AI Backed Crowd-Sourced How-to Manuals

by

Daniel J. Scarafoni

Submitted in Partial Fulfillment

of the

Requirements for the Degree

Master's of Science

Supervised by

Professor Philip Guo

Department of Computer Science
Arts, Science & Engineering
Edmund A. Hajim School of Engineering & Applied Sciences

University of Rochester
Rochester, New York

2014

Abstract

So called unskilled jobs such as deli clerks and table waiting in fact require a number of skills to do well; these skills are often learned on the job through experience and mentorship situations. The process could be simplified with instruction written by professionals. Canned Mentorship is a system which allows small groups of professionals to create comprehensive instructions for day-to-day tasks in their jobs for use by newer staff, reducing the need for supervision. This program uses a front-end user interface which allows users to view the instructions as they are being formed and vote on editions and revisions. A back-end server collects the natural language data and parses it to coordinate the groups ideas. The end result is a list of instructions describing a particular task.

Table of Contents

1 Introduction

Crowdsourcing has shown to be an effective means of solving tasks that are computationally difficult, or otherwise impractical, for artificial intelligence programs to solve. (**?**). There are many instances, of course,where humans are more effective than machines and vice-versa. To pick the most obvious example: humans are more efficient at qualitative, ambiguous tasks, while artificial intelligence (AI) agents tend to be better at more quantitative, direct tasks. Naturally, most tasks do not fall cleanly into one category or another; there are many tasks which cannot be done perfectly by human or AI agents. For this reason, the merging of human and AI is necessary for the completion of a number of tasks.

Consider, for example, the task of building an instruction manual for a common profession. If a restaurant wanted to create a manual for a number of everyday tasks in the restaurant in order to train new employees quickly. ADD SOMETHING ABOUT THIS BEING ABOUT SKILLED USERS

It would use data gathered from existing employees to solve this task, as the ambiguous and qualitative data of procedural tasks such making food and settling tables is far beyond the current capabilities of artificial intelligence. One experienced employee could be delegated to the task-instruction creation, but there are a number of issues with this method. First, an over-reliance on one individual leaves the system vulnerable to error. An individual may make mistakes when writing instructions. The only solution to this problem would be to have other users check the first users work, which automatically means that multiple users will be needed regardless. Second, many task, especially the qualitative, ambiguous ones previously mentioned, can be done in multiple ways. While the differences may not necessarily be erroneous in nature, they can pose problems. Certain delis, for example, have managers with different policies on the quality of meat used for sandwiches. If a trainee is taught the preference of one manager and not the other, issues can arise if the employee switches managers. Third, there are many instances where individuals have been shown to be slower than crowds (Lasecki and Bigham, 2013; **?**). It it likely that crowd-sourced users will also be more efficient than single users in this domain.

One major issue with gathering instructions from users is dealing with redundancy. Because many people have similar notions of instructions for various tasks, users tend to give similar instructions for a given step. This leads to redundancy. For example, if a group of 3 users were asked to write the first step for making a peanut-butter and jelly sandwich, the users might respond with

1. get two slices of bread
2. get two slices of bread, a knife, peanut butter, and some jelly
3. get the ingredients.

It is obvious in this situation that because suggestion 3 is described completely (and more thoroughly) by suggestion 2, that it is redundant and thus can be removed.

Failure to do so can impede the crowd-sourcing process. First, it introduces a form of noise into the system. With multiple identical sentences, if users are asked to vote on the suggestions, the voting process may be subverted. Consider the same situation as the last example, but the three inputs are

1. get two slices of bread
2. get the ingredients
3. get the ingredients.

If two users believe that option 2 (or equivalently, option 3) is correct, they could vote for either. If the third user votes for option 1, then there is no clear majority from voting, even though the consensus is clear.

This problem ultimately is an issue of natural language processing, and is unique in that it deals with very short documents in a very small corpus. A quick and efficient method of removing and/or minimizing redundancies in the context of this problem is essential to the completion of the task.

