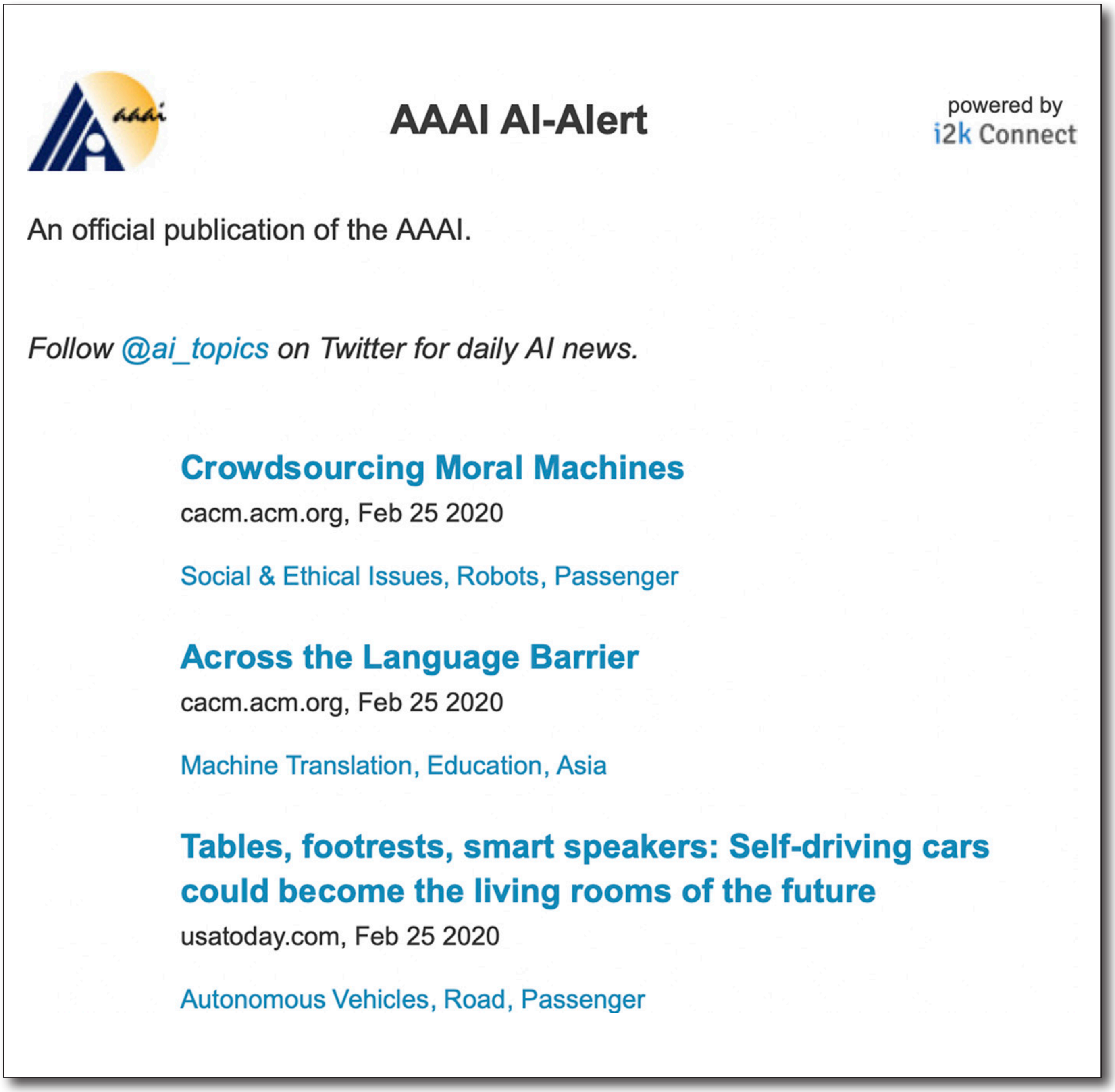
Evolution of a Robust Artificial Intelligence System: A Case Study of the Association for the Advancement of Artificial Intelligence's *AI-Alert*

Joshua Eckroth

■ *Since mid-2018, we have used a suite of artificial intelligence (AI) technologies to automatically generate the Association for the Advancement of Artificial Intelligence's AI-Alert, a weekly email sent to all Association for the Advancement of Artificial Intelligence members and thousands of other subscribers. This alert contains ten news stories from around the web that focus on some aspect of AI, such as new AI inventions, AI's use in various industries, and AI's impacts in our daily lives. This alert was curated by-hand for a decade before we developed AI technology for automation, which we call "NewsFinder." Recently, we redesigned this automation and ran a six-month experiment on user engagement to ensure the new approach was successful. This article documents our design considerations and requirements, our implementation (which involves web crawling, document classification, and a genetic algorithm for story selection), and our reflections after a year and a half since deploying this technology.*

Robust artificial intelligence (AI) systems deployed in practice develop incrementally, from careful specification, development, experimentation, and refinement, to programs that are useful and used. The Association for the Advancement of Artificial Intelligence (AAAI)'s *AI-Alert* service is one such system read by thousands of AI students, faculty, and practicing professionals. The alert is received by about 12,000 subscribers each Tuesday and includes ten news stories about AI. Figure 1 shows part of a recent alert.

AI-Alert is sponsored by AAAI and implemented and maintained by i2k Connect, Inc. Stories that make up the alert are pulled from AITopics.org, which is populated by an automated system. AITopics is a repository and advanced search engine for AI-related news, as well as conference, journal, and preprint articles, and classic texts. AITopics was founded more than twenty years ago and has undergone significant evolution in this time. Today, content is automatically acquired, enriched with metadata including topics and concept tags (keywords), and indexed on AITopics, resulting in about 150 to 250 new documents added to the site each day. These documents are mostly news stories from specific sources such as BBC and *The New York Times*, as well as stories posted on the #artificialintelligence Twitter hashtag (about 200,000 stories combined), but also include articles from AAAI conference publications (about 40,000 articles), *AI Magazine* (about 5,000



AAAI AI-Alert

powered by i2k Connect

An official publication of the AAAI.

Follow @ai_topics on Twitter for daily AI news.

Crowdsourcing Moral Machines
cacm.acm.org, Feb 25 2020

Social & Ethical Issues, Robots, Passenger

Across the Language Barrier
cacm.acm.org, Feb 25 2020

Machine Translation, Education, Asia

Tables, footrests, smart speakers: Self-driving cars could become the living rooms of the future
usatoday.com, Feb 25 2020

Autonomous Vehicles, Road, Passenger

Figure 1. Top Portion of a Recent AI-Alert Email.

articles), Neural Information Processing Systems conferences (about 15,000 articles), articles published on arXiv.org in the machine learning (ML) and AI categories (about 45,000 articles combined), and about 900 classic articles and books.

