Context Trees: Crowdsourcing Global Understanding from Local Views

Vasilis Verroios and Michael S. Bernstein*
Stanford University
{verroios, msb}@cs.stanford.edu

Abstract

Crowdsourcing struggles when workers must see all of the pieces of input to make an accurate judgment. For example, to find the most important scenes in a novel or movie, each worker must spend hours consuming the entire plot to acquire a global understanding and then apply that understanding to each local scene. To enable the crowdsourcing of large-scale goals with only local views, we introduce *context trees*, a crowdsourcing workflow for creating global summaries of a large input. Context trees recursively combine elements through written summaries to form a tree. Workers can then ground their local decisions by applying those summaries back down to the leaf nodes. In the case of scale ratings such as scene importance, we introduce a weighting process that percolates ratings downwards through the tree so that important nodes in unimportant branches are not overweighted. When using context trees to rate the importance of scenes in a 4000-word story and a 100-minute movie, workers' ratings are nearly as accurate as those who saw the entire input, and much improved over the traditional approach of splitting the input into independent segments. To explore whether context trees enable crowdsourcing to undertake new classes of goals, we also crowdsource the solution to a large hierarchical puzzle of 462,000 interlocking pieces.

Introduction

The fox knows many things, but the hedgehog knows one big thing. — Archilochus, 700 B.C.

Crowdsourcing collects parallel, isolated pieces of knowledge at scale. However, when the problem cannot be decomposed and contributors instead must see all the inputs to make a single decision, crowdsourcing struggles. For example, how might we crowdsource the summary of a novel? If each person looks at one paragraph, it is difficult to know how important the paragraph is relative to unseen ones. Worse, it is impossible to know about dependencies that might turn a minor detail at the beginning of the story into a major plot point once the climax is revealed. Crowdsourcing's strength — the isolation of small, independent subproblems — becomes its weakness when the subproblems are no longer independent.

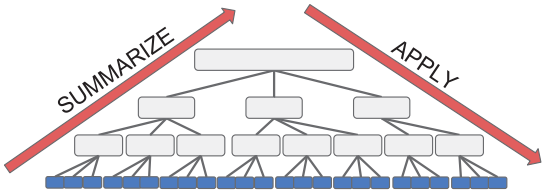


Figure 1: Context trees enable crowds to build up global summaries of many inputs (in blue), then apply those summaries back down the tree to ground their local decisions.

To succeed, the crowd must be able to act with global understanding when each contributor only has access to local views. This is a counter-intuitive proposal: how can a crowd make judgments about an entire story if nobody has the ability to see the big picture? Crowdsourcing techniques often sidestep this problem by avoiding information dependencies between tasks. Instead, they aggregate local contributions algorithmically so that no crowd member ever needs a higher-level view, as for example with clustering (Chilton et al. 2013; Bragg, Mausam, and Weld 2013; Tamuz et al. 2011; Gomes et al. 2011). When algorithms cannot aggregate the results, a single worker must do it instead (Kittur et al. 2011; Kulkarni, Can, and Hartmann 2012). These approaches cannot scale larger than what a single contributor can consume. If the task is iterative, workers can leave an annotated history for later workers (Lasecki et al. 2012b; 2013), but this again only succeeds if the history is compact enough to be consumed by the next worker.

In this paper, we focus on extractive summarization via crowdsourced microtasks — for example of novels and movies — as a representative problem requiring global understanding. In extractive summarization, the goal is to tag each element with an importance score and then filter to the top-rated elements and/or trim elements according to the their score. These summaries would allow a user, for example, to know which parts of their essay to cut or watch a movie at accelerated speed. Crowds have been able to shorten text (Bernstein et al. 2010), but only through uniform cuts and not through global consideration of which sections should be kept and which should be cut entirely. These

*This research was supported by the National Science Foundation under grant no. 1351131.

cuts can only be completed by simultaneously understanding the local context as well as how it fits into the larger picture. Solving such a problem could unlock goals such as the automatic retargeting of content to mobile devices; support for strategic decisions that involve consuming days, weeks, or months worth of news; or user-controlled resizing of a book or movie so that it can be consumed with as much or as little time as is available. As the amount of user-generated content continues to scale up, attention remains constant: it will be crucial that we can efficiently "compress" this content as easily as we generate it.

To solve problems that require global knowledge to make local decisions, we introduce *context trees*, a hierarchical summarization crowdsourcing workflow. The general approach is to *summarize upwards, then apply downwards* (Figure 1). In the upward phase, workers write a short summary of each raw input, then vote for the most accurate summaries and recursively write new summaries to aggregate several child summaries. The top layer then produces a summary for the whole storyline. In the downward phase, workers start at the top of the tree and use the final summaries to rate the importance of each child node. When this process reaches the bottom, workers rate the raw input based on the hierarchical summaries at all higher levels of the tree.

In practice, workers' context tree ratings are still only accurate when considered in relative terms between the items they examine. For example, workers who rate several scenes in a single subplot can only accurately rate those scenes relative to each other: their scores are not always grounded against the rest of the plot. Thus, we use ratings from higher tree levels to weigh opinions at lower levels. If workers at a higher level declared a subplot to be unimportant, the scenes inside that subtree will be downweighted appropriately.

In an evaluation, we use context trees to generate importance ratings of each scene in a 4193-word story and a 104-minute movie. We compare context trees to a baseline *local* approach where workers use a traditional crowdsourcing technique and rate each section independently, and to a *global* approach where workers read the whole story or watch the whole movie. We find that the performance of context trees is near that of workers who consumed the entire input, and significantly improved over the local condition. In most realistic crowdsourcing scenarios, where a global approach would be unrealistic to scale, context trees may produce near-optimal performance. Finally, we demonstrate that context trees can generalize to complex, multi-layered problems by solving a 462,400-piece interlocking puzzle.

Crowdsourcing workflows typically decompose a task into smaller and smaller subproblems (Kulkarni, Can, and Hartmann 2012). With this work, we suggest that it may be worthwhile to explore techniques that reverse this process, recursively combining higher- and higher-level actions to produce globally coherent behavior even though no worker has bandwidth to see the whole input.