In the simply sense, the task it to take a series of unlabeled sentences and break them into groups depending on similarity. This naturally frames the problem as an unsupervised learning problem. More specifically, it is a problem that can be solved with a clustering algorithm. Further, there are multiple means of catagorizing sentences. Returning to the first list of sentences, it could be argued that setnence one, “get two slices of bread,” and sentence two, “get two slices of bread, a knife, peanut butter, and some jelly” should be merged because the former is a subset of actions of the latter. However, it could also be argued that the two should not be merged because they ultimately display two distinct thoughts. It is therefore good to find a clustering system that allows for grades or shades of similarity classiffication. Because this is one of the core features of agglomerative hierarchical clustering, it is hypothesized that this form of clustering will provide the best model for solving the problem.

In total, the task of building a set of instructions (or a “how-to” manual) efficiently from a group of users requires efficient crowd-sourcing techniques as well as sufficiently sophisticated AI to augment this process. Although there has been much research on using crowds to answer single questions, there has not been much work done in the next level of abstraction: having users answer lists of questions (i.e. instructions) (Lasecki *et al.*, 2013; Bigham *et al.*, 2010).

add objectives, purpose, etc

The remainder of this paper will be organized as follows:

- the Background section will discuss previous work done in both crowdsourcing and redundant sentence removal, as well the unresearched areas where this project will make its contributions

- The Methodology section will describe the Canned Mentorship system in detail, both at an algorithmic and implementation level. It will also descirbe the parameters tested and the experimental setups.
- The results section will describe the experimental results of the study, and discuss the any anomalies or other observations encountered during the testing process.

2 Background

2.1 Crowdsourcing

Crowdsourcing has shown itself to be a very effective computational resource. Formally, it is defined as

> the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call. (Brabham, 2008).

2.1.1 Current Scope and Usage of Crowdsourcing

Though crowdsourcing covers a very broad range of activities and fields, it is possible to define separate catagories. Brabham defined three separate categories:

1. crowdfunding
2. crowd labor
3. crowd research.

Crowdfunding is the use of crowd-based resources to fund projects. Websites such as gofundme.com, kickstarter, and patreon.com are all crowdfunding website which allow elicit funds from users. This method is particularly useful for freelance artists (Brabham, 2008).

Crowd labor utilizes the crowd for computation and other work-like tasks. Generally, this involves the users doing low effort tasks for low pay. In many situations, this can result in a much cheaper product than non crowdsourced, professional settings. Crowdsourcing is also often used for tasks which are prohibitively difficult or impossible with current computational means.

For example, the stock photo website iStockphoto collects images uploaded by users, and pays them a small compensation for their effort. This website is able to lease the rights to these images for \$1, which undercuts professional photographers' prices by over 90%, Crowd labor also extends to a number of other fields, including industrial research (Howe, 2006).

2.1.2 Hierarchical Classification in Natural Language Processing

The idea of using Hierarchical clustering in order to separate sentences has been mentioned in the literature previously. Skabar et al. illustrates recent successes in using this method of unsupervised learning to classify text. !!moar here!!

Cimiano et al. worked with classifying terms into hierarchies using various clustering methods. They were able to conclude that despite variance in classification efficacy between humans and machines, hierarchical clustering was found to perform as well as human users. In fact, they noted that a large portion of the difference in efficacy was a result of the algorithm chosen for classification. Further, the study shows that hierarchical agglomerative clustering was one of the top performing algorithms, and one of the more efficient (their algorithm operated in $O(n^2)$ time. This study, however, only compared words, not complete sentences. (Cimiano *et al.*, 2004).

2.1.3 Similarity of Short Strings

There are a number of algorithms for comparing similarity between sentences. A survery by Achananuparp et al. studied a wide variety of techniquest. In particular, the study compared the efficacy of lexical metrics, which measured the syntactic and word content similarity of sentences, and linguistic metrics, which compares semantics similarities between two sentences (Achananuparp *et al.*, 2008).

Their method for semantic difference utilized wordnet, and relied on calculating the distance to the most recent common ancestor in between two words' term hierarchies (Achananuparp *et al.*, 2008; Abdalgader and Skabar, 2011). In this study, the most effective methods utilized a diverse combination of both lexical and linguistic methods for comparing sentences (Achananuparp *et al.*, 2008). !!embellsih this more if we can!!