Because each story in the alert comes from AITopics, each story has been automatically classified into multiple topics across a few topic hierarchies: Technology, which covers 256 topics (six levels deep) and identifies the kinds of technologies mentioned in the story, such as deep learning and face recognition; Industry, which covers 828 topics (six levels deep) and identifies the areas of application for the AI technology, such as road transportation (for example, self-driving cars), upstream oil and gas production, and voting machines; and Location, which covers 9,302 locations around the world (six levels deep), from continents to countries to states, regions, counties, and cities. The top three topics are shown below each story in the alert; clicking one of these topics sends the reader to AITopics' collection of stories and articles about that topic. Readers can refine their search to combine filters such as finding recent stories about uses of speech recognition in self-driving cars deployed in Germany. Each *AI-Alert* generated since early 2017 is archived on AITopics.[1]

AAAI members are automatically subscribed to *AI-Alert*, and anyone may subscribe by visiting AITopics. Figure 2 shows the history of subscribers since we moved from a listserv email system to a modern mass-mailing provider. Large jumps in subscriber counts are due to mass imports of new AAAI members. Over about 2.5 years, about 1,700 or eleven percent of readers have unsubscribed. *AI-Alert* consistently beats computer and electronics industry marketing email averages (MailChimp, 2018) for open rate (our average thirty percent versus their nineteen percent) and click rate (our 4.2 percent versus their 2.0 percent).

During the timespan of our recent experiment, which we detail in this article, we gathered a variety of more specific user-engagement metrics. These metrics were published previously in our Innovative Applications of Artificial Intelligence conference paper about our experiment (Eckroth and Schoen, 2019). An average of thirty percent of recipients open any given week's alert, although most (about sixty percent) have never clicked a link in any alert (albeit 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 3.

The x-axis shows the number of clicks, and the y-axis shows the number of readers with that click count. Note that the x-axis is logarithmic.

Each alert contains ten 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 ten. Thus, in nearly every alert, nine or ten out of ten 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 4. Among stories that were clicked, most were clicked about thirteen times (mean) by different readers, although, rarely, a story was clicked many more times. Figure 5 shows this distribution.

In summary, the active readership of *AI-Alert* is relatively small. About one-third open the email, and about forty percent have ever clicked one or more stories. Thus, about twelve percent 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 emails containing links to news stories about a given subject because, to our knowledge, such detailed statistics have never been published.

A Brief History of *AI-Alert*

AI-Alert began in 1998 as a simple web page on the newly founded AITopics website. Bruce Buchanan, AAAI Fellow and former president and secretary-treasurer of AAAI, and Jonathan Glick updated the alert, then called AI in the News, on a regular basis (Buchanan and Glick, 2002). Figure 6 shows a screenshot of AITopics from 2001. In 2007, Buchanan, Glick, and Reid Smith (AAAI Fellow) migrated AITopics to a wiki to facilitate more efficient editing (figure 7), but the alert was still manually produced. In 2010, Liang Dong helped Buchanan and Smith create web crawlers and automatic document classification using support vector machines to identify AI-related news. They named this technology "NewsFinder" and began the era of *AI-Alert* automation (Dong, Smith, and Buchanan, 2011). This work was then refined by me (Eckroth et al., 2012).

Due to the limitations of support vector machines, particularly their need for abundant training data, NewsFinder was only able to classify stories into one of nineteen categories (for example, robots, vision, and education). This restriction led Buchanan, Smith, Schoen, and me to develop a rule-based approach and hierarchical scoring algorithm for document classification. This technology was used to organize news stories about AI into Industry and Technology hierarchies, among others, so that a single story could be classified into, for example, deep learning, ethics and social issues, and drones. Buchanan and Smith then founded i2k Connect, Inc. and joined with me to further develop this technology for classifying and organizing all types of documents. The AITopics website was migrated to the Drupal engine (figure 8) to facilitate automatic inclusion of news stories from the web. Eventually, our technology outgrew Drupal and we developed our own engine to support more advanced filtering and searching as well as visual analytics (figure 9). This latest iteration is the version of AITopics active today, and this article includes several visualizations generated by the website.

Although AITopics had undergone significant improvements over the years and was able to crawl and classify news stories from hundreds of web sources, the *AI-Alert* selection algorithm was still relatively simplistic. It sometimes generated alerts with too many stories on the same subject or from the same sources. Every few weeks we would receive feedback from readers that there were too many robot stories, or a particular event such as IBM's Watson appearance on Jeopardy! was overrepresented. We realized we had a multicriteria optimization problem in which topics, sources, dates, and other factors had to be balanced to get a diverse and representative set of ten stories for each weekly alert. A common technique for solving multicriteria optimization problems is genetic algorithms, so we developed an implementation and experiment to convince ourselves that this new algorithm was the right approach.

A Checklist for Building AI Applications

Not long before we began developing a new approach for automating *AI-Alert* to solve the multicriteria optimization problem, Reid Smith was asked to give the

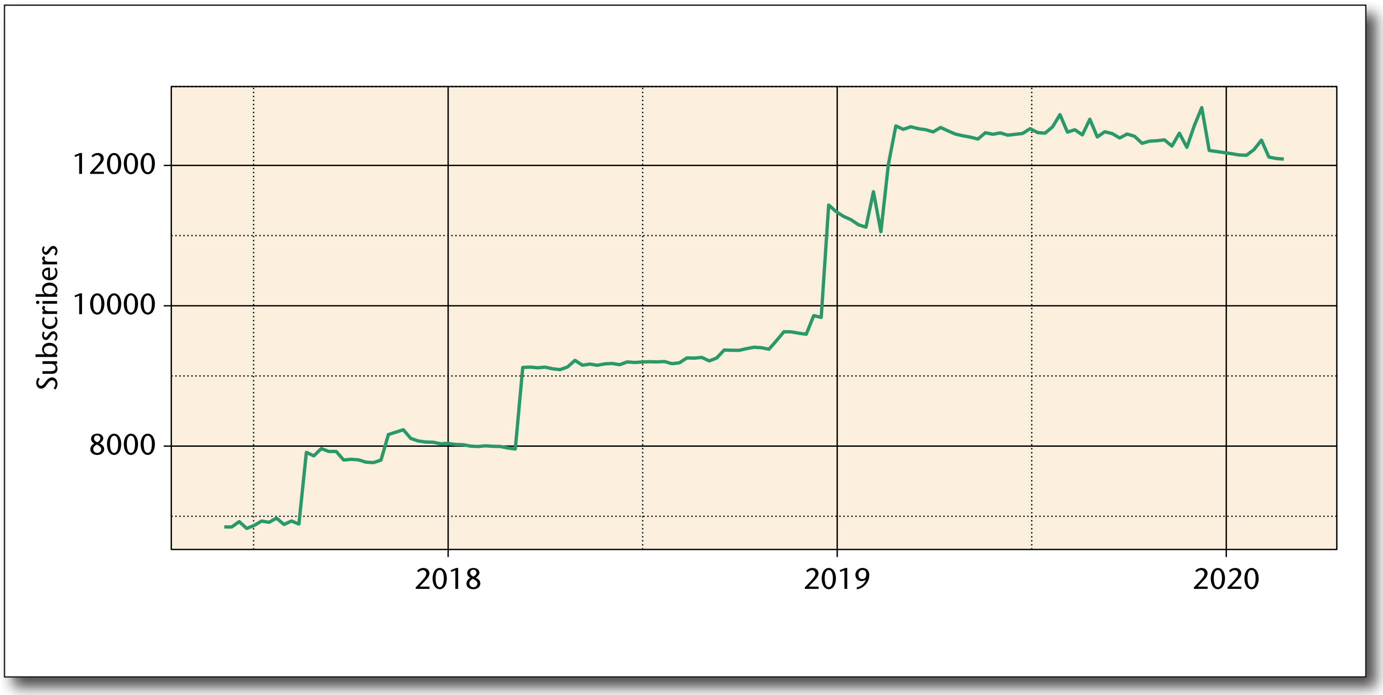


Figure 2. Count of AI-Alert Subscribers Since Switching to a Modern Mass-Email Service.