Related Work

Crowdsourcing research often solves complex tasks by intelligently splitting and recombining the outcome of simple tasks (e.g., crowd entity resolution (Wang et al. 2013; Whang, Lofgren, and Garcia-Molina 2013) and clustering (Gomes et al. 2011; Tamuz et al. 2011; Biswas and Jacobs 2012)). We extend this work by focusing on problem domains where all obvious decompositions lead to simple tasks that are impossible to answer accurately without having a global view of all inputs. Context trees contribute a mechanism for having the crowd author such global views along the way to solving a large distributed task.

Categorization is one clear case where algorithms can produce globally consistent results without any worker seeing a global view. Cascade (Chilton et al. 2013) and DELUGE (Bragg, Mausam, and Weld 2013) build a multi-level categorization of items by having workers suggest and vote on the categories of each item. Crowds can also produce globally consistent coding of qualitative data if they gather sufficient domain knowledge (André, Kittur, and Dow 2014). Context trees may be an effective method of generating these global constraints, or giving workers a sense of the larger qualitative dataset before coding.

Workers can also take an active role in decomposing tasks and aggregating them later. CrowdForge shards the problem into parallelizable tasks and then aggregates their output into a meaningful single outcome, e.g., an article (Kittur et al. 2011). Turkomatic also follows a divide-and-conquer approach, recursively dividing each task into smaller and smaller sub-components (Kulkarni, Can, and Hartmann 2012). These papers demonstrate that it is possible to recursively *decompose* complex work: context trees aim to recursively *combine* smaller tasks into larger perspectives instead.

Crowd-powered systems are another domain where the crowd must act in a globally consistent way. Otherwise, the crowd agent might be mercurial and act one way now, then change opinion in the next round. To accomplish this, crowds can pass along memories to each other so that new untrained workers can effectively replace the experienced workers that leave (Lasecki et al. 2012b). Using similar principles, Chorus (Lasecki et al. 2013) builds a global representation of a long chat conversation by maintaining a record of the most important points of the conversation. Itinerary planning (Zhang et al. 2012) and real-time captioning of spoken language (Lasecki et al. 2012a) apply similar mechanisms. Still, these mechanisms are well-suited to tasks where this additive global representation can be understood quickly by a single worker. Context trees might enable these tasks to scale up to much larger problem spaces.

Problem Statement

Our goal is for crowds to be able to act with global knowledge even though each crowd member can only see a small subset of the input. In this section, we lay out content summarization as an instance of this general problem and sketch out the main challenges of interdependence and parallelism.

In summarization, we begin by splitting up a large piece of content such as a book or movie into many small *atoms* such as chapters or scenes. In most scenarios this subdivision can be accomplished automatically through scene detection algorithms. We do not assume a perfect splitting of the content into atoms: even when scenes are split poorly, they will be recombined via context trees.

The final objective is to produce an importance rating for each atom. An algorithm or worker can then generate a summary by filtering to the top-k atoms, or by trimming each atom proportionally to its rating. This framework is similar to *extractive* summarization in natural language processing (Gupta and Lehal 2010).

Our running example consists of a short story called *Hard Feelings* by Barbara D'Amato, which is 4193 words long. The novel is split into 29 atoms (Table 1): the average atom is 145 words and 1.5 paragraphs. Atoms were split at paragraph boundaries; no atom split within a paragraph. The main characters of the story are two police officers who are accused of leaving a person to die in an apartment that had caught fire. The female officer claims that the person was already dead when she reached him — his forehead was cold to the touch. On the other hand, the male officer states that the person was still warm when the he touched the victim, implying that they abandoned him. In the climax, the story reveals that the person was in fact already dead. The male officer had been touching something cold before examining the victim, while the female officer had been touching something hot. Due to the contrast, the male officer felt the victim to be warm, and the female officer felt the victim to be stone-cold. All of the important events leading to the final conclusion are shuffled over the storyline, making it very difficult to understand the importance of each atom when isolated.

Challenges

There are two basic challenges for any potential solution to rating the importance of each atom in this story.

Quality and parallelism are in tension. It is difficult for a human reading part of the story to understand the current atom's meaning and relative importance. For instance, a gun-shooting scene may be a key component in a political movie or one of the many meaningless gun-shooting scenes in a military movie. While a single contributor will have the context to make globally consistent decisions, that person might need hours, days, or months to process a long storyline, for example a ten-season TV series or the popular fanfiction Harry Potter and the Methods of Rationality, which spans roughly 2000 paperback pages. Parallelizing across many workers would complete much faster, but also sacrifice the background necessary to make accurate decisions.

Importance ratings are only reliable locally. Even with global summaries at hand, workers' ratings are typically only reliable relative to their other ratings and cannot easily be averaged to produce a global ranking. For example, workers are often positively biased toward the importance of the events they spent time inspecting (Norton, Mochon, and Ariely 2012). The effect is that many nodes end up with highly clustered, blandly positive ratings. In addition, workers will use different ranges of the rating scale (Herlocker, Konstan, and Riedl 2002). For example, in a 7-point Likert scale, one worker may give 7s to important atoms and 5s to non-important ones, while another worker may use 5 as her highest rating and 1 as her lowest one.

Given these challenges, an efficient solution must provide mechanisms so that workers can build a global view while preserving a high level of parallelism. Simultaneously, an effective solution must correctly normalize biases in individual workers' ratings.

Context trees

Context trees construct hierarchical summaries where each node describes the main events taking place in its child nodes and labels each child node with a weight signifying its importance to the global input. Specifically, each tree node stores two pieces of information: *a)* a text summary of the events taking place in the node's descendant atoms, and *b)* a rating for the most important event in the node's descendant atoms. The summary *(a)* is generated in an *upward phase* while the tree is built, and the rating *(b)* is generated during a *downward* phase.

Upward Phase

The upward phase, which generates the summary, requires two parameters. The first parameter is b, the desired branching factor of the context tree. In practice, there may be a different b for the leaf layer, because it may take more or less time to view atoms and summarize them than to combine existing summaries. The second parameter is r, the replication factor, which determines how many workers summarize or vote on each node. In future work, r might be adaptively determined, e.g., (Sheng, Provost, and Ipeirotis 2008).

In the mystery novel example, we use a branching factor b of 4 in the bottom layer and a b of 3 in the upper layers (Figure 3), so each worker sees 4 atoms or 3 previous summaries and writes a meta-summary. We used an r of 4, meaning that 4 workers provided input on each node.