2.1.4 Clustering and Crowdsourcing

Though there are many tasks which have required filtering of small crowd input to solve tasks and answer questions, there has not been widespread use of clustering in projects that attempt to answer user questions or create and modify instructions. The Toolscape project, for example, read in user input to annotate how-to videos. Clustering algorithms were used in simple situations: for example, to cluster time selections on the video in question.

When comparing string annotation inputs from the users, however, as simple string comparison method was implemented (Kim, 2013).

The Chorus Project, which allowed the crowd to answer questions from the user, used no back-end AI at all, and simply had users vote on the best solution. Though the end product had a high answer accuracy rating, the latency of responses was on the order of minutes in most cases. This system also had a complex, hierarchical incentive system and a specialized interface to keep users engaged, and their answers accurate (Lasecki *et al.*, 2013). The usage of machine learning tecniques in the backend was not explored, and remains a field of investigation open to future study.

Scribe built on this, and used statistical natural language processing in order to merge text input from crowds. This project, which was designed to allow real-time captioning of crowd input on audio, is able to merge sentence input using a backend AI agent which utilizes the multiple sequence alignment algorithm to build a substitution matrix to identify potential typos in typing. As text is entered by the users, the algorithm creates a graph which traverses the text tokens between concurrent input strings to form the best reconstruction of the sentence. This study isunique in that it utilizes an AI backend to supplement a crowdsourcing systems (Naim *et al.*, 2013).

More prolific than AI backed crowdsourcing, however, is crowd-supplemented AI. Crowds have been shown to be a valuable asset for labeling data for machine learning, a task which is difficult to automate(Pedro and Hruschka Jr, 2012). !!put more stuff here!!

2.2 Methodology and Algorithms

This project used data gathered from a crowdsourced workflow generatign project, *Canned Mentorship*. This data was run through a number of clustering algorithms, and the results were evaluated by human users.

3 Methodology

The efficacy of the program was dependent on its ability to coordinate users and ensure that they were able to work together to generate the final instructions. Thus, it was necessary to not only design an effective workflow, but to also make sure that the program implementation was easy to understand, robust, and close to the theoretical workflow.

3.1 Workflow

Turkomatic ran into difficulties in organizing the workflow. Their system was unable to coordinate workflows, in no small part due to the complexity of the algorithm. Their algorithm used recursive workflow partitioning, and allowed for complex sub-task creation. This led to significant difference of opinion between users, and little consensus was obtained without the use of a single task evaluator who would oversee the task (also known as "derailment" (Kulkarni *et al.*, 2012).

To combat this problem, Canned Mentorship utilizes an iterative, one directional workflow. Instead of a complex, largely unordered workflow of turkomatic, Canned Mentorship makes users create the instruction list in the order it will be executed. This was done to avoid problems with granularity and starvation that were problematic in this system.

3.1.1 Instruction Generation

CannedMentorship uses a very simple task generating algorithm, with no explicit task subdivision or branching. The system simply asks users to create steps in sequence, from beginning to end, in order. Each step is completed in order, with the latter steps coming directly after former steps chronologically. This avoids the issue of task subdivision. Users do not have to deal with the complex uncertainties of task completion, and do not have to worry about planning the task.

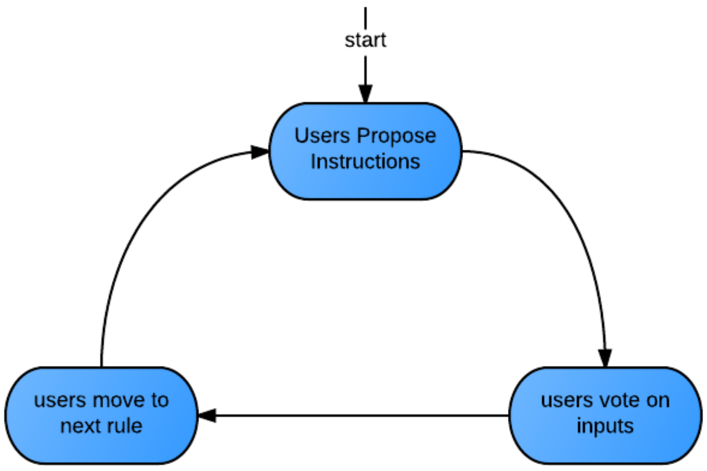
Previously, three-stage find, fix, verify algorithms have been used successfully in crowd-sourcing (Kim, 2013; Bernstein *et al.*, 2010). Canned Mentorship builds off of this system.