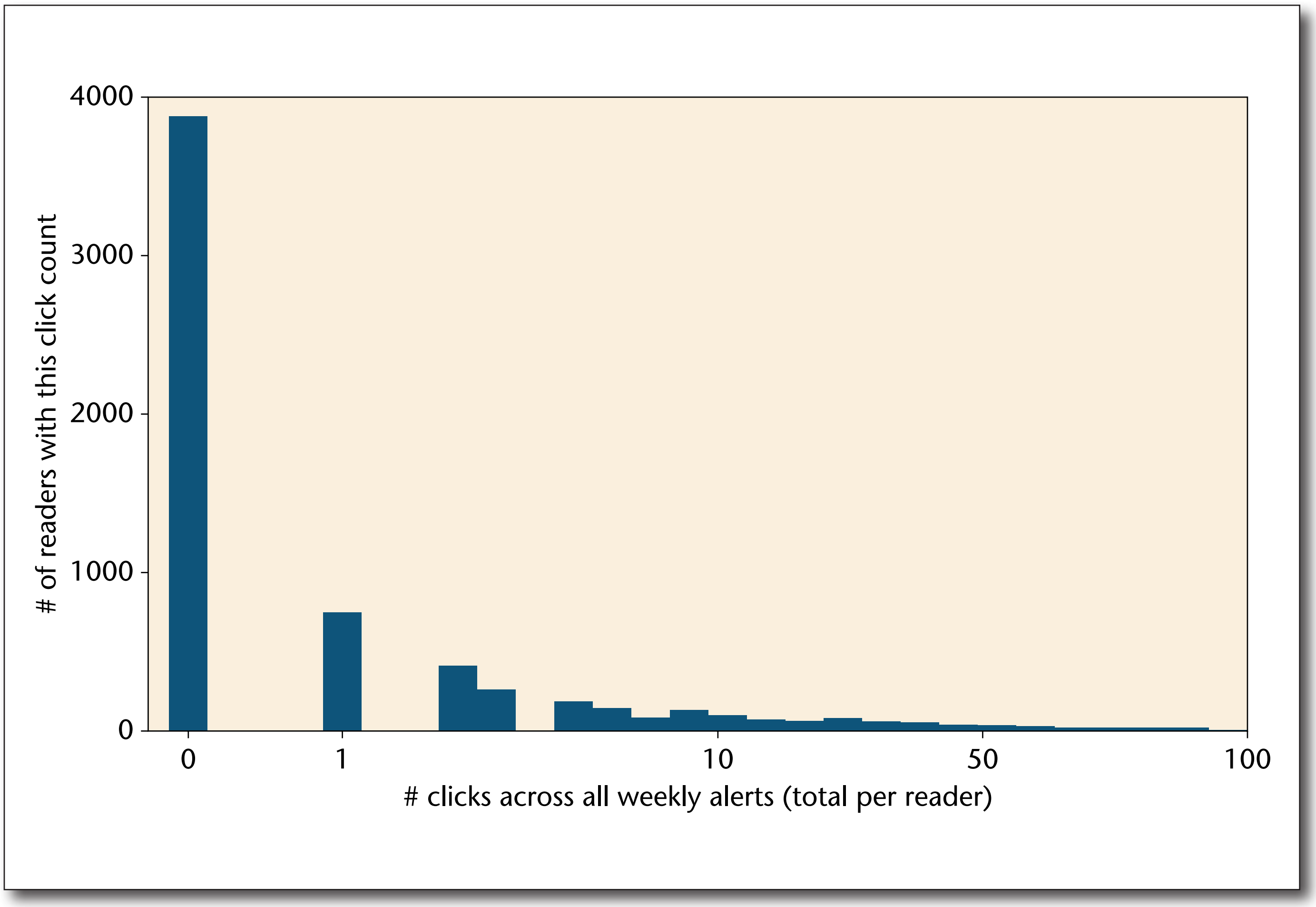


Figure 3. Frequency Plot of the Number of Readers Who Have Clicked Links in the Alerts.

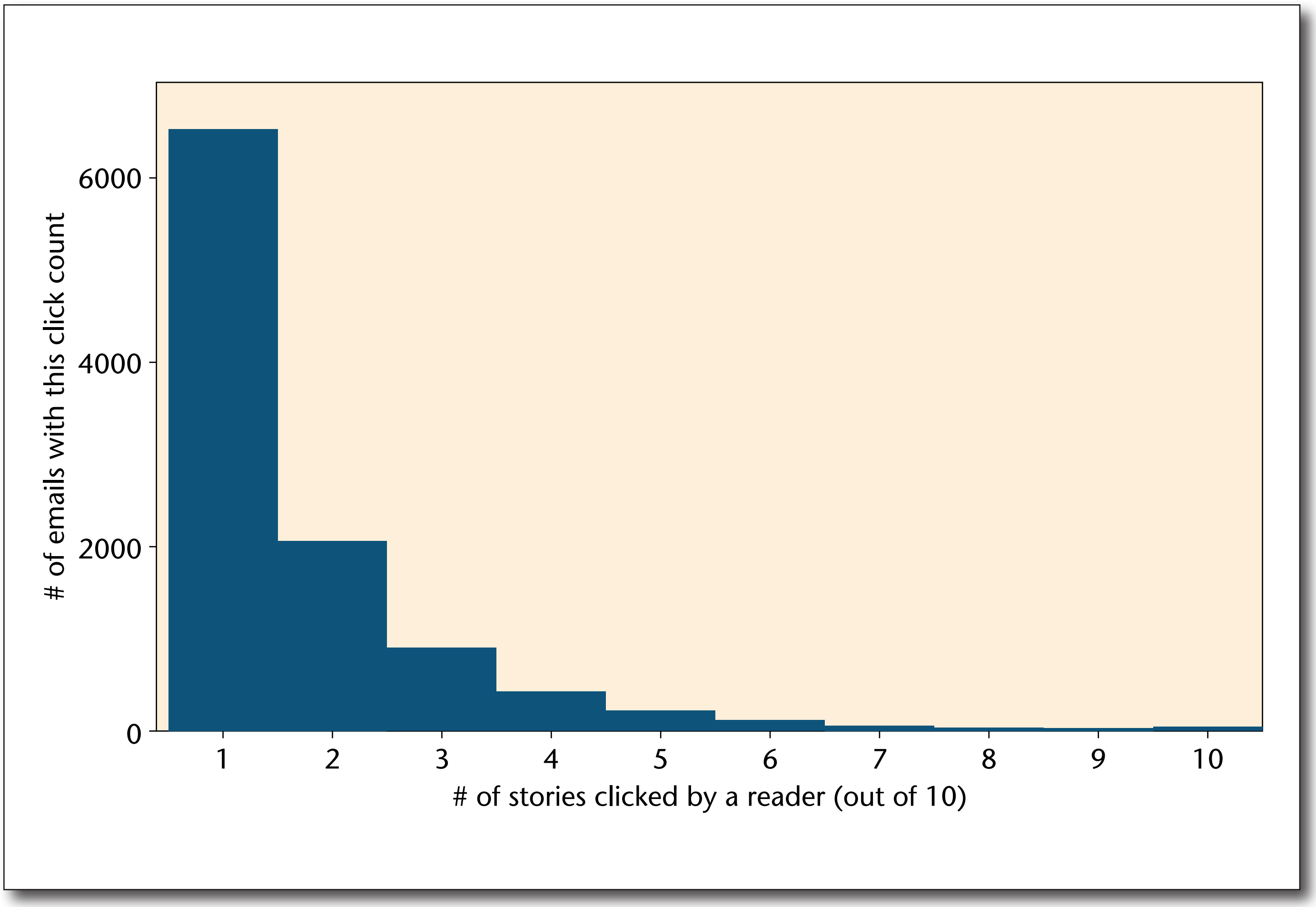


Figure 4. Frequency Plot of the Number of Stories (Out of Ten) in Each Alert That a Reader Clicks.

It is clear that most readers who click a story just click one story.

Robert S. Engelmore Memorial Lecture at the Innovative Applications of Artificial Intelligence conference in 2016. In this talk, Smith reviewed the history of deployed AI applications represented in the Innovative Applications of Artificial Intelligence conference. This talk transformed into an article in *AI Magazine* (Smith and Eckroth, 2017), where the authors developed a checklist for assisting developers and managers as they plan, build, and deploy AI applications. I further expanded on this checklist in my book, *AI Blueprints: How to Build and Deploy AI Business Projects* (Eckroth, 2018). We next review this checklist as it applies to our redesign of the algorithms behind *AI-Alert*. This review helps explain our motivations and implementation choices for the alert and also familiarizes readers with the checklist so it may be applied in other projects.

The checklist helps ensure AI is being used appropriately and that the system is deployed successfully and monitored during its subsequent evolution. The checklist is grouped into four sections.

Characterize the Problem

First, we should check that AI technology is appropriate for the problem at hand. AI and ML may introduce significant technical debt (Sculley et al., 2014), such as hidden assumptions and dependencies on data formats and processing workflows. For business purposes, the AI problem should be known to be solvable rather than an open-ended research problem, and it should have clear boundaries so everyone is clear what role the AI plays in the larger system.

The AI Solves a Clearly Stated Business Problem

We wanted a new algorithm for generating *AI-Alert* because the old algorithm was not producing a sufficiently diverse set of stories. The other requirements for the alert, namely that each story should focus on one or more aspects of AI, and each alert should contain ten stories from the prior week, were to be maintained from the existing implementation.

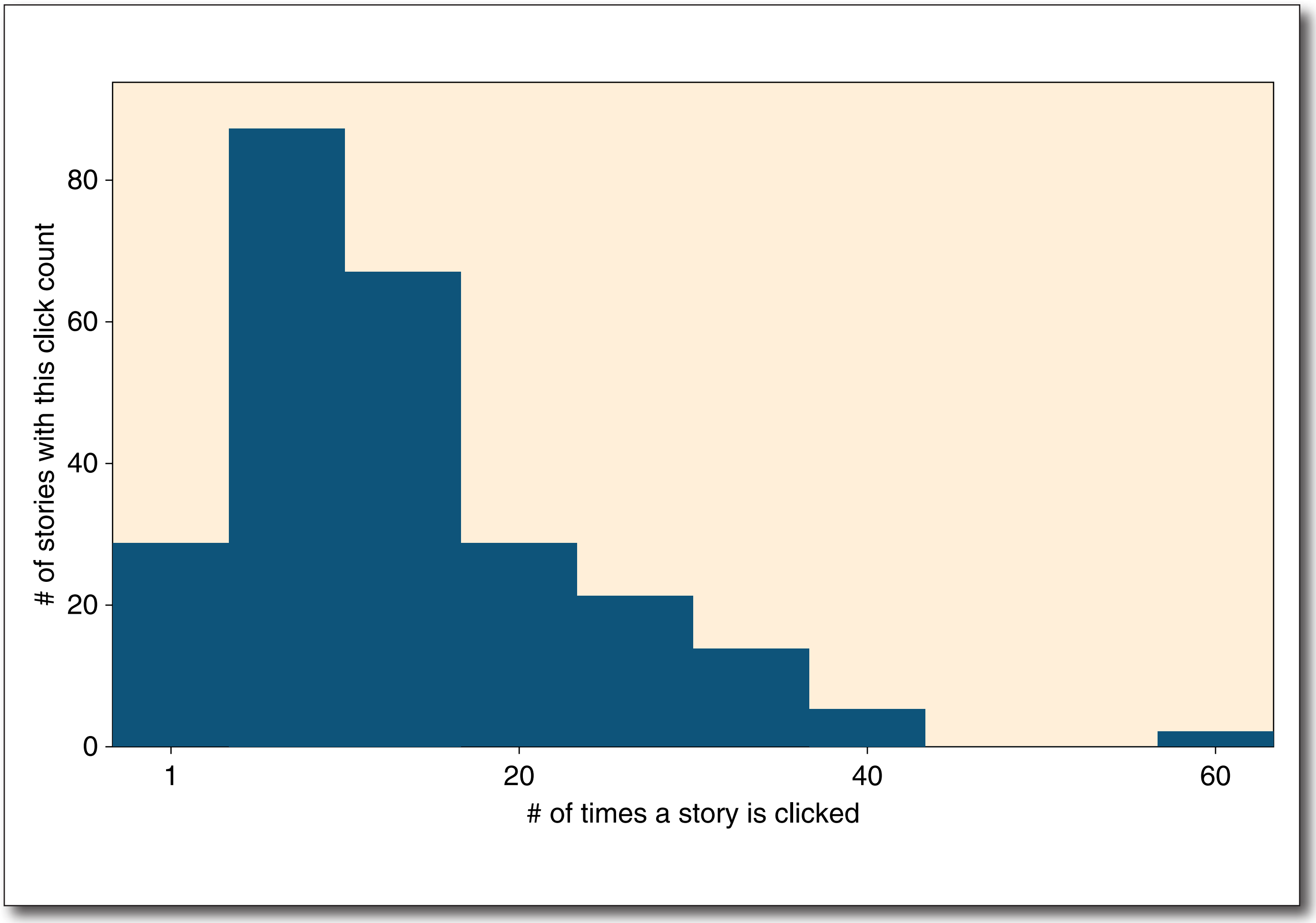


Figure 5. Frequency Plot of the Number of Clicks a Story Receives.

Most stories receive between five and twenty clicks, while only a few stories receive many more clicks.

The Problem Is Known To Be Solvable by AI

Once we framed the problem as a multicriteria optimization problem, we believed AI was appropriate for the task. The prior algorithm used in NewsFinder did not use AI. It was rather simple: It first found the most common topics represented by the week's stories, then picked a representative from each topic until it had picked ten stories. The representative was chosen based on how strongly the story matched one of the most popular topics. This algorithm made intuitive sense when we designed it, but in practice it proved to lack diversity.

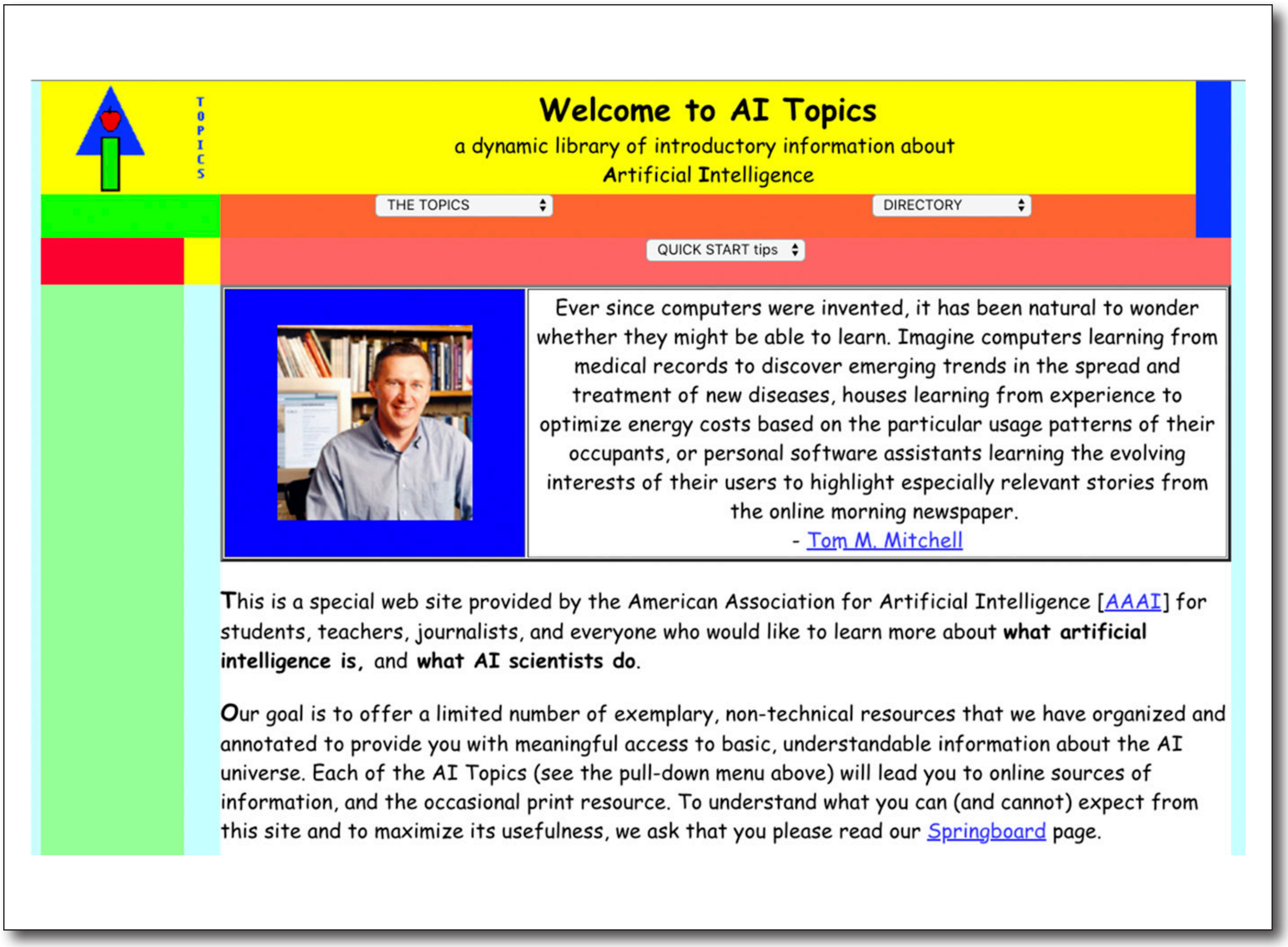
To guide our implementation, we characterized our multiple criteria as follows. Each week, about 1,500 stories must be filtered down to the ten best candidates. Each story should be closely, not just tangentially, related to AI. No single AI topic should dominate the alert; that is, diversity should 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, although high-quality publishers should be preferred. Because 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.

The AI Uses Established Techniques

There are several approaches for solving multicriteria optimization problems, one of which is genetic algorithms. Because we have had prior experience developing genetic algorithms, we chose this method.

The Role of the AI within the Larger System Is Clearly Defined and Bounded

It is important to clarify the role that each AI subsystem plays in a larger system. Much of a software system is made up of traditional algorithms (acquiring and updating data, sorting, filtering, reporting, and so forth). An AI component adds a bit of "magic" that can easily be seen as either a promising fix for any



Welcome to AI Topics

a dynamic library of introductory information about Artificial Intelligence

Ever since computers were invented, it has been natural to wonder whether they might be able to learn. Imagine computers learning from medical records to discover emerging trends in the spread and treatment of new diseases, houses learning from experience to optimize energy costs based on the particular usage patterns of their occupants, or personal software assistants learning the evolving interests of their users to highlight especially relevant stories from the online morning newspaper.
- Tom M. Mitchell

This is a special web site provided by the American Association for Artificial Intelligence [AAAI] for students, teachers, journalists, and everyone who would like to learn more about **what artificial intelligence is**, and **what AI scientists do**.

Our goal is to offer a limited number of exemplary, non-technical resources that we have organized and annotated to provide you with meaningful access to basic, understandable information about the AI universe. Each of the AI Topics (see the pull-down menu above) will lead you to online sources of information, and the occasional print resource. To understand what you can (and cannot) expect from this site and to maximize its usefulness, we ask that you please read our Springboard page.

Figure 6. AITopics Website From 2001, With Hand-Written HTML Pages.

software faults or the source of every problem, and possibly both perceptions simultaneously. In other words, it is not helpful for developers to think that the AI will be able to handle that (the unrelated task), or that the fault (in this unrelated subsystem) must be the AI again. Everyone should be clear about what the AI is and is not contributing to the larger system. In our case, the AI subsystem that we were changing in NewsFinder had the singular role of selecting stories from a candidate set. It is not responsible for finding the stories on the web, classifying the stories into AI-related topics, formatting the email, nor ensuring its delivery to subscribers.

Develop a Method

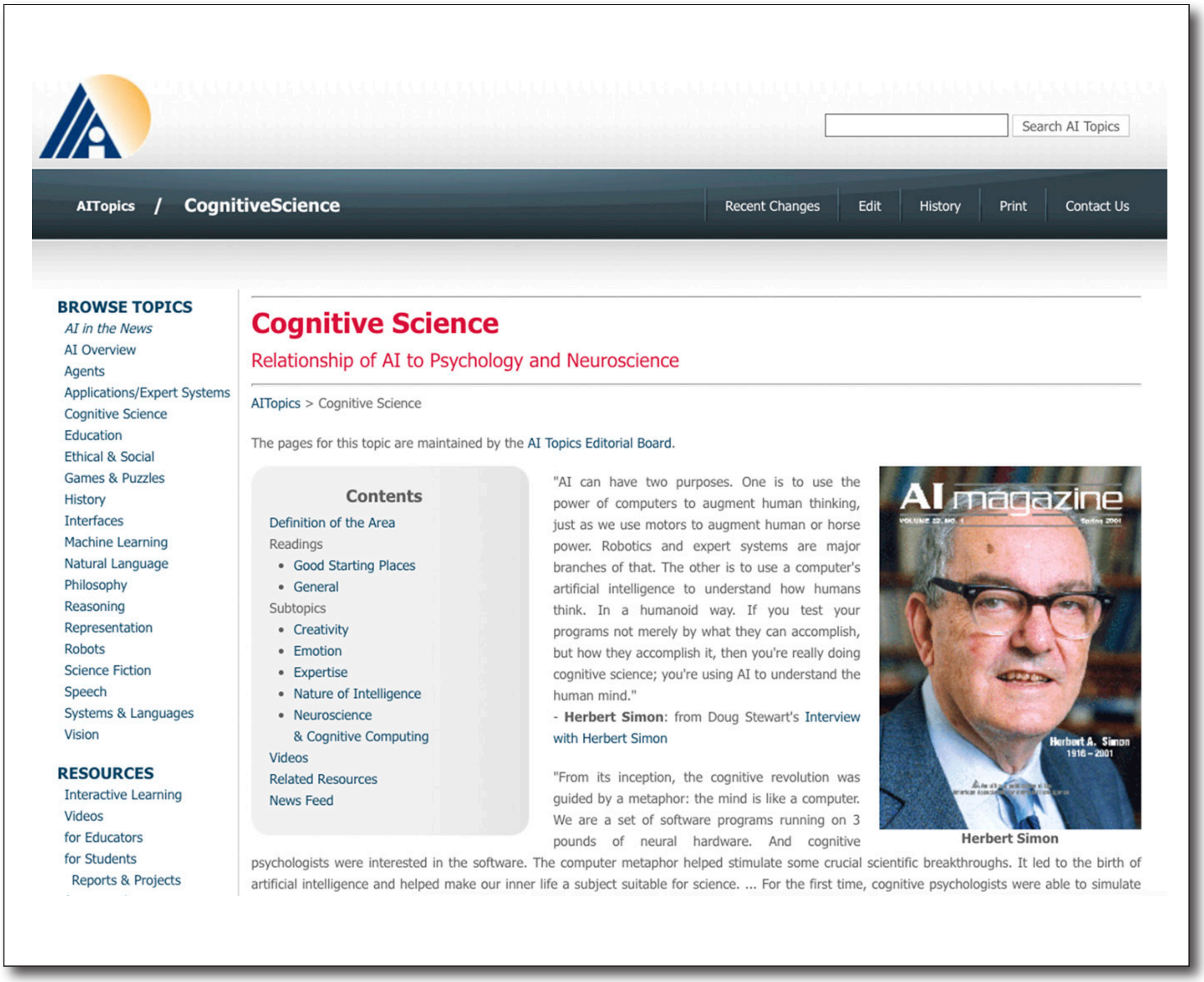
Once the goal and role of the AI component has been clarified, one must develop a method and implementation. Here, too, there are some considerations to keep in mind. A successful implementation is more likely if significant, ground-breaking research is not required and the implementation is able to use established techniques. Also, sufficient high-quality training data might be required for ML, and some AI techniques require significant computational resources.

The Method Does Not Require Significant New Research

While it is easy to imagine exotic applications of futuristic AI (for example, "what if I could write an essay using just my thoughts?"), most businesses do not have the time or resources to engage in open-ended research projects. There are many AI and ML techniques that have proven successful in a wide range of applications. Attempting to venture beyond this well-understood ground opens one to the risk of failure and a lot of wasted time and money. In our case, we knew that implementing and experimenting with genetic algorithms would not take significant time (just a few months), so the risk was low.

The Method Is Relatively Mature and Commonplace

AI and ML techniques are being invented at a rapid pace. The preprint service arXiv.org and AI and ML


Search AI Topics

AITopics / CognitiveScience

Recent Changes | Edit | History | Print | Contact Us

BROWSE TOPICS

AI in the News
AI Overview
Agents
Applications/Expert Systems
Cognitive Science
Education
Ethical & Social
Games & Puzzles
History
Interfaces
Machine Learning
Natural Language
Philosophy
Reasoning
Representation
Robots
Science Fiction
Speech
Systems & Languages
Vision

RESOURCES

Interactive Learning
Videos
for Educators
for Students
Reports & Projects

Cognitive Science

Relationship of AI to Psychology and Neuroscience

AITopics > Cognitive Science

The pages for this topic are maintained by the AI Topics Editorial Board.

Contents

Definition of the Area
Readings
- Good Starting Places
- General

Subtopics
- Creativity
- Emotion
- Expertise
- Nature of Intelligence
- Neuroscience & Cognitive Computing

Videos
Related Resources
News Feed

"AI can have two purposes. One is to use the power of computers to augment human thinking, just as we use motors to augment human or horse power. Robotics and expert systems are major branches of that. The other is to use a computer's artificial intelligence to understand how humans think. In a humanoid way. If you test your programs not merely by what they can accomplish, but how they accomplish it, then you're really doing cognitive science; you're using AI to understand the human mind."
- **Herbert Simon**: from Doug Stewart's Interview with Herbert Simon

"From its inception, the cognitive revolution was guided by a metaphor: the mind is like a computer. We are a set of software programs running on 3 pounds of neural hardware. And cognitive psychologists were interested in the software. The computer metaphor helped stimulate some crucial scientific breakthroughs. It led to the birth of artificial intelligence and helped make our inner life a subject suitable for science. ... For the first time, cognitive psychologists were able to simulate

Figure 7. AITopics Website From 2007, Using a Wiki Engine.

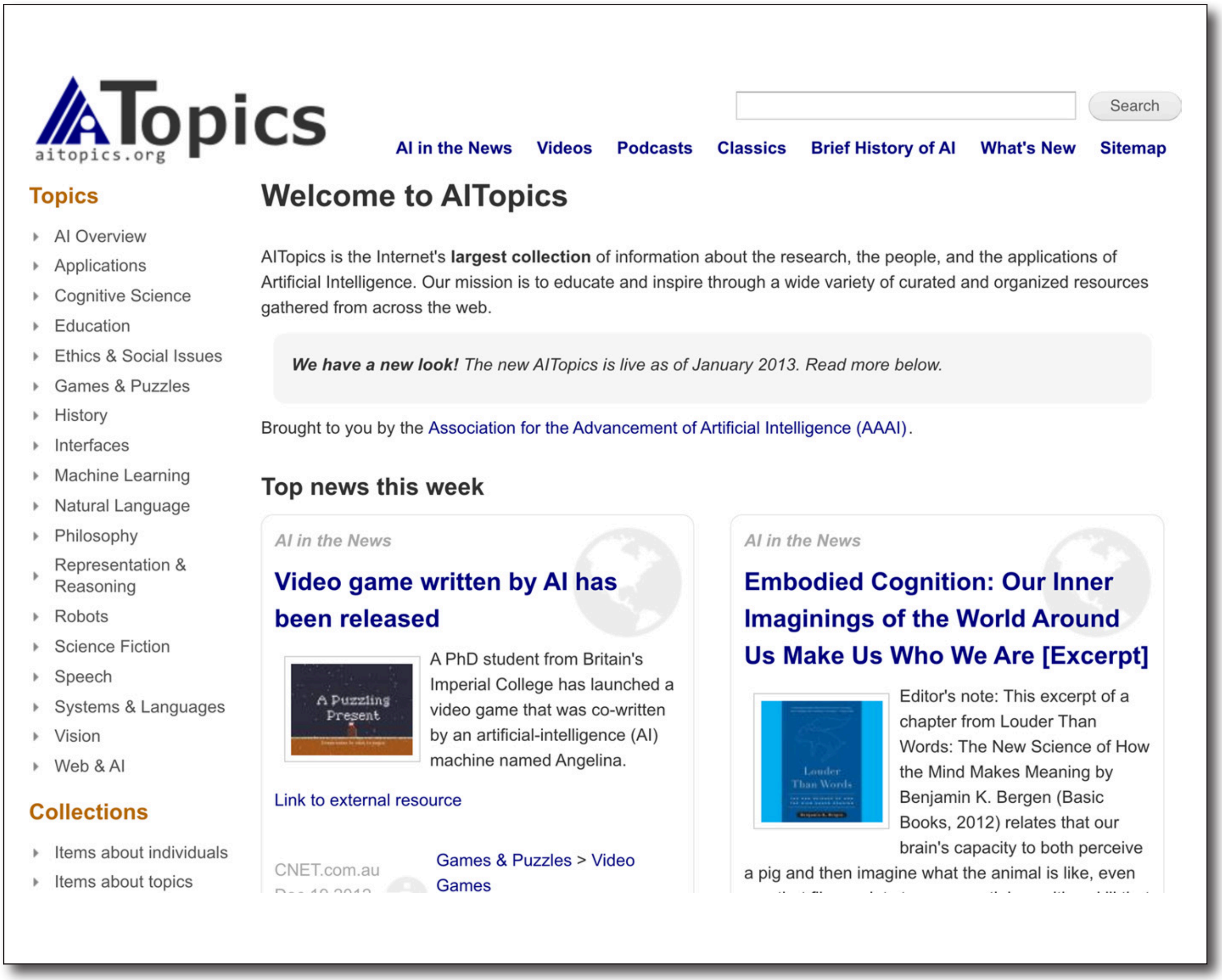
conferences are full of new applications of many techniques, particularly deep learning. With the rise of GitHub, code and datasets often accompany the paper, allowing any developer to try the technique in a new situation. However, research-quality code is not necessarily production-quality code, and the technique might work only in limited applications. There is less risk in adopting a mature technique and software platform. For NewsFinder, we were able to use a simple and straightforward implementation of genetic algorithms in a software library that was compatible with our existing software architecture.[2]

The Necessary Hardware Resources and Training Data Are Available

Many applications of AI, particularly those that involve ML, require abundant training data and powerful hardware such as graphics processing units. Our use case does not have these same requirements, as our datasets are relatively small and the genetic algorithm finishes in just a few seconds on moderately fast server hardware. However, we have seen in other applications that the quality and quantity of the training data has a significant impact on the success of a ML project. One may even need to invent ways to acquire training data if the application area is new.

Design a Deployment Strategy

Even the most intelligent AI may never be used. Potential users of a new AI tool are often surprisingly unwilling to change their habits. The new system must integrate seamlessly into their existing workflow or offer such amazing new advantages that they are willing to learn new habits. Also, when planning to deploy the AI subsystem into production, one must be careful to ensure data are provided to the subsystem in a consistent manner and outputs from the AI subsystem are checked for validity.


AITopics
aitopics.org

Search

AI in the News | Videos | Podcasts | Classics | Brief History of AI | What's New | Sitemap

Topics

- AI Overview
- Applications
- Cognitive Science
- Education
- Ethics & Social Issues
- Games & Puzzles
- History
- Interfaces
- Machine Learning
- Natural Language
- Philosophy
- Representation & Reasoning
- Robots
- Science Fiction
- Speech
- Systems & Languages
- Vision
- Web & AI

Collections

- Items about individuals
- Items about topics

Welcome to AITopics

AITopics is the Internet's **largest collection** of information about the research, the people, and the applications of Artificial Intelligence. Our mission is to educate and inspire through a wide variety of curated and organized resources gathered from across the web.

We have a new look! *The new AITopics is live as of January 2013. Read more below.*

Brought to you by the Association for the Advancement of Artificial Intelligence (AAAI).

Top news this week

AI in the News

Video game written by AI has been released

A PhD student from Britain's Imperial College has launched a video game that was co-written by an artificial-intelligence (AI) machine named Angelina.

Link to external resource

CNET.com.au

Games & Puzzles > Video Games

AI in the News

Embodied Cognition: Our Inner Imaginings of the World Around Us Make Us Who We Are [Excerpt]

Editor's note: This excerpt of a chapter from Louder Than Words: The New Science of How the Mind Makes Meaning by Benjamin K. Bergen (Basic Books, 2012) relates that our brain's capacity to both perceive a pig and then imagine what the animal is like, even

Figure 8. AITopics Website From 2013, Using the Drupal Engine.

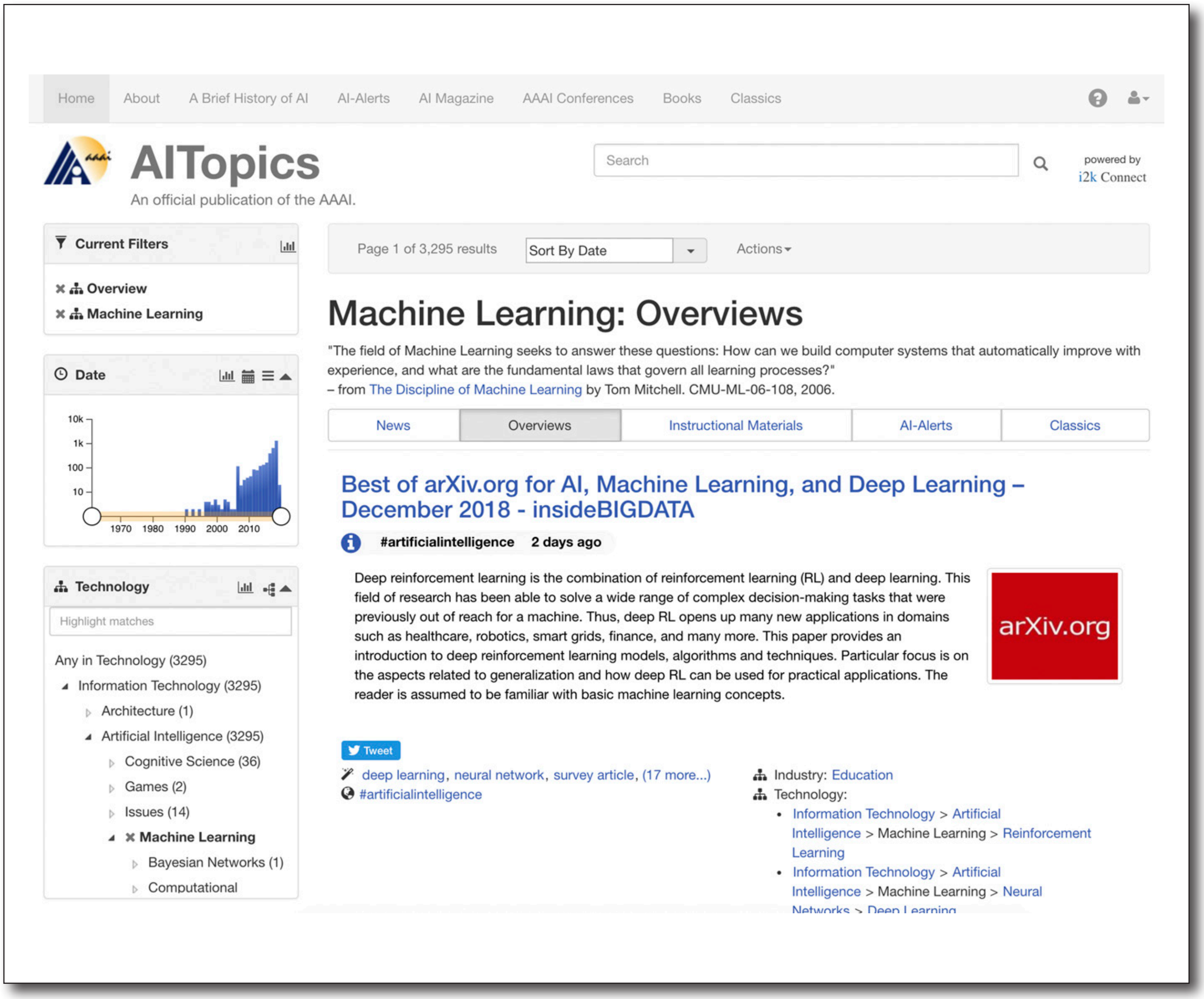
Plan a user experience, if the AI is user-facing, which fits into an existing habit or workflow, requiring very little change by the user. NewsFinder is automated, requiring no user interaction, until the point at which the generated email must be reviewed and sent. We keep a human in the loop to verify that it continues to operate as expected and that the stories included in the alert are interesting and relevant. As we will discuss at the end when we look at the evolution of *AI-Alert* since deploying our new algorithm, the need for human editorial oversight has increased over time.

Ensure the AI Adds Significant Value with Minimal Barriers to Adoption

While NewsFinder automatically acquires and organizes stories for the alert, this automation can become a hindrance if an editor wishes to influence some of these automated processes. For example, to ensure successful adoption of the technology, we found it necessary to have the ability to manually insert one or more news stories. Likewise, we needed a way to remove stories that were selected by the genetic algorithm. With these capabilities, the AI still adds significant value because these kinds of editorial changes require relatively little time, but these capabilities also help remove barriers to adoption.

List the AI's Assumptions or Requirements about the Nature (Format, Size, and Characteristics) of its Inputs and Outputs

AI algorithms and models trained by ML are often highly tuned to the specific nature of the data on which the algorithm was developed and model was trained. In our case, NewsFinder expects to receive candidate news stories from the AITopics database. If these items gradually shift from news stories to blog posts, event announcements, product announcements, tutorials, and so on, then *AI-Alert* would no longer showcase recent news about AI. Another assumption built into NewsFinder is that the stories it examines will be written



Any in Technology (3295)
Information Technology (3295)
Architecture (1)
Artificial Intelligence (3295)
Cognitive Science (36)
Games (2)
Issues (14)
Machine Learning
Bayesian Networks (1)
Computational

Machine Learning: Overviews

"The field of Machine Learning seeks to answer these questions: How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?"
– from The Discipline of Machine Learning by Tom Mitchell. CMU-ML-06-108, 2006.

Best of arXiv.org for AI, Machine Learning, and Deep Learning – December 2018 - insideBIGDATA

#artificialintelligence 2 days ago

Deep reinforcement learning is the combination of reinforcement learning (RL) and deep learning. This field of research has been able to solve a wide range of complex decision-making tasks that were previously out of reach for a machine. Thus, deep RL opens up many new applications in domains such as healthcare, robotics, smart grids, finance, and many more. This paper provides an introduction to deep reinforcement learning models, algorithms and techniques. Particular focus is on the aspects related to generalization and how deep RL can be used for practical applications. The reader is assumed to be familiar with basic machine learning concepts.

deep learning, neural network, survey article, (17 more...)
#artificialintelligence

Industry: Education
Technology:
- Information Technology > Artificial Intelligence > Machine Learning > Reinforcement Learning
- Information Technology > Artificial Intelligence > Machine Learning > Neural Networks > Deep Learning

Figure 9. AITopics Current Website, Using i2k Connect's Technology.

in English. If non-English stories are acquired, various aspects of the genetic algorithm that look at word diversity in the title and summary may break down. Some assumptions are easy to codify. For example, we use a language detector to filter out stories that appear to be written in languages other than English. Other assumptions are harder to define, such as what constitutes a news story as compared with a press release.

Articulate Boundary Conditions on the AI's Inputs and Outputs, and Develop a Plan to Either Ignore or Correct Out-of-bounds and Bogus Inputs and Outputs

We believe it is important to maintain editorial oversight of NewsFinder. As editor-in-chief of AITopics, I review each generated alert and possibly re-generate the alert after adding or removing specific news stories. We have developed easy-to-use functions to perform these editorial tasks.

List All the Ways the AI's Outputs are Used to Automate Some Task, and the Potential Impact Bad Output May Have on that Task, on a User's Experience, and on the Company's Reputation

Because NewsFinder might generate low-quality alerts due to, for example, too many blog posts or press releases in the week's news, an editor ultimately sends the alert to all subscribers. We could trivially automate this step using our mass-mailing provider's application programming interface. However, doing so would open us to significant risk in producing a bad alert. At the price of a few minutes each week for editorial oversight, we are able to maintain quality and continue to provide value to our readers.

Design and Implement a Continuous Evaluation

Once an AI system is deployed to production, its assumptions about the world and learned statistical relationships about its incoming data are tested. But the world can change. Any AI system that processes human-generated content must be carefully watched because people will often find ways to influence the system. Even without assuming intention, changes in the distribution of data might cause an AI system to break down. For example, YouTube has been criticized for showing extremist content next to kid-friendly videos.[3] Just the abundance of conspiracy theory videos on YouTube can also impact how often they are recommended, giving viewers the impression of legitimacy of these fringe topics.[4] AI systems must be maintained like any other system, and AI that continuously learns or reacts to its inputs is especially susceptible to manipulation.

Define performance metrics

These