A Genetic Algorithm for Finding a Small and Diverse Set of Recent News Stories on a Given Subject: How We Generate AAAI's AI-Alert

Joshua Eckroth, Eric Schoen
i2k Connect, LLC

Abstract

This paper describes the genetic algorithm used to select news stories about artificial intelligence for AAAI's weekly *AI-Alert*, emailed to nearly 11,000 subscribers. Each week, about 1,500 news stories covering various aspects of artificial intelligence and machine learning are discovered by i2k Connect's *NewsFinder* agent. Our challenge is to select just 10 stories from this collection that represent the important news about AI. Since stories and topics do not necessarily repeat in later weeks, we cannot use click tracking and supervised learning to predict which stories or topics are most preferred by readers. Instead, we must build a representative selection of stories *a priori*, using information about each story's topics, content, publisher, date of publication, and other features. This paper describes a genetic algorithm that achieves this task. We demonstrate its effectiveness by comparing several engagement metrics from six months of "A/B testing" experiments that compare random story selection vs. a simple scoring algorithm vs. our new genetic algorithm.

Since 2001, the AITopics.org information portal has hosted links to news, overviews, journal and conference papers, and classic texts about artificial intelligence (Buchanan and Glick 2002; Buchanan, Eckroth, and Smith 2013). Every item is classified according to a deep hierarchy of AI-related topics as well as industry topics such as transportation, health and medicine, and education. Initially, all items on the site were collected and classified manually. In 2010, we introduced automatic discovery and classification of news stories with *NewsFinder* (Eckroth et al. 2012).

AAAI had originally hired a webmaster who spent approximately ten hours a week to find and generate the alerts manually (as part of other duties covered by an NSF grant). We estimate it took about 3-6 minutes to search for and review each candidate story, or about 10-20 candidates per hour. At a nominal $35 per hour the cost of the alert service was thus about $18,000 per year to scan and select from a few hundred candidate stories per week.

When the webmaster left, AAAI continued the manual service with volunteer help. This task requires a person with some knowledge of AI and editorial skill, so it was difficult to find a replacement, and relying on volunteers is not sustainable. In January 2017, we automated the weekly email alert containing the week's most important and interesting news stories about artificial intelligence and machine learning. Known as *AI-Alert*, this service is an official publication of AAAI and received without charge by virtually all its members and affiliates and many members of the interested public, nearly 11,000 people in all. Each alert contains just 10 stories from the 1,500 or so found by *NewsFinder* each week from nearly one hundred sources including AAAI journals and conferences, arXiv.org, links on Twitter marked with the #artificialintelligence hashtag, and many reputable news sources.

In the first year of our automated *AI-Alert*, stories were selected for the alert according to a scoring algorithm we call "TopClass." In short, this algorithm grouped the week's stories according to their topic classifications, then picked one story from each of these groups. We eventually felt this algorithm was too simplistic as we sometimes saw too many stories from the same day of week or same news source. We also saw that the algorithm sometimes failed to pick a good news source for a specific event that was reported by several sources in the week. Thus, we realized that story selection was a multicriteria decision problem and a technique like genetic algorithms may be able to do a better job. We developed a new genetic algorithm for story selection and conducted a six-month experiment in which readers received either the original TopClass selections, the new genetic algorithm selections, or random stories in their alert. After concluding the experiment, we are able to report in this paper that the genetic algorithm indeed performed significantly better. Since the conclusion of our experiment in July 2018, *AI-Alert* has used the genetic algorithm for all weekly alerts.

To give a sense of reader engagement since July 2018, we can report the percent of readers who open the email, the percent who unsubscribe, and the percent of readers who click at least one story (each story is displayed with its title and a link to the original source). *AI-Alert* consistently beats Computer and Electronics industry marketing email averages (MailChimp 2018) for open rate (our average 30% vs. their 19%), unsubscribe rate (our 0.11% vs. their 0.29%), and click rate (our 5.5% vs. their 2.0%). Except for a final editorial review, in which we sometimes remove stories that we believe are inappropriate and stories that were misidentified as newly-published, the production of each week's alert is entirely automated.

In this paper, we describe how we automated the selection process of *AI-Alert* while maintaining high quality. Our challenge in developing this automation can be summarized as follows.

- Each week, about 1,500 stories must be filtered down to the 10 best candidates.
- Each story should be closely, not just tangentially, related to AI.
- No single AI topic should dominate the alert; i.e., diversity should be preferred even if much of the news media wishes to focus on a single event for that week.
- Stories should span the entire week and not just a single particularly active day.
- Duplicate and overlapping stories should be removed so no single event dominates the alert.
- No single publisher should dominate the alert, though high-quality publishers should be preferred.
- Since the topics and news events change week-to-week, there is almost no opportunity for supervised learning, so the stories must be selected from *a priori* features rather than reader feedback.

As noted above, we use a genetic algorithm to solve this multicriteria optimization problem for selecting stories for each week's alert. Experiments with this algorithm are discussed later in this paper.

The challenge of selecting stories for an alert can be seen in the following example from March, 2018. During one of Uber's self-driving car tests in Arizona, a pedestrian was tragically killed on March 18, 2018. *NewsFinder* acquired its first story about the event on March 19 by monitoring the #artificialintelligence hashtag on Twitter. This story was published by The New York Times. An alert was already scheduled to be generated on March 20, and by this point in time *NewsFinder* had acquired 101 stories about the event. The alert was to cover stories from the afternoon of March 13 to the morning of March 20, and 1,581 stories fit this criteria. Of those stories throughout the week, 161 of them were about autonomous vehicles (which includes the stories about the Uber crash). Due to several diversity criteria, the genetic algorithm ultimately selected two stories about the Uber crash: the aforementioned New York Times story and a follow up story (also published on March 19) from CNET, which stated that Uber had quickly suspended their self-driving car operations. After sending the alert, we found that 34% of clicks were directed to these two stories. This example shows that the alert is neither overwhelmed by a flurry of stories about the same event in a short time period, nor does it miss some of the important events that readers will want to see.

The rest of this paper is organized as follows. First, we summarize statistics about recent reader engagement with the *AI-Alert*. Then we give an overview of the design and deployment of our alert generator. Next, we address related work on the subjects of search result diversification and news aggregation. We then proceed to describe our genetic algorithm, and follow this with results from a six-month 'A/B' test. Finally, we discuss the results and explain how the alert is maintained on an on-going basis.

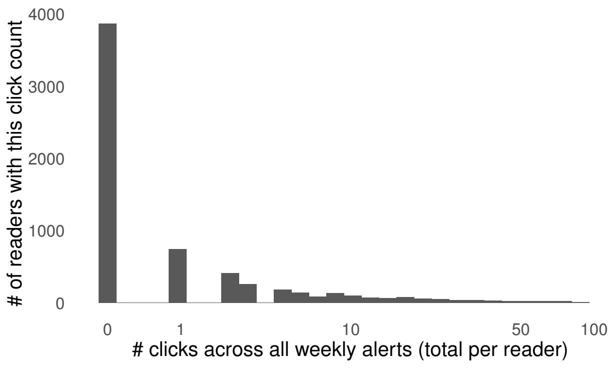

Figure 1: Frequency plot of the number of readers who have clicked links in the alerts. The x-axis shows the number of clicks and the y-axis shows the number of readers with that click count. These clicks are totaled across all 25 weekly alerts. Note that the x-axis is logarithmic.

Engagement with *AI-Alert*

Before we explain our experiments in building a better *AI-Alert* with a genetic algorithm, it is important to understand how readers typically engage with the alert. The metrics we report here summarize the readers of 25 weekly alerts that were each generated using our genetic algorithm during our six-month experiment, January 16, 2018 to July 3, 2018. Later, when we discuss experimental results, we will show how some metrics differ depending on the selection algorithm.

While an average of 30% of recipients open any given week's alert, most (about 60%) have never clicked a link in any alert, though we have anecdotal evidence that these subscribers derive value from reviewing the titles alone. For those who have clicked a link at any time, there is a long tail distribution where a very small number of readers have clicked many links. This distribution is visualized in Figure 1.

Each alert contains 10 news stories. During our experiment, at least one story in every alert was clicked by at least one reader. The mean number of stories in an alert clicked by any reader was 9.56 and the median number was 10. Thus, in nearly every alert, 9 or 10 out of 10 stories were clicked by some reader.

Most readers who do click a story just click one. Again, we have a long tail distribution, as seen in Figure 2. Among stories that were clicked, most were clicked about 13 times (mean), though rarely a story was clicked many more times. Figure 3 shows this distribution.

In summary, the active readership of *AI-Alert* is relatively small. About one-third open the email, and about 40% have ever clicked one or more stories. Thus, about 12% of all recipients actively engage with the alerts in any one week. It is not clear to us if these numbers are typical for weekly

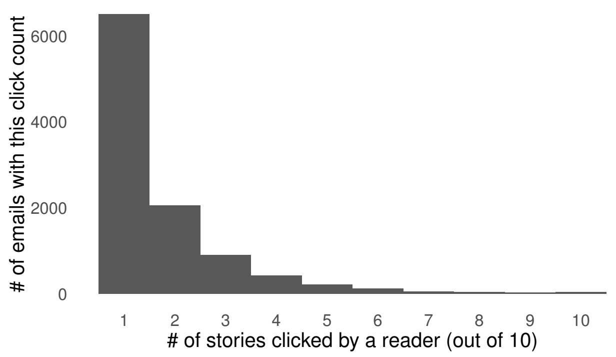

Figure 2: Frequency plot of the number of stories (out of 10) in each alert that a reader clicks. It is clear that most readers who click a story just click one story.

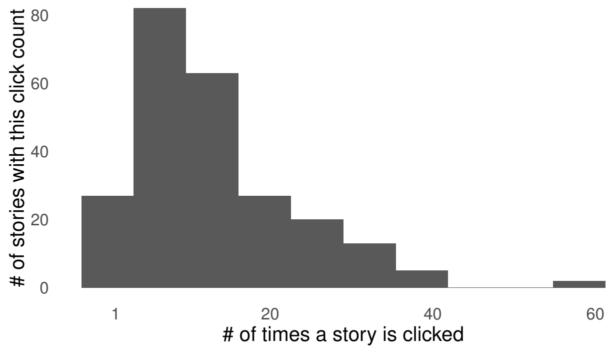

Figure 3: Frequency plot of the number of clicks a story receives. Most stories receive between 5-20 clicks while few stories receive many more clicks.

emails containing links to news stories about a given subject because, to our knowledge, such detailed statistics have never been published.

These insights about the readership of *AI-Alert* contextualize our experiment in generating a more valuable set of 10 stories each week.

System Design and Deployment

NewsFinder is composed of several components from i2k Connect's suite of technologies. These components include web crawlers and Twitter agents, a document enrichment service, an alert generator that finds good stories to include in the alert, and an email generator that uses a template to format the email to a consistent style. The document enrichment component uses proprietary topic classification technology to identify the various topics that a news story covers. We primarily use two deep hierarchies of technologies, including various kinds of artificial intelligence and machine learning, such as deep learning, reinforcement learning, rule-based expert systems, constraint solvers, etc. And we use an industry hierarchy to identify whether a story is discussing drones, robots in healthcare, fraud detection, marketing, etc. Once news stories are found and enriched

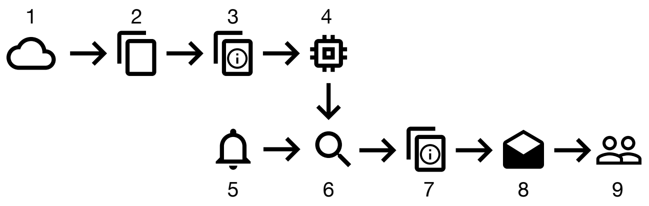

Figure 4: System design, showing two separate processes: acquisition and enrichment of news stories (top); and generating and emailing alerts based on a timer (bottom). The steps shown are as follows. (1) *NewsFinder* crawls the web to find news stories and extracts the main body text (2) of each story. This text is then analyzed to enrich it with metadata such as which types of AI technologies and applications are mentioned in the text, if any, resulting in an enriched text (3). This enriched text is then saved to a search database of candidate stories (4). In the second process, a weekly timer (5) wakes up the alert generator, which first searches the database for relevant stories (6), filters these stories to a select ten (7), generates an email based on a formatting template (8), and sends this email to subscribers (9).

with classifications and other metadata, they are saved into a search index for later retrieval. Then, once a week, the alert generator wakes up and retrieves all stories in the search index that have a classification somewhere from the AI subhierarchy in the technology hierarchy. Stories are further filtered through several quality-control steps, explained below, leaving about 200 candidates each week. These are then filtered down to the top 10 using the genetic algorithm we describe below. An overview of these steps is shown in Figure 4.

NewsFinder, and virtually all of i2k Connect's technologies, is implemented in the Clojure language, a Lisp variant that produces Java bytecode and therefore can run anywhere the Java Virtual Machine is available. Since web pages can be tricky to parse to find the main body text, we use the "snacktory" library by Peter Karussell[1] and several site-specific rules to handle edge cases with complex formatting. We use the "darwin" library by Jony Hudson[2] for the genetic algorithm. Finally, we use the "enlive" library by Christophe Grand[3] to generate the alert based on an HTML template.

The alert generator runs as an HTTP service that is activated by POST request submitted by a weekly *cron* job. The alert generator queries the search index (running Solr) to find stories and ultimately produces an HTML email. This email is saved to a file rather than emailed directly to subscribers because we wish to review the alert before it is sent. If any stories in the alert are inappropriate (the story is offensive or the website hosting the original story has deleted or moved the story), or the story's publication date was not correctly identified by our system, then we modify the POST request to "ignore" those stories, and regenerate the alert. Usually,

[1]https://github.com/karussell/snacktory

[2]https://github.com/JonyEpsilon/darwin

[3]https://github.com/cgrand/enlive

regeneration is not required, but when it is needed, we usually only need to regenerate the alert once or twice at most. The resulting HTML file is then uploaded to SendGrid, our mailing list provider, and sent to subscribers at a predefined time (11:00 AM Eastern US time).

It is worth noting that the alert generator can create alerts of different kinds. Using the same genetic algorithm, a daily email alert may be generated for stories about any topic that is covered by one of our hierarchies. We also use the alert generator to automatically submit a daily link to our Twitter account.[4] Since we only post one tweet per day, the genetic algorithm is not used here; instead, a random story from a high-quality news source is selected.

The alert generator was developed and refined by several people over a period of two years (January 2016 to January 2018), though not continuously during this period. The core functionality for finding candidate news stories, filtering them with the TopClass algorithm, and then formatting the alert with an HTML template required a few weeks of a single software engineer's time. Once the system was operational, adding the genetic algorithm required about a week of effort. The main challenge was inventing a "fitness function," explained in more detail below.

Related Work

To our knowledge, there is no prior published work on automating the generation of diverse and timely news digests on a given subject and delivered via email to a wide audience. The apparent uniqueness of our task is due to the combination of several factors: (1) we emphasize finding diverse news stories about a given subject (artificial intelligence in our case) from high-quality sources; (2) we have no profile information about the readers of *AI-Alert*, as users can subscribe with just an email address and no prior interaction with AITopics; (3) the alert is delivered via email so we are interested in reader engagement metrics like click rate rather than precision/recall metrics from a groundtruth document corpus. Prior work addresses each of these factors but not the combination of all three.

Regarding the challenge of finding diverse news stories about a given subject, we find related work in diversification of search results. Diverse search results can increase the likelihood that users find the documents they are seeking. Agrawal, et al. (2009) prove that the problem of optimally diversifying search results for a given search query is NP-hard. They proceed to develop an efficient greedy algorithm for diversifying results and demonstrate its effectiveness from a study using human judges. Dou, et al. (2011) focus on diversifying results according to multiple dimensions. They develop a greedy algorithm to examine a query's various meanings based on uses in web pages, among other sources. They demonstrate that their system has a better ability to find relevant documents by examining diverse meanings for the query. Cecchini, et al. (2018) also look at search diversification and use a genetic algorithm to evolve a population of queries to improve coverage and relevance of search results.

[4]https://twitter.com/ai_topics

Personalized news has been addressed by multiple authors. For example, Gabrilovich, et al. (2004) developed *Newsjunkie* to deliver stories from live news streams while avoiding stories that report on the same breaking event or report about events that the user has already reviewed. More recently, Li, et al. (2011) developed a news recommendation system that personalizes the stories per user interest, while maintaining novelty and diversity. We note that *AI-Alert* is generated for a general audience rather than for each user specifically according to their interests, though research in generating representative news appears to focus on personalization schemes.

Mishra and Berberich (2016) focused on generating a digest of a news event by selecting excerpts from documents and using multicriteria optimization to maximize diversity in the digest's perspective on the events. They use integer linear programming to solve this optimization problem. Their approach is interesting because, presumably, we also could use integer linear programming instead of a genetic algorithm to solve our multicriteria optimization problem. However, their approach is not directly applicable because we are not interested in generating summaries of the news, but rather linking readers back to the original source.

Experimental Methodology

Our experiment focuses on finding a better selection algorithm for including stories in the weekly alert. We will determine success by measuring the click rate, i.e., the ratio of the count of users who clicked a story to the number of alert recipients, expressed as a percentage. We expect click rate to improve with a better selection algorithm. In other words, once a reader has opened the email, we expect that reader will find more relevant and interesting stories and thereby be more likely to click them.

Our experimental methodology is known as "A/B testing," where two different alerts are tested during the same week on two distinct subsets of subscribers. In fact, we tested three different selection algorithms (random, TopClass, genetic), giving us a kind of "A/B/C" test. For each of the 25 alerts generated and sent during the six-month experiment, three distinct subsets, each containing a *random* 20% of subscribers, were assigned to each of the three algorithms. Readers did not know, and indeed would have no way of knowing, which algorithm generated the alert they received. The email subject lines were identical in each case, so we did not expect open rates to differ for the A/B/C variations of the alert. The test lasted four hours each week. After this time period, whichever of the three emails received the highest click rate was then sent to the remaining 40% of the readership. We did not measure engagement after the four hour window closed each week.

Because each reader had a random chance of being in any one of four groups (experimental group A, B, C, or the group that received the best performing version after the four hour time window), readers likely received emails generated by several different versions of the selection algorithm over time. Assuming at least one algorithm is poor and one is great, readers saw a mix of quality in the alerts in this experimental period. Thus, it is unlikely that any reader's interest

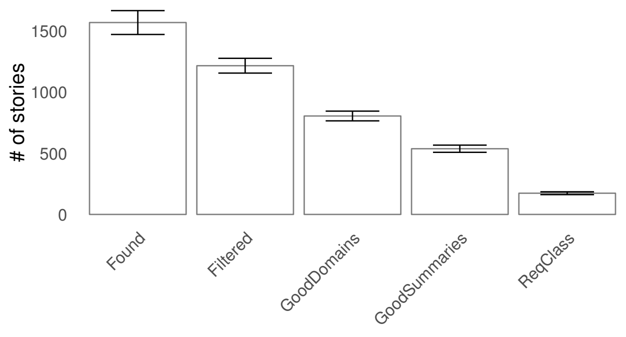

Figure 5: Average counts of stories after each filter in the pre-filtering stage.

in the alerts increased or declined during these six months because it is unlikely that any reader got consistently poor or consistently great versions of the alert. For this reason, we do not expect any changes in open rate or unsubscribe rate.

Pre-filtering

Regardless of the selection algorithm, we first want to ensure that all stories provided to the selector meet certain minimum quality guarantees. Each candidate story must pass through a series of filters. The number of documents that satisfy each filter each week are shown in Figure 5. Stories are first obtained from the search index by querying on the topic "artificial intelligence" from the topic hierarchy. Each week, this yields about 1,500 stories, labeled "Found" in the figure. Next, a set of simple title and URL filters are applied to remove any stories that appear to be blog posts, job postings, press releases, etc. ("Filtered"). Then, a blacklist of bad domains is referenced to remove any stories coming from these domains ("GoodDomains"). This blacklist is manually curated as we discover new websites with inappropriate or irrelevant content. These new websites usually come from crawling the Twitter hashtag #artificialintelligence. Next, we automatically generate a summary of each story using a modified version of Luhn's technique (Luhn 1958). This summary is examined for blacklist words like "I," "we," "leaked," and others, which often indicate the story is a personal opinion or rumors. Any stories having these words in their summaries are filtered out ("GoodSummaries"). Finally, the system examines the topic classifications for each story. Since we want stories that have significant AI content, we require that the confidence of at least one AI topic is above 80%. Only stories with sufficiently high confidence are retained ("ReqClass"). As shown in the figure, these pre-filters reduce the average of 1,500 stories to about 200.

Algorithm 1: Random

Given a set of pre-filtered stories from the week, random selection is as simple as it sounds: a random 10 stories are selected from this set. They are then ordered by date.

Algorithm 2: TopClass

The TopClass algorithm scans the pre-filtered stories and collects all topic classifications for the stories. Recall that each story necessarily has some classification in the AI subhierarchy of AITopics' technology hierarchy, but it may also have other classes from the technology hierarchy or industry hierarchy. Next, the TopClass algorithm finds the 10 most common classes, and then picks a representative story for each class, i.e., the story that is most confidently classified into that class. Finally, these representative stories are ordered by date.

Algorithm 3: Genetic

The genetic algorithm approach attempts to balance several criteria for selecting diverse and representative stories. Before explaining these criteria, we first describe a few important aspects of the algorithm. Any genetic algorithm must have a way of representing an "individual" in the population, an initial population, a crossover function, a mutation function, and a fitness function.

Genetic representation. In our case, an "individual" is a set of 10 stories, so our genetic representation is simply a fixed-size list of stories, ordered by date.

Initial population. In the first generation, the initial population consists of 100 random sets of 10 stories each.

Crossover function. The crossover function takes two individuals and produces two new individuals, each of which share much of the information from the original individuals. Given two sets of 10 stories, a random pivot point p is selected, between 1 and 9 (inclusive), and two new individuals are generated: stories in positions $[1, p]$ in the first individual combined with stories $(p, 10]$ in the second individual; and the second new individual takes stories $[1, p]$ from the second and $(p, 10]$ from the first. Any duplicate stories in the new individuals are removed and replaced with random (non-duplicating) stories from the week's collection.

Mutation function. The mutation function randomly perturbs an individual so that a wide range of the possible variation of individuals is explored as the genetic algorithm runs through generations. Our mutation function simply randomly selects a story in the set of 10 that make up the individual and replaces it with a story not already in that set.

Fitness function. Finally, and most importantly, we define a fitness function that takes into account the various criteria we wish to optimize.

One of the most important aspects of our fitness function is a calculation of diversity. We wish to optimize for diversity in the alert in terms of dates of the stories (so not all stories are from the same day), topic classes (so not all stories are about the same kind of technology or industry), URL domains (so not all stories are from the same source), and words in titles and summaries (so not all stories cover the same event, even if their topic classifications somewhat differ due to different perspectives of that event). In order to

Symbol	Meaning
C_{avg}	Mean confidence of classes
D_{avg}	Mean score of domains
T_{div}	Diversity of dates
C_{div}	Diversity of classes
D_{div}	Diversity of domains
W_{div}	Diversity of words in titles
W'_{div}	Diversity of words in summaries

Table 1: Symbols and definitions for the various criteria that make up the fitness function.

calculate diversity, we first define a function that counts the number of elements in a vector $\bar{x}$ that equal a particular s:

$$C(\bar{x}, s) = \sum_{i=1}^{|\bar{x}|} [x_i = s], \tag{1}$$

where $[P] = 1$ whenever P is true, 0 otherwise. Next, we make use of the Gini-Simpson diversity index (Jost 2006) to measure the probability that two stories picked from the set of 10 have different dates, classes, domains, and/or words in their titles and summaries. The diversity index is defined as:

$$D(\bar{x}) = 1 - \frac{\sum_{i=1}^{|\bar{x}|} C(\bar{x}, x_i)(1 - C(\bar{x}, x_i))}{|\bar{x}|(1 - |\hat{x}|)}. \tag{2}$$

For example, when calculating the diversity of dates, we let $\hat{x}$ contains all the dates of the stories in an individual. Then the fraction gives us the ratio of stories with a particular date to those without, averaged across all represented dates. Thus, $D(\bar{x})$ is a larger value (closer to 1.0) when more stories have distinct dates.

With these equations in mind, we define our fitness function as the product of several criteria. The genetic algorithm will attempt to minimize the value of this function, so we negate the product to optimize for maximum fitness. We consider each of the criteria to be equally important, so they are all weighted equally. The criteria and their corresponding symbols are shown in Table 1. Recall that the diversity calculation ranges from 0 to 1, and each topic classification has a confidence value between 0 and 1 (previously expressed as a percent). A single story will likely have multiple classifications, each with a confidence value; and a set of 10 stories will have even more classifications and confidence values. We wish to find the average confidence value and optimize for a larger average. This way, we will prefer stories that are strongly about one or more topics and not just general overviews of a range of topics. Likewise, we use a manually curated list of URL domains and scores between 0 and 1 to compute the average domain score for a set of 10 stories.

The fitness function F is defined follows:

$$F = -C_{\text{avg}} * D_{\text{avg}} * T_{\text{div}} * C_{\text{div}} * D_{\text{div}} * W_{\text{div}} * W'_{\text{div}} \tag{3}$$

At each generation, the genetic algorithm selects 25 pairs of individuals and generates new individuals from crossover and mutation. The selection algorithm is classical tournament selection, in which the most fit individuals from random subsets of the population are chosen for crossover.

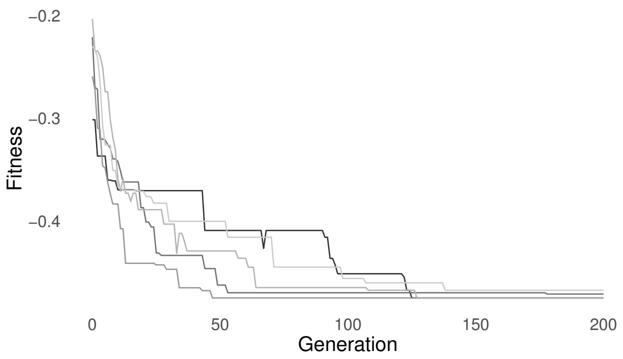

Figure 6: Minimum fitness values per generation for five runs with different random seeds.

Results

We run the genetic algorithm for 500 generations, but as can be seen in Figure 6, about 150 generations suffices. Running 500 generations takes about 1.5 minutes on an Intel Xeon E5 with 96 GB memory. Including querying the search index and generating the HTML output, the whole process takes about 2 minutes.

If we run the algorithm on the same pre-filtered corpus of one week's stories, but vary the random seed, we get different initial populations and different individuals selected for crossover and mutation from the tournament selector. Thus, we can expect the output to differ on each run. In fact, an experiment with five different random seeds shows that most of the stories in the final output are identical or similar. Two pairs of stories oscillated in the five runs: in some outputs, the story "Australia unleashes starfish-killing robot to protect Great Barrier Reef," from japantimes.co.jp was included. In other runs, this story was replaced by "Why is Facebook keen on robots? It's just the future of AI," from circa.com. It is worth noting both stories are about robots. The second pair of stories that oscillated are "Toyota to invest $500m in Uber in driverless car deal," from bbc.co.uk and, "Toyota Joins Uber on Its Tortuous Journey to Self-Driving Cars" from wired.com. It is worth noting both stories are about the same event.

The real measure of success is whether or not readers actually *preferred* the alerts generated by the genetic algorithm over the random or TopClass algorithm. We can measure this preference by tracking clicks on the stories themselves. Known as "click rate," this metric is the ratio of users who clicked a story to users who received the email. Figure 7 shows a frequency plot of the click rate for each algorithm. We can see that the genetic algorithm often received a higher click rate than both random and TopClass algorithms.

Of course, a figure is not sufficient evidence that the genetic algorithm performs best. Table 2 lists some summary statistics for click rates for the different algorithms. We see that the genetic algorithm has the highest rate in all the statistics (min, mean, median, etc.). Furthermore, we can calculate statistical significance of these differences in click rates, as shown in Table 3. Using pair-wise t-tests, we calculated

Algorithm	Min.	1st Q.	Median	Mean	3rd Q.	Max.
Genetic	2.90	4.11	5.61	5.55	6.49	11.12
Random	2.83	3.96	4.68	4.87	5.79	8.23
TopClass	2.48	3.49	4.28	4.39	5.23	6.86

Table 2: Click rates (%) for emails generated by each algorithm over the experimental period. "1st Q." and "3rd Q." mean first and third quartile, respectively.

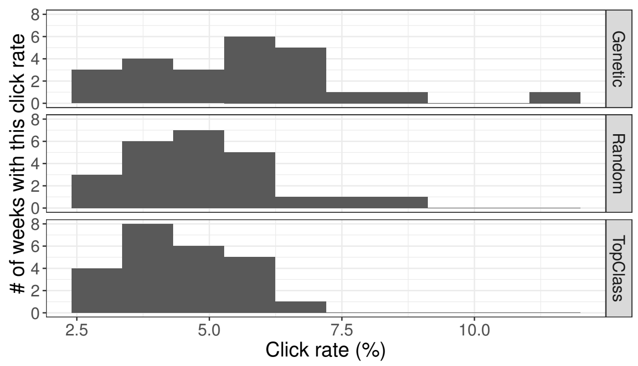

Figure 7: Frequency plot of the number of weeks (number of distinct alerts) that received various ranges of click rates. The results from each story selection algorithm are separated as vertical facets.

the difference in click rates per week for each algorithm. We see that the genetic algorithm performed better than both random and TopClass, and in the case of TopClass, by a significant margin (the difference between genetic and random selection was nearly but not quite statistically significant). Interestingly, the random algorithm performed better than TopClass, perhaps by providing more diversity in the selection. This outcome demonstrates that simplistic heuristics can sometimes do more harm than good.

We also compared open rates and unsubscribe rates throughout the weeks. As we described above, we did not expect these metrics to change depending on the selection algorithm since there was no information provided to the reader about which version of the alert they received each week, and since each reader received a different (random) version of the alert each week, there is little chance for the reader to "learn" that the alerts were improving or getting worse over time. Thus, their open rate or unsubscribe rate should not be affected by the selection algorithm. Indeed, Table 3 shows that these metrics did not significantly differ depending on which algorithm was used for each reader. Over the course of the experiment, the open rate for all alerts (regardless of the selection algorithm) declined about 0.01% per week, but this decline was not significant. The unsubscribe rate also declined by about 0.0003% per week, but this was also not significant.

In summary, these results show that the genetic algorithm produces stories that are more likely to be clicked. Thus, readers seem to like these stories more. It is worth noting again that nothing about the genetic algorithm or the set of

Measure	Algs. compared	Mean ±	p-value
Click rate	Genetic−Random	+0.68	0.0834
Click rate	Genetic−TopClass	+1.16	0.0345
Opens	Genetic−Random	+0.02	0.982
Opens	Genetic−TopClass	−0.29	0.706
Unsubscribes	Genetic−Random	−0.02	0.589
Unsubscribes	Genetic−TopClass	+0.01	0.660

Table 3: Results of pair-wise t-tests with 23 degrees of freedom. The measured variable was click rates (%).

pre-filters is specific to stories about artificial intelligence; we expect our approach would work equally well on stories about any subject.

Discussion

Our six-month experiment successfully validated our belief that the genetic algorithm approach is a better story selector than random or TopClass algorithms. Once the experiment concluded in July 2018, we activated the genetic algorithm for all readers of *AI-Alert*, and our weekly click rates and open rates remain strong while our unsubscribe rates remain low.

Our automation of the generation of *AI-Alert* continues to save us significant time compared to manually assembling and formatting the alert ourselves. Instead of spending many hours finding relevant candidates and selecting a few interesting stories for each weekly alert, our current approach only requires checking the alert's output and removing a bad story or two. This takes only about 10 minutes a week. We do not expect to fully automate the alert since we always want to maintain some editorial control over its contents. We value our readership too much to allow an obviously inappropriate news story or spam content to infiltrate the alert.

Maintenance

The code that generates *AI-Alert* on a weekly basis requires very little maintenance. The i2k Connect technology that finds and classifies news stories is maintained separately as a multi-purpose suite of technologies that support the alert, the AITopics website, and the Society of Petroleum Engineers' research portal,[5] among other use cases. Thus, those components are maintained and upgraded on a continuous basis primarily to support other use cases.

Maintenance specific to *AI-Alert* includes maintaining and upgrading the "snacktory" configuration files that help our system extract body text from news stories around the

[5]https://search.spe.org/i2kweb/SPE/search

web. For example, CNN's website has a specific layout that differs significantly from The New York Times' layout, and sometimes we need to define special patterns in order for snacktory to find the body text. We also maintain and curate two blacklists that apply to the pre-filtering stage of processing. These blacklists give bad URL domains and bad words in titles and summaries (not just offensive words, but also words that indicate opinions, rumors, etc.). We consider it important that the alert be able to include stories outside of a small set of whitelisted publishers, and the Twitter #artificialintelligence hashtag often provides stories from websites we have never seen before. Sometimes, these stories or websites are not appropriate for the alert, so we add a blacklist rule. This is done during the short editing time period each week. In the future, we hope to find a more automatic way of maintaining these blacklists, but it is not clear at this time how that may be done. Finally, we also maintain a list of URL domains and scores so that stories from subjectively-good sources like nytimes.com and bbc.co.uk are prioritized in the fitness function of the genetic algorithm over lower quality or unknown domains.

As mentioned above, sometimes *NewsFinder* assumes the wrong publication date for a story. These out-of-date stories are caught during our editing phase and removed from the alert. However, we expect that this problem can be solved with a bit more sophistication in our publication date detector. Different news publishers have different ways of writing a publication date on their story, if they even include a date at all. We are actively working on improving date parsing, but if no date is present, we default to the date that the story was discovered by *NewsFinder*.

All things considered, maintaining *NewsFinder* and editing the weekly *AI-Alert* requires very little time. It is mostly set-it-and-forget-it. A significant contributor to such low maintenance requirements is a alert generator, including our genetic algorithm for selecting stories, that considerably reduces the time required to create an alert that represents the important AI news of the week.

Conclusion

This paper described how we designed and implemented an automated weekly alert sent by email to thousands of subscribers. The alert focuses on the week's news about artificial intelligence, covering a diverse set of stories from high quality sources. The key to our strong reader engagement lies in a genetic algorithm that filters thousands of stories acquired throughout the week to just 10 stories for inclusion in the alert. These stories are selected according to several criteria such as topic, content, and date diversity and the reputation of the publisher. We demonstrated that the genetic algorithm produces alerts that result in more engagement, measured by click rate, by readers of the alert.

We believe *AI-Alert* is valuable to more readers in the general population than the nearly 11,000 subscribers we have at this time. Any person can sign up for free on the AITopics website, and our list of subscribers are periodically expanded to include new AAAI members and affiliates. However, some targeted marketing effort will be required to dramatically expand the reach of the alert.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1534798.

References

Agrawal, R.; Gollapudi, S.; Halverson, A.; and Ieong, S. 2009. Diversifying search results. In *Proceedings of the second ACM international conference on web search and data mining*, 5–14. ACM.

Buchanan, B. G., and Glick, J. 2002. AI Topics. *AI Magazine* 23(1):87.

Buchanan, B. G.; Eckroth, J.; and Smith, R. G. 2013. A virtual archive for the history of AI. *AI Magazine* 34(2).

Cecchini, R. L.; Lorenzetti, C. M.; Maguitman, A. G.; and Ponzoni, I. 2018. Topic relevance and diversity in information retrieval from large datasets: A multi-objective evolutionary algorithm approach. *Applied Soft Computing* 69:749–770.

Dou, Z.; Hu, S.; Chen, K.; Song, R.; and Wen, J.-R. 2011. Multi-dimensional search result diversification. In *Proceedings of the fourth ACM international conference on Web search and data mining*, 475–484. ACM.

Eckroth, J.; Dong, L.; Smith, R. G.; and Buchanan, B. G. 2012. NewsFinder: Automating an AI news service. *AI Magazine* 33(2):43.

Gabrilovich, E.; Dumais, S.; and Horvitz, E. 2004. Newsjunkie: providing personalized newsfeeds via analysis of information novelty. In *Proceedings of the 13th international conference on World Wide Web*, 482–490. ACM.

Jost, L. 2006. Entropy and diversity. *Oikos* 113(2):363–375.

Li, L.; Wang, D.; Li, T.; Knox, D.; and Padmanabhan, B. 2011. SCENE: A scalable two-stage personalized news recommendation system. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 125–134. ACM.

Luhn, H. P. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development* 2(2):159–165.

MailChimp. 2018. Average email campaign stats of MailChimp customers by industry. https://mailchimp.com/resources/research/email-marketing-benchmarks/.

Mishra, A., and Berberich, K. 2016. Event digest: A holistic view on past events. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 493–502. ACM.