In addition to writing and voting on summaries, the upward phase asks workers to vote on the most representative atom (leaf) for that subtree. The most representative atom propagates from the child nodes to the parent so that workers processing the upper layers of the tree can take a look of the raw content and not just read text summaries of other workers. The purpose of this early voting is to provide a glimpse of the actual content to the workers writing the summaries. The votes are discarded once the upward phase is complete.

The worker interface thus presents three questions:

1. *Vote on the most informative summary for each child node* (non-leaf nodes only): workers see all r summaries for each child node, giving them several perspectives on the content, and vote on which one is best.
2. *Summarize the events described in the child nodes* via a text summary.
3. *Pick the most representative of the atoms propagated to the child nodes*: This atom propagates to the parent node so that workers can see the raw content if they wish.

In practice, workers may reference the same event or entity in different ways, for example calling a character either by her first or her last name. This lack of alignment can make it more difficult for workers in the higher levels of the tree to combine nodes. So, we ask the user or requester to seed the system with canonical names for any major entities or characters, and workers must use the canonical name whenever they refer to that entity. In the future, we believe that this step

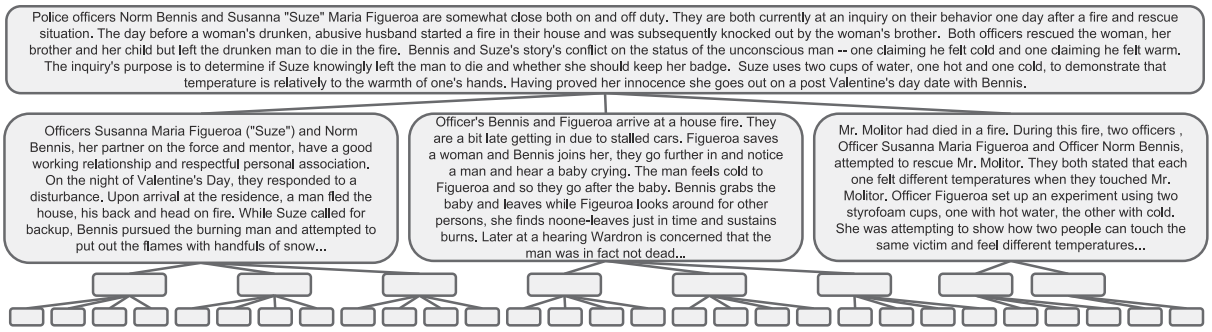


Figure 2: Summaries generated in the upper layers of the context tree for the mystery novel *Hard Feelings*.

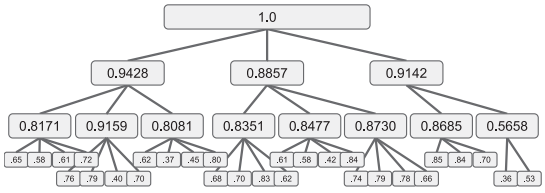


Figure 3: Context trees sample rating for the mystery novel *Hard Feelings*, after the downward phase.

could be automated, e.g., using named entity identification (novels) or face recognition (movies).

When the tree grows a root node, the upward phase has produced a global view of the storyline in a hierarchical form. Figure 2 shows the higher level summaries for the mystery novel. This hierarchical summary allows workers to understand the storyline at any level of granularity. Our next task is to use this global context to rate the local views.

Downward Phase

The downward phase applies the hierarchical summaries to help a second set of workers make informed ratings of the importance of each tree node. To this end, our approach must account for workers' biases and their differing use of the rating scale. The main principle of our rating process is to weigh each node in comparison to its siblings. In particular, we present the summaries on the path from the root node to the current node, then ask each worker to rate the importance of the current node's children. Starting from the root summary and working her way down to the current node, the worker can better understand the events she has to rate.

Each crowd task corresponds to a non-leaf node of the tree. The task interface asks workers the two following questions for each child of the non-leaf node:

1. *Given the overall summary and the summary of this child node, describe the most important event in this node.*
2. *Rate the importance of the event you described (*1 *to* 7*).*

A more straightforward approach would be to ask workers to rate the entire node instead of just the most important event. However, there is no forcing function that ensures workers have read the summaries, and workers ratings' are affected by how detailed or brief the summary is. Asking workers to describe the most important event in the summary ensures that they have read and thought about it (Kittur, Chi, and Suh 2008). It also focuses the workers on the content of the event rather than how concise or not the summary is. Just as in the upward phase, we use a replication factor r in order to get more than one rating for each node of the context tree.

We introduce a rating normalization that compensates for the fact that workers may not correctly balance the importance of their own subsection against other sections. In addition, different workers use different ranges on the rating scale, which makes it more difficult to produce an aggregated ranking. We normalize each child node rating to $[0, \mu_p]$, where μ_p is the mean normalized rating that other workers gave the *parent* node. In this way, we ensure that all workers have been normalized to the same range, and that the rating for a child is no greater than the importance of the parent, so a subtree cannot be more important than its parent. For instance, suppose a worker rates three child nodes 3, 3, and 6, and the normalized rating for the parent node is $\mu_p = 0.5$. We normalize first child's rating into $\frac{3}{6} * \mu_p = 0.25$, second child's rating into $\frac{3}{6} * \mu_p = 0.25$, and third child's rating into $\frac{6}{6} * \mu_p = 0.5$. The final rating for each child is the average over the r ratings. The downward phase can run across all nodes in parallel: rating normalization can take place after all of the ratings are collected.

Figure 3 gives a sample rating for the mystery novel while Table 1 depicts the top-5 short-story atoms referring to this rating; using checkmarks in column CT.

Evaluation

In this paper, we suggest that crowds can successfully create hierarchical summaries via context trees, to make decisions requiring global knowledge even though each contributor can only see a small piece of the whole. In this section, we evaluate our claim by applying context trees to rate the importance of 30 atoms in a 104-minute movie and 29 atoms in a 4193-word story[1]. We compare context trees to a *local* condition where each worker can only see a small number of atoms and to a *global* condition where each worker sees the entire input. We compared these three approaches to ratings collected by paid contractors who had experience in movie and novel reviews and saw the entire content before rating.