In order to create an instruction step, users follow a cyclical process. First, users suggest a step. Second, they vote on the available propositions. Finally, they expand onto the next steo. This process for adding a new rule is illustrated with figure 3.1. Please note that this figure does not illustrate the overall workflow of the application, only the process for adding a new rule that users are involved in.

The "find" stage is already decided by the system. The users always work on the next instruction, which is the "found" area. At this point, the users are asked what the step in progress should be. The users type their proposition for the next instruction, which is a string of text one to two sentences in length. The inputs are collected and compiled by the server.

At this point, the voting stage begins. This point, which is analogous to the "verify" stage noted previously, is when users select the best proposition from the group. The instruction with the majority vote becomes the finalized instruction.

In the final step, users expand onto the next instruction. This step, appropriately, is similar to the "expand" stage employed in projects such as Soylent. At this stage, the instructions loop back to the beginning, and users expand the instructions by adding the next instruction that follows the newly added one.


start
Users Propose Instructions
users vote on inputs
users move to next rule

3.1.2 Additional user control

In addition to the above, users could were also given other control flow methods. After an instruction was finished, and before another was started, users would be able to vote to finish the instruction set. If the majority voted that the current instruction set was a sufficient description for the prompt, then the task finished.

Also, Canned Mentorship implemented a "pseudo leader" position. This position designated one individual to be the "pseudo-leader" and was given slightly more control over the rule formation than other users. At the end of each step input from the pseudo leader was required in order to advance to the next step. This was done to add a "requester" figure to the setup. Previous studies have shown that although users can have significant coordination and agreement difficulties when designing workflows, the presence of a leader type user with more authority over the workers can significantly improve results (Kulkarni *et al.*, 2012).

3.2 Implementation and Webapp details

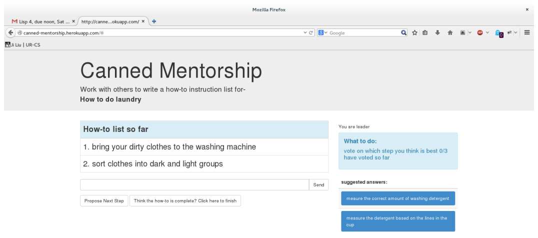
Our project began by creating a small webapp which coordinated the instruction generating process. This program, also dubbed "Canned Mentorship," consisted of two separate programs:

1. a website interface for coordinating working users
2. a back-end AI script for classifying answers and removing redundant inputs

3.3 System Implementation

3.3.1 The Webapp

The webapp was a program written using Python Flask and was hosted on Heroku webapps. It was designed as a lightweight, simple system which hosted the users for the duration of the task and also coordinated their actions.


Mozilla Firefox

Lisp 4, due noon, Sat ... | http://canne...okuapp.com/

canned-mentorship.herokuapp.com/#

Ji Liu | UR-CS

Canned Mentorship

Work with others to write a how-to instruction list for-

How to do laundry

How-to list so far

1. bring your dirty clothes to the washing machine
2. sort clothes into dark and light groups

You are leader

What to do:

vote on which step you think is best 0/3 have voted so far

suggested answers:

meaure the correct amount of washing detergent

measure the detergent based on the lines in the cup

Figure 3.2: Rejection sampling was also considered as a placement mechanism.

As noted earlier, Heroku was chosen as the host location for the webapp. Other hosting systems were considered early on in development. meteor

digitalocean

that thing phillip liked

google apps

finish this

3.3.2 Algorithms Tested

Our study studied a total of fifteen distinct algorithms, including three clustering algorithms: agglomerative hierarchical clustering, DBSCAN, and affinity propagation.

Hierarchical clustering was the primary clustering algorithm in question for this study, for the reasons outlined in section one. DBSCAN and affinity propagation were included. DBSCAN clusters based on density, and affinity propagation sends signals between potential clusters, adding points to clusters based on how well they represent different groups. These two were chosen in part to give context for the performance of agglomerative clustering, but also to test their efficacy with the chosen feature selection and distance algorithms.

For evaluation, it makes sense to recount that the filtered suggestions (the input minus the redundant ones) will ultimately be evaluated by humans. The metric for performance of the algorithms, therefore, should present the user with the original list of propositions, the filtered list, and then be asked to rate how well the filtered list represents the original on a varying scale (perhaps from 1 to 5, with five being "perfectly" and 1 being "poorly").

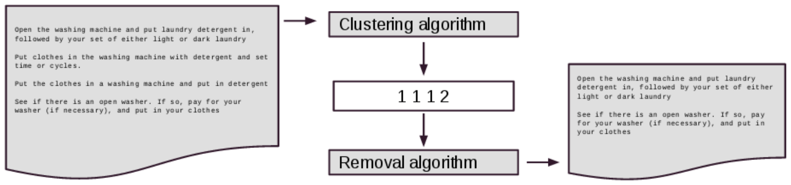
3.3.3 Algorithm Details

The given input to all the algorithms was a list of instructions, one came from each user in the participant. Each input was approximately one sentence long, and described a step in the performance of a task. Preprocessing involved converting all of the letters in the sentence into lower case, removing stopwords (such as "a" "an" and "the") and punctuation, and stemming every word (converting words to their base form, such as "runs" $\rightarrow$ "run." Once this preprocessing finished, the inputs were utilized by one of the respective algorithms. Although the next steps were ultimately dependent on which algorithm was used, the end result was a clustering of the sentences into groups based on similarity.

One of the algorithms tested was the "Bag of Words" algorithm for feature selection. This was the simpleset algorithm tested, and used tdf-if extraction to create a feature matrix. Once this was done, the resulting matrix was input into one of the clustering algorithms, using standard Euclidean distance (l_2 norm) to calculate the distance (or similarity) between points.

An N-grams/bag of words hybrid was also implemented, which took the existing bag of words feature matrix above, and also included the ngrams features of the sentences. This had the additional benifit of preserving some of the sentence structure in the feature matrix.

Several semantic distance algorithms were also used. The wordnet library has been used in the past to calculate semantic distance between terms of the same part of speech (Achananuparp *et al.*, 2008; Abdalgader and Skabar, 2011). We implemented the algorithm used by Achananupart et al. which calculated the similarity of each word with the most similar word in the opposing sentence. This algorithm taskes advantage of word hierarchical taxonomy in wordnet. Words in wordnet are related by hypernym/hyponym relationships. This allows semantic distance to be calculated as a function of the distance



Open the washing machine and put laundry detergent in, followed by your set of either light or dark laundry

Put clothes in the washing machine with detergent and set time or cycles.

Put the clothes in a washing machine and put in detergent

See if there is an open washer. If so, pay for your washer (if necessary), and put in your clothes

Open the washing machine and put laundry detergent in, followed by your set of either light or dark laundry

See if there is an open washer. If so, pay for your washer (if necessary), and put in your clothes

Figure 3.3: Data was gathered from the webapp.

of the most recent common ancestor between two words (Abdalgader and Skabar, 2011).

Similarly, MIT's Conceptnet program allows for much quicker semantic comparison. Conceptnet utilizes a hypergraph which links terms in a language based on a wide array of characteristics. In this program, the similarity between two words can be computed as a function of the graph difference between the two input terms. When testing these two methods, a feature matrix was not utilized. Instead, a distance matrix of distances between sentences was constructed using the above heuristics, and this was linked with the relevant clustering algorithm.

I also developed a novel algorithm for this task. This algorithm, which was inspired by Achananuparp et al's findings that combinatory sentence similarity methods are more accurate, begins by converting the input sentences into a bag of words/ngrams feature matrix. Once this is done, the matrix is converted into a distance matrix, created by taking the average of the standard euclidean distance between feature vectors and the semantic distance calculated through wordnet. This result is then linked with the relevant clustering algorithm.

