2011.

T. Sakai, "On the Reliability of Information Retrieval Metrics Based on Graded Relevance," *Information Processing and Management*, vol. 43, no. 2, pp. 531-548, 2007.

M. Sanderson, M.L. Paramita, P. Clough, and E. Kanoulas, "Do User Preferences and Evaluation Measures Line Up?," *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 555-562, 2010.

A.D. Shaw, J.J. Horton, and D.L. Chen, "Designing Incentives for Inexpert Human Raters," *ACM Conference on Computer Supported Cooperative Work*, pp. 275-284, 2011.

V.S. Sheng, F. Provost, and P.G. Ipeirotis, "Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 614-622, 2008.

R. Snow, B. O'Connor, D. Jurafsky, and A.Y. Ng, "Cheap and Fast—But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks," *Conference on Empirical Methods in Natural Language Processing*, pp. 254-263, 2008.

R. Typke, R.C. Veltkamp, and F. Wiering, "A Measure for Evaluating Retrieval Techniques based on Partially Ordered Ground Truth Lists," *IEEE International Conference on Multimedia and Expo*, pp. 1793-1796, 2006.

J. Urbano, J. Morato, M. Marrero, and D. Martín, "Crowdsourcing Preference Judgments for Evaluation of Music Similarity Tasks," *ACM SIGIR Workshop on Crowdsourcing for Search Evaluation*, pp. 9-16, 2010.

E. Yilmaz, J.A. Aslam, and S. Robertson, "A New Rank Correlation Coefficient for Information Retrieval," *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 587-594, 2008.

D. Zhu and B. Carterette, "An Analysis of Assessor Behavior in Crowdsourced Preference Judgments," *ACM SIGIR Workshop on Crowdsourcing for Search Evaluation*, pp. 21-26, 2010.