[1] http://hci.stanford.edu/publications/2014/contexttrees/

Atom	G	CT
Their commander, Sazerac, sat with them in the waiting room. He was as unhappy as they were. Finally he spoke. “There’s no way I can stop this. But it bothers me. I wouldn’t have figured you for a shirker, Figueroa.” - “What do you mean?” - “Don’t you realize how they see this?” - “Yes, boss. They know that Bennis and I have two different stories about last night, and so they think one of us is improperly describing the case. So they think one of us is lying. Which would be a reprimand.” - “NO, Figueroa. It’s not that minor.” - “Minor! I’ve never had a reprimand and I don’t intend to have one now! Uh. Sir.” Sazerac sighed. “Listen to me. We’re talking separation from the department. Maybe prosecution.” - “For what?” - “They think you left that man to die in the fire, Figueroa. And made up your story later to cover up. To make it seem he was dead.” - Bennis groaned. “But he was dead, sir! Figueroa would never- ”	✔	
There was smoke coming out around the top of the door. Bennis felt the door to see if it was cool, which it was. The last thing he wanted was to start a backdraft. Then he backed up to take a kick at the door. But that instant it opened and a man came running out. His hair and jacket were on fire, and he was screaming. He didn’t even see the two cops, but crashed frantically down the stairs. Bennis spun and went after him, knowing Figueroa would put out the emergency call to the dispatcher. He raced down the stairs three at a time and still couldn’t catch up with the terrified man until, leaping, screaming down the cement steps outside the front door, the man fell. The fire on his hair and jacket had spread. Bennis rolled him over in the snow three or four times. Then, thinking to chill the charred flesh and prevent further burn damage, he grabbed up handfuls of snow and slapped it all over the man’s head and back.,	✔	✔
The hall was still cool and the air in it was fresh, so she pushed the woman out and yelled after her, “Warn your neighbors!” There was no time to make sure she did. Suze Figueroa saw Bennis coming up the stairs. She yelled, “There’s a man inside.” On hands and knees she crawled back in, touching the hot wall to make sure where she was going. She found the man, but at the same instant she and Bennis heard a baby start to scream. Bennis was next to her now and he slapped the man’s cheek, but the man didn’t move. Figueroa felt the man’s forehead. Smoke swirled above him, but he was lifeless and cold. The baby screamed louder, from a back bedroom. “I gotta get the kid,” she said to Bennis. She wasn’t sure he could hear over the roar of the fire, but then she found him following her as they crawled to the bedroom. “How many kids?” she yelled. There was just one crib. Bennis stood up, grabbed a little girl out of the crib, put her solidly under one arm, and ran like a quarterback for the front door. Suze stayed to check for another child.	✔	✔
He and Wardron held each other’s eyes for two or three seconds. Figueroa said, “Sir, may I ask what the circumstances were that led up to the fire? We came into a situation midway-” - “Which is no excuse.” - “I’m not suggesting that it is. But the fire obviously started in that apartment, and there were three adults inside who didn’t seem to have made any effort to put it out. Why was Mr. Molitor lying in the middle of the living room floor? If he’d been shot, for instance, I would think I’d be entitled to know that. It certainly wasn’t the smoke that killed him. He was down on the floor where the air was good.” - “I don’t suppose there’s any reason not to tell you. We have a reasonably full picture, from the statement of the woman and the statements of the neighbors.” The Molitors had begun fighting in the afternoon-a husband, his wife, and the wife’s brother. Fighting and drinking, and drinking and fighting. Among the burned remnants of their apartment were dozens of beer cans and the fragments of two bottles that had contained scotch.,		✔
Wardron added, “The entire building was engulfed when the fire department finally made it. Shortly after that, the top three floors of the structure collapsed into the basement. What was left of Mr. Molitor looked a lot like a blackened pipe cleaner.” Figueroa had been staring at the tabletop. “Wait!” she said suddenly. She knew that wasn’t the way to talk to the brass, and said in a quieter voice, “If you’ll give me a minute, to go get something, I think I can explain what happened.” She got up. Wardron said, “You can explain it right here.” - “If I may leave for just a minute, sir, I can demonstrate.” - “One minute, then.”		✔
Figueroa took two Styrofoam cups from the table that held the coffee urn and was back in less than a minute with two cups of water. “Would you put the fingers of your left hand in one of these and the fingers of your right hand in the other, Deputy Wardron?” - “No. Explain to me what you think you’re trying to do.” - “Well maybe Commander Sazerac will, while I explain. We can always repeat it.” Sazerac, intrigued, did as she said. “One cup is hot water and one is cold. On the night of the fire, Officer Bennis had patted snow all over the brother, outdoors, and then ran back into the burning apartment. While he was doing that, I was pulling the woman out of the kitchen. She was hot to the touch and felt like she was starting to blister. When I came back, I felt along the hot wall. The instant Officer Bennis returned from outside, we both touched Mr. Molitor.”,	✔	✔
Commander Sazerac said, “I begin to see.” - “Mr. Molitor was dead, but only ten to twenty minutes dead, so his skin was probably about the temperature of mine today. Commander, will you use both hands to touch my forearm?” Sazerac did so. He smiled. “Amazing. Your arm feels warm to my right hand and cold to my left.” Sazerac turned to Wardron. “The same arm,” he said. “And it feels entirely different.” He gestured to Wardron. “Want to try it?” Figueroa and Bennis sat in their squad car. Bennis said, “Reminds me of this case I had once.” Figueroa sighed loudly, but Bennis knew she liked his stories. “Guy decided to rob a fraternity house late at night, on a night when there had been a late snow. Flat, untouched snow leading up to the door. So he says to himself ’If I walk in backward, they’ll think it was somebody from inside who stole the stuff, because there won’t be any tracks leading in.’ ” - “Not a bad plan.”	✔	

Table 1: Top-5 atoms for the novel *Hard Feelings*, from the ground truth (column G) and from context trees (column CT).

Method

We recruited workers from Amazon Mechanical Turk (AMT, www.mturk.com) to rate the importance of atoms in the movie and the short story. The short story, *Hard Feelings*, was described previously. For our second dataset, the movie *The Master Plan*, workers produced the following top-level summary: “*Kristie is a teen aged girl caught in the middle of an old conflict between her scientist grandfather and her fundamentalist father. She appears to be deeply religious herself, but is constantly shut down and drowned out when she tries to discuss her beliefs with anyone. Her situation comes to a head when her grandfather dies alone, shortly after her father has forbidden her to ever see or speak to him again. She finds comfort in her own faith, but her (also deeply religious) mother is shocked when she hears that faith articulated in terms of heaven being better than earth. The movie ends with Kristie dealing with the conflict between her own belief and understanding and that of others.*”

In the *local* condition, each worker is presented with a small portion of the full movie/story – four consecutive atoms – and is asked to give a 7-point Likert rating for the importance of the atoms she was presented with. Each worker can only do this task once. On the other hand, in the *global* condition, we paid workers to watch the whole movie or story and rate all the atoms. In many situations, a global condition is impractical — thus motivating context trees — but it is important to establish the global condition as an optimal result. In both the global and local conditions, each atom is rated by four workers and its final importance rating is the average of the four responses. The context tree used a replication factor, r, of four workers.

Are the benefits of context trees entirely from the upward phase, which generates the top-level summary? How much does the rating adjustment in the downward phase matter? To answer these questions, we include an additional *ablation* condition, which stops halfway through the context tree: we provide the same root summaries used by our context tree runs but do not execute the downward phase. Instead, we ask workers directly for a 7-point Likert rating of each atom’s importance, given the top-level summary. In other words,

after the upward phase, the ablation condition will run the same task with local condition with the additional inclusion of the top-level summary from the context tree.

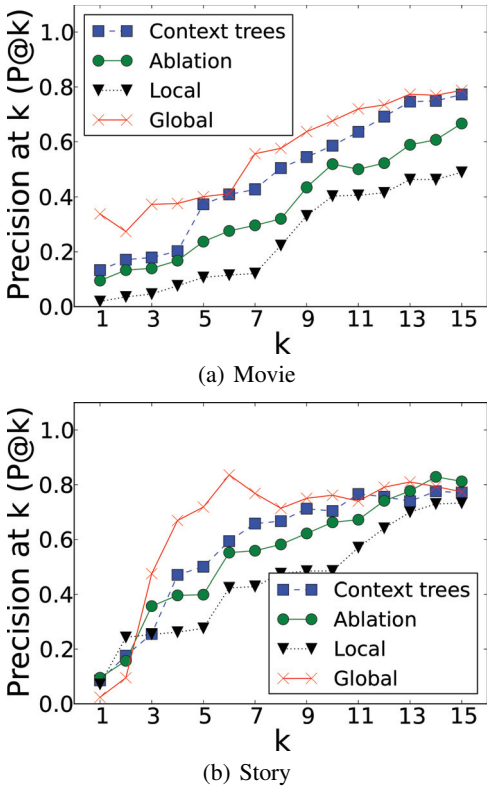
For all approaches, each node's final rating is the average over four workers' ratings. However, so that we might understand stability in this evaluation, we collected ten, instead of just four, ratings per node. Then, we generate $\binom{10}{4} = 210$ different outcomes (samples) by picking every possible combination of four ratings. For example, one context tree sample might only include the second, sixth, ninth, and tenth workers' ratings in the downward phase. This sampling process takes place only in the downward phase. Since written word and voting of the best summaries is used in the upward phase, we expect the quality of the best summaries in each tree node to be approximately the same across different runs of the upward phase.

For the upward phase of context trees, the price we set for a bottom layer task that involved the atoms, was \$1.50 for the movie and \$0.70 for the story. A higher layer task in the upward phase includes, in addition, the reading of summaries from the lower layer and requires more work. Hence, a higher layer task was priced at \$3.00 for the movie, and \$1.40 for the story. All downward-phase tasks were priced at \$1.50 for the movie and \$0.70 for the story. These values were tuned to optimize quality and throughput. With the above pricing and four answers per task, the overall cost of running context trees was \$210 for the movie and \$70 for the story. On the other hand, the overall cost for local was \$60 for the movie and \$22 for the story, and for global, \$80 and \$40, respectively. We required that workers completing the tasks have at least a 97% approval rating on over 500 tasks and live in the United States.

Ground truth We constructed ground truth importance ratings for each atom by recruiting 4 experienced freelancers from Elance (www.elance.com) to consume the entire film and story and then rate each atom. Doing so allows us to distinguish the quality of the global condition, which used AMT and where workers may slack, from a platform where we trust the output and could specifically hire workers who have experience with movie reviews. Workers sorted the top-15 atoms in each story based on their importance. Any atoms that were not included in the top-15 tied in the last position of the list. The overall cost for the Elance reviews was \$160 for the movie and \$80 for the story.

The average Spearman correlation across the 6 pairs of raters is 0.743 for the story and 0.855 for the movie, indicating a strong agreement between raters' rankings. The correlation was higher for the movie because it had more atoms clearly serving artistic purposes rather than plot; all raters ranked these atoms at the bottom. Our ground truth consists of the averaged ranked list of atoms from the 4 raters.

Measures We measure the precision at k ($P@k$) and Spearman's rank correlation between the sorted list of atoms each method produces and ground truth. Precision at k measures the percentage of scenes that agreed with the ground truth ranking of top-k scenes.



(a) Movie

(b) Story

Figure 4: $P@k$: Context trees' performance is close to the global condition where workers see the entire content, when they can choose $k > 5$ atoms. Context trees' improvement over only local views is between 50% and 300%.

Results

We first quantify the quality of context trees compared to the local condition where workers have limited insight into the larger story and to the global condition where the AMT workers have supposedly seen the entire film or read the entire mystery novel. Figure 4(a) depicts, for the movie, $P@k$ on the y-axis and $k \in [1, 15]$ on the x-axis. All datapoints are the average $P@k$ over the 210 samples for each approach.

Context trees' improvement over the local condition is between 200% and 300% when k is below 9, and between 50% and 100% for $k \in [9, 15]$. In other words, context trees were particularly effective when fewer scenes had to be selected. Moreover, context trees' precision is very close to global for $k > 4$. The average standard deviation across samples for all k was 0.136 for context trees, 0.110 for local, 0.141 for global, and 0.132 for ablation.

Figure 4(b) depicts the $P@k$ (means over 210 runs) for the story. Context trees' improvement over local is not as high as in the movie, still, when k is lower than 12, context trees precision is 50% higher than local on average. In addition, context trees' margin to global remains below 0.05 for $k > 7$. Note also that all approaches have a higher precision for almost all k values, compared to the movie. This improvement is because it is more difficult to realize which atoms contribute to the plot when watching the movie com-

pared to reading the story, especially when having just a local view. The average standard deviation across all k for the story was 0.127 for context trees, 0.119 for local, 0.105 for global, and 0.145 for ablation.

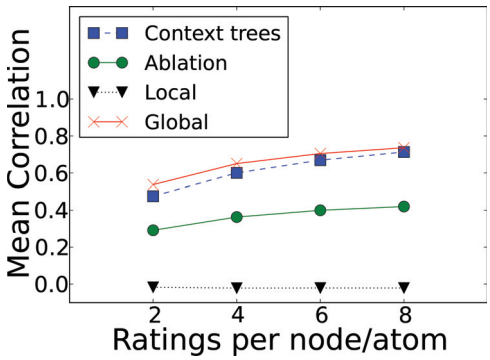
The ablation condition performs near the precision of context trees for the story, but remains much lower than context trees for the movie — further evidence that the movie dataset is more difficult. Specifically, in Figure 4(a), ablation's precision sits between the precision for the local and context tree condition. Since a video is worth a thousand (or maybe a million) words, the top-layer summary inevitably leaves out a number of important plot details. In this case, the benefit of applying the downward phase can be substantial. On the other hand, the top-layer summary for the story mentions briefly almost all of the important pieces of the plot and, hence, workers can rate accurately the importance of each atom in the ablation condition. The benefits of the downward phase are also measured by the correlation with the ground truth rankings in the next section, and are further emphasized in the puzzle experiment to come.

When $k < 6$, the precision of all approaches is low. In both cases, this is because the top five atoms are all critical elements of the plot, roughly equivalently important. So, there is significant disagreement on their ordering, and it is more difficult for any approach to do well on average.

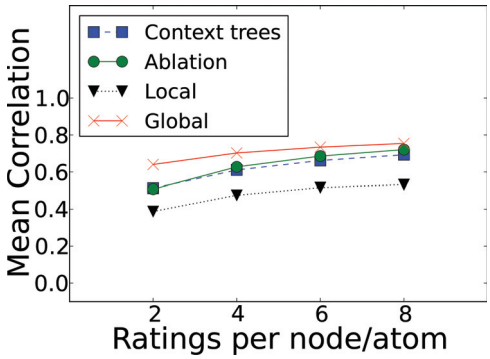
In order to examine the statistical significance of our results, we computed bootstrapped confidence intervals for the difference in $P@6$ between each pair of conditions. We chose $P@6$ as the dependent variable because it represents significant extractive compression ($\frac{6}{30} = 20\%$). Bootstrapping allows us to avoid prohibitive costs while testing these differences by repeatedly sampling from the 210 samples per condition. In our bootstrap, we computed 1000 bootstrap estimates using 100 samples to generate each estimate. We used 99.2% confidence intervals, which corresponds to $p = .05$ adjusted using Bonferroni correction to take into account familywise error across six comparisons. All pairwise differences are statistically significant (e.g., the confidence interval does not contain 0.0) except for the difference between global and context trees for the movie, and the difference of context trees and ablation for the story.

Effect of the number of ratings Figures 5(a) and 5(b) plot the average Spearman rank correlation between each condition and the ground truth when 2, 4, 6, and 8 ratings are used for each atom or tree node. For both the movie and the story, context trees increases its correlation with the ground truth at a higher rate, compared to global and local, as the number of ratings rises. These results suggest that once context trees have enough ratings for each node, they are practically equivalent to global. Once again the results suggest that the movie is a more challenging dataset: local does not improve with more ratings, and ablation and context trees are separated by a significant margin in Figure 5(a).

Cost To gain this increase in effectiveness, context trees require more work. The cost of context trees depends on the values of the replication factor, r, the branching factor, b, and the size of the content to be summarized. Here we provide a simple analytical evaluation for the cost of context



(a) Movie



(b) Story

Figure 5: Spearman rank correlation with the ground truth rankings. Context trees' correlation increases more rapidly than other approaches as more ratings become available.

trees. A single atom reaches the root of the tree, b atoms reach the children of the root, b^2 atoms reach the children of the children of the root, and so on. Initially, let us assume that there are $n = b^m$ atoms. Hence, workers need to watch/read n atoms in the bottom of the tree and $\sum_{i=0}^{m-1} b^i$ atoms on the higher layers of the tree. Overall, the total number of atoms workers read/watch in the upward phase is: $n + \sum_{i=0}^{m-1} b^i = n + \frac{b^m - 1}{b-1} = n + \frac{n-1}{b-1}$. In general, we assume that the time commitment for reading and writing summaries during the upward phase must be approximately the same with the time needed to read/watch the most representative atoms. Therefore, the overall effort to read and write the summaries is approximately the same with the effort to read/watch $\sum_{i=0}^{m-1} b^i$ atoms.

In the downward phase, we again assume that the effort to describe the most important event of a node and rate it, is approximately the same with writing a summary for that node. Overall, the effort needed for the downward phase is approximately the effort needed to read/watch $\sum_{i=0}^{m-1} b^i$ atoms, for the higher layers of the tree, plus the effort to read the initial n atoms in the leaves of the tree.

Taking also into account the replication factor r, the total effort for context trees to summarize n atoms of T minutes each, is $T * r * (2n + 3\sum_{i=0}^{m-1} b^i) = 2Trn + 3Tr\frac{n-1}{b-1}$. On the other hand, techniques such as local need a single pass over the n atoms. Therefore, when we ask the opinion of r

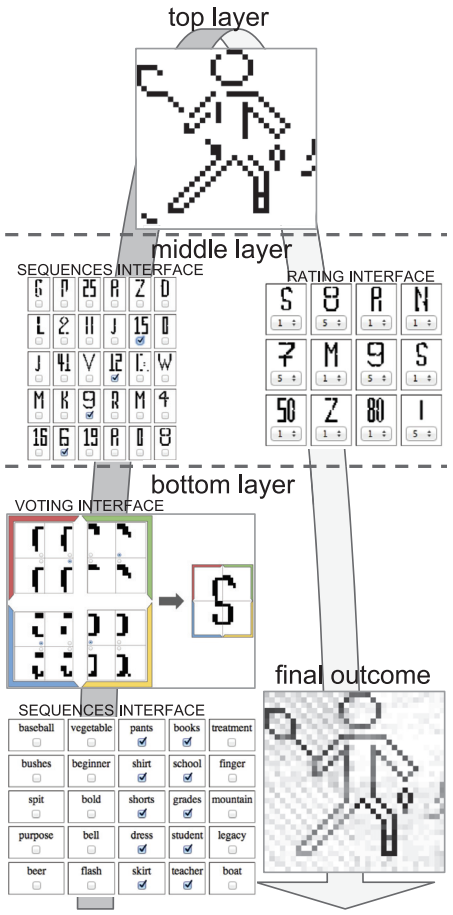


Figure 7: Context trees applied to the hierarchical puzzle: local puzzle solutions percolate up to create a draft top-level solution, then pass back down to fix the local puzzles.

workers, local requires Trn man-hours. For instance, when $r = 4$, $T = 3$ min, $n = 30$, and $b = 3$, for a rate of \$10/hour, local costs \$60, while context trees cost $120 + 87$ =\$207. In the general case, we can say that for a b close to 4, the cost of context trees is three times the cost of local.

Hierarchical Pattern Detection

Context trees offer the opportunity to crowdsource complex new tasks that require global understanding. Summarization is an obvious application, but context trees also allow large groups to work on other information tasks where useful data can be passed up the hierarchy, assembled, and then passed back down to inform local decisions. Stacked multi-layer puzzles, for example, become feasible with this approach.

We thus crowdsourced a puzzle solution to a 462,000-piece puzzle that operates simultaneously at three stacked levels of hierarchy: encoded within a 680×680 grid of words is a 34×34 grid of numbers, and encoded within that grid of numbers is a picture. Figure 6 shows a 5×5 portion of the grid at the bottom layer of the puzzle. There are two word sequences that appear: *a)* horse, dog, cat, giraffe, and *b)* cloud, rain, sun, mist, snow. Any word that is highlighted as part of a sequence turns on a "pixel" in the middle layer. The other pixels remain off in the middle layer. In this example, the turned-on pixels draw the letter "S" in the middle layer. The digits and letters from the middle layer form, in turn, arithmetic and letter sequences. In the 7×7 portion of the middle layer, in Figure 6, two sequences appear: *a)* 3, 5, 7, 9, 11, 13, 15, 17, 19, and *b)* D, F, H, J, L. As in the bottom layer, the sequences turn on or off the respective pixels for the next layer. In this case, the pixels turned-on form the racket of the tennis player in the top layer image in Figure 6.

This puzzle has some similarities with summarization tasks: the objective is to incrementally pass along important "bits" of information that shape higher layers' views. Then, the top-layer image can help workers in the lower layers refine their guesses. Nevertheless, there is an important difference between summarization for media content and the puzzle task: it is much more time-consuming for a single human to find the top-layer image in such a puzzle than figure out the plot of a movie or novel. With the movie or novel, a human will just need a single pass to figure out the plot. With the puzzle, the process is far more iterative and nonlinear.

Procedure

Each level of the puzzle asks workers to find patterns of elements. The user interface for the bottom layer is given by the "sequences interface" in Figure 7: workers indicate the word sequences they detect using checkboxes; in this case a worker detected a clothes-related sequence (pants, shirt, etc.) and a student-related sequence (books, school, etc.).

Each layer's checkbox puzzle encodes a grid of pixels. Every checkbox, when checked, turns on a corresponding pixel at the next higher level of the puzzle. So, patterns at each level produce pixel patterns that look like lines, curves, and diagonals at the next level up. These shapes combine to form the elements in that next level of the hierarchy.

The next step is for workers to vote on which worker's solution to promote upwards. Workers inspect the bitmask images produced by the checkboxes in adjacent grid sections. This interface is depicted in Figure 7, by the "voting interface": there are four adjacent 10×10 grid segments (upper left, upper right, lower left, lower right), each with four outcomes from different workers. Using radio buttons, a worker selects the bitmask image for each grid segment that is best — the interface shows how the four selected images fit together to create a symbol, a letter 'S' in Figure 7. (This voting task corresponds to selecting the best summary written by a worker for a specific part of a movie/novel.)

Combining the images from the voting task in Figure 7 produces the middle layer of the puzzle. For example, the 'S' letter in Figure 7, becomes an entry in the puzzle's middle layer; just like in Figure 6. We switch back to the sequences interface, and we ask workers to detect sequences of numbers and letters. The middle layer of Figure 7 depicts the sequences interface, where the numbers and letters are the outcome of the voting interface. Once again, workers identify patterns with the checkboxes to produce the lines and curves that form the top layer — the final solution to the puzzle, an image — and complete the upward phase.

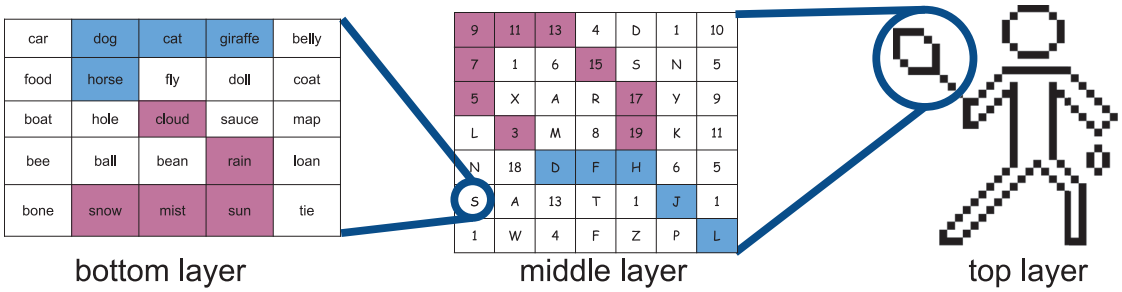


Figure 6: Hierarchical puzzle: In the bottom layer workers are asked to detect sequences of words. In the 5-by-5 segment shown here, one worker detects an animal-related sequence, while another worker detects a weather-related sequence. The two sequences form letter "S" in the middle layer. In the middle layer, one worker detects a sequence of odd numbers, while another worker detects a sequence of odd letters. The two sequecens form the racket of the tennis player in the top layer image.

Since some of the word sequences in the bottom layer are difficult to detect, and since some sequences split between tasks (atoms), the upward direction produces a noisy outcome with several pixels or entire line segments missing.

The compounding noise and difficulty from the middle layer make the top layer even more difficult. The imperfections become apparent when comparing the outcome after the upward phase in the top layer of Figure 7 to the true tennis player icon in Figure 6.

The downward phase percolates these noisy results back down to the lower levels and gives workers the opportunity to fix any mistakes. Workers rate the importance of each item (rating interface in Figure 7), indicating how important an element is for figuring out what the top-layer image shows. For example, a number/letter that is part of the sequence forming the racket or the ball is extremely important, while a letter that does not form any sequence is not important at all. Through the ratings, workers correct for missing sequences or misidentified sequences that may alter the meaning of the image, for example making the tennis player icon look like a badminton icon. This effect can be seen in the final outcome in Figure 7, where each "pixel" of the bitmap has an intensity proportional to its importance rating. In the top layer in Figure 7, the racket and the torso of the tennis player are largely missing, but after the downward phase they become much clearer, as the final outcome in Figure 7 shows.

Conclusion

Context trees combine many local views of a large input space to create a global view that can be consumed at several levels of detail. By having the big picture in mind, contributors can provide a meaningful answer about the importance of a scene in a movie, the importance of a paragraph in a novel, or detect hidden patterns in a multi-layer puzzle. Context trees suggest that it would be possible to crowdsource shorter versions of long articles, videos, or books, and let users filter information from the constantly expanding web of creative content. However, they are also a step toward solving the more general problem of crowdsourcing globally informed actions when no individual can see the entire decision space. Such a point of view opens opportunities in distributed cognition, crowd agents, and mobilizing a large group to act with one purpose.

A general limitation of context trees is that the atoms and their summaries must remain below each worker's time/cognition limit for a single task. If the branching factor b is high, a worker may get overwhelmed by looking at many atoms or child summaries and produce a low quality summary as a result. However, a larger b does allow each worker to see a larger percentage of the overall storyline, which gives them more context to create a good summary. Future systems might be able to analytically find the optimal b for a given task. In addition, context trees are limited to domains where it is possible for workers to create summaries that require less time to read or consume than the raw atoms.

There are many opportunities for more powerful hierarchical representations than the ones used here. For example, we envision that it might be possible to solve complex planning problems with context trees. Many participants could begin by designing local solutions, then percolating their ideas upward to test if the solutions are globally consistent (Zhang et al. 2012). The overall form of the proposed solutions could then filter back down to inform local decision-making. In this way, context trees could enable crowdsourcing to scale to extremely complex domains.

References

André, P.; Kittur, A.; and Dow, S. P. 2014. Crowd synthesis: extracting categories and clusters from complex data. In *Proc. CSCW*.

Bernstein, M. S.; Little, G.; Miller, R. C.; Hartmann, B.; Ackerman, M. S.; Crowell, D.; and Panovich, K. 2010. Soylent: a word processor with a crowd inside. In *Proc. UIST*.

Biswas, A., and Jacobs, D. W. 2012. Active image clustering: Seeking constraints from humans to complement algorithms. In *Proc. CVPR*.

Bragg, J.; Mausam; and Weld, D. S. 2013. Crowdsourcing multi-label classification for taxonomy creation. In *Proc. HCOMP*.

Chilton, L. B.; Little, G.; Edge, D.; Weld, D. S.; and Landay, J. A. 2013. Cascade: Crowdsourcing taxonomy creation. In *Proc. CHI*.

Gomes, R.; Welinder, P.; Krause, A.; and Perona, P. 2011. Crowdclustering. In *Proc. NIPS*.

Gupta, V., and Lehal, G. 2010. A survey of text summarization extractive techniques. *Journal of Emerging Technologies in Web Intelligence*.

Herlocker, J.; Konstan, J. A.; and Riedl, J. 2002. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Journal of Information Retrieval*.

Kittur, A.; Smus, B.; Khamkar, S.; and Kraut, R. E. 2011. Crowdforge: Crowdsourcing complex work. In *Proc. UIST*.

Kittur, A.; Chi, E. H.; and Suh, B. 2008. Crowdsourcing user studies with mechanical turk. In *Proc. CHI*.

Kulkarni, A.; Can, M.; and Hartmann, B. 2012. Collaboratively crowdsourcing workflows with turkomatic. In *Proc. CSCW*.

Lasecki, W.; Miller, C.; Sadilek, A.; Abumoussa, A.; Borrello, D.; Kushalnagar, R.; and Bigham, J. 2012a. Real-time captioning by groups of non-experts. In *Proc. UIST*.

Lasecki, W. S.; White, S.; Murray, K. I.; and Bigham, J. P. 2012b. Crowd memory: Learning in the collective. In *Proc. Collective Intelligence*.

Lasecki, W. S.; Wesley, R.; Nichols, J.; Kulkarni, A.; Allen, J. F.; and Bigham, J. P. 2013. Chorus: a crowd-powered conversational assistant. In *Proc. UIST*.

Norton, M. I.; Mochon, D.; and Ariely, D. 2012. The {IKEA} effect: When labor leads to love. *Journal of Consumer Psychology*.

Sheng, V. S.; Provost, F.; and Ipeirotis, P. G. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proc. KDD*.

Tamuz, O.; Liu, C.; Belongie, S.; Shamir, O.; and Kalai, A. 2011. Adaptively learning the crowd kernel. In *Proc. ICML*.

Wang, J.; Li, G.; Kraska, T.; Franklin, M. J.; and Feng, J. 2013. Leveraging transitive relations for crowdsourced joins. In *Proc. SIGMOD*.

Whang, S. E.; Lofgren, P.; and Garcia-Molina, H. 2013. Question selection for crowd entity resolution. In *Proc. VLDB*.

Zhang, H.; Law, E.; Miller, R.; Gajos, K.; Parkes, D.; and Horvitz, E. 2012. Human computation tasks with global constraints. In *Proc. CHI*.