Each of the previously mentioned feature-extraction/distance metric formulae were tried with each of the three clustering algorithms, resulting in fifteen distince algorithms. This clustered the sentences into distinct groups. For each group, all but the longest sentence were removed. This was because this tecnique was shown to be effective in previous studies (**?**), and because when two sentences were grouped together, the shorter sentence tended to contain less information and precision while still trying to express the same thought. !!this sounds wonky!!

The new sentences were then collected and evaluated by participants, randomly sampled from the University of Rochester's student population. These students were given the algorithms' results, and told to rate how well they removed redundant thoughts on a scale from one to five (with one being the lowest and five being the best). The entire process can be seen in 3.3.

4 Results

The primary variables tested in this experiment were the efficacy of the merging algorithms at removing redundancy and the efficency of these algorithms.

4.0.4 Removal Efficacy

The algorithm combinations' results were varied. The top performing algorithm was the custom made algorithm combined with hierarchical clustering, which outperformed all others by a significant margin.

An f-test of the results indicated that there was significant difference between the answer received ($F \leq 0.05$) !!does this work?!!. A secondary T-test between the highest custom algorithm and the second-highest yielded a significant difference as well ($p < 0.05$). This shows strong evidence that the custom algorithm performed the best of all the algorithms.

The results also showed, interestingly, that the bag of words algorithm performed very well when utilized with the hierarchical aggomerative clustering, performed second best. This seems to indicate that although simple bag-of-words was not an effective form of sentence comparison, in the right situations, it can be utilized as an effective means of classifying sentences (at the very least, for human evaluation). Figure 4.1 illustrates the results. Note that in this chart, the custom algorithm is labeled as "diverse" in reference to its usage of components from a diverse assortment of algorithms.

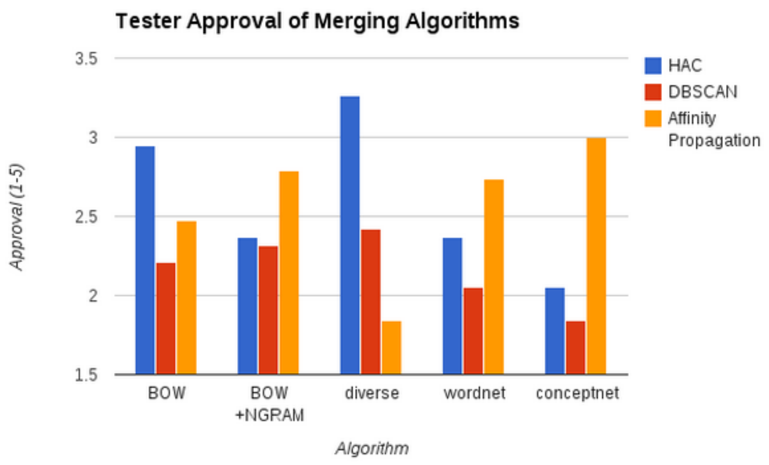


Figure 4.1: The user approval of various methods.

4.0.5 Runtime analysis

The runtime of each algorithm was also evaluated. the bag of words and n-grams only models performed very well, and were able to finish in significantly less time than the other algorithms.

The custom diverse algorithm performed slower than the other algorithms tested, though this result may not necessarily scale with the size of the matrix. The most efficient algorithms for agglomerative clustering run in $O(n^2)$ time, and this is not extensively impacted by the additional computation to compute wordnet similarity. At most it can be increased to $O(n^2T)$, where T is the distance to the nearest common ancestory in the wordnet hierarchy between two terms. !!!is n3 time still efficient?!!

Further, affinity propagation, which operates slower than hierarchical clustering, was shown to actually be quicker. Most likely, the data used in this experiment was not large enough to accurately gage the runtime of these algorithms.

One efficiency that was sufficiently measured, however, was that of the conceptnet algorithm. The full conceptnet was too large to be deployed onto a small webapp server (it is approximately 5gb in size). Thus, for this experiment, the rest api was utilized. The latency from the http requests necessary to compute the distance between

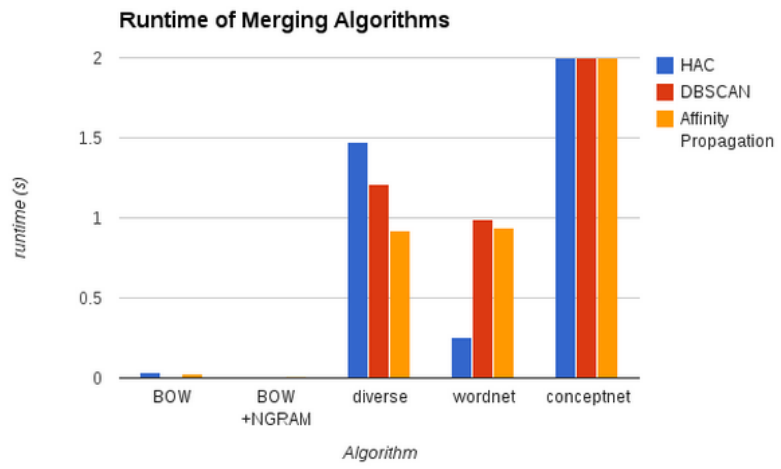


Figure 4.2: The runtime of various algorithms. Note that the conceptnet time was so large that the bar had to be truncated for this graph.

two terms, combined with the server side computation of this task, led the conceptnet runtime to be much larger than any of the other methods. Figure 4.1 illustrates the results.

5 Conclusion and Discussion

This project provides much needed data on the efficacy of machine learning algorithms to sort and cluster crowd input. Our findings indicate that the most effective algorithm for doing so is the diverse algorithm coupled with hierarchical agglomerative clustering. This finding is consistend with findings in previous studies (Achananuparp *et al.*, 2008). The project also shows a great disparity between feature extraction and distance finding within clustering algorithms. For example, the diverse algorithm performed much, much worse dependng on which clustering algorithm was used. The same is also true for the conceptnet algorithm. What factors cause this is not presently known, and further research is needed to try and identify underlying causes.

Further, the runtime analysis of this project was inconclusive. The runtime results did not correlate with the theoretical runtime of most algorithms, which indicates that the testing data was not of sufficiently large to test this feature. That being said, the custom algorithm does place much additional tax on the runtime of the hierarchical agglomerative clustering algorithm; this indicates that this particular combination can be scaled up to big data sized problems. However, further testing is necessary to verify this the theory.

Bibliography

Khaled Abdalgader and Andrew Skabar. Short-text similarity measurement using word sense disambiguation and synonym expansion. In *AI 2010: Advances in Artificial Intelligence*, pages 435–444. Springer, 2011.

Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. The evaluation of sentence similarity measures. In *Data Warehousing and Knowledge Discovery*, pages 305–316. Springer, 2008.

Michael S Bernstein, Greg Little, Robert C Miller, Björn Hartmann, Mark S Ackerman, David R Karger, David Crowell, and Katrina Panovich. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 313–322. ACM, 2010.

Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, et al. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 333–342. ACM, 2010.

Daren C Brabham. Crowdsourcing as a model for problem solving an introduction and cases. *Convergence: the international journal of research into new media technologies*, 14(1):75–90, 2008.

Philipp Cimiano, Andreas Hotho, and Steffen Staab. Comparing conceptual, divise and agglomerative clustering for learning taxonomies from text. In *Proceedings of the 16th*

Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, 2004.

Jeff Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

Juho Kim. Toolscape: enhancing the learning experience of how-to videos. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 2707–2712. ACM, 2013.

Anand Kulkarni, Matthew Can, and Björn Hartmann. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1003–1012. ACM, 2012.

Walter S Lasecki and Jeffrey P Bigham. Interactive crowds: real-time crowdsourcing and crowd agents. In *Handbook of Human Computation*, pages 509–521. Springer, 2013.

Walter S Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F Allen, and Jeffrey P Bigham. Chorus: a crowd-powered conversational assistant. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 151–162. ACM, 2013.

Iftekhar Naim, Daniel Gildea, Walter S Lasecki, and Jeffrey P Bigham. Text alignment for real-time crowd captioning. In *HLT-NAACL*, pages 201–210, 2013.

Saulo DS Pedro and Estevam R Hruschka Jr. Collective intelligence as a source for machine learning self-supervision. In *Proceedings of the 4th International Workshop on Web Intelligence & Communities*, page 5. ACM, 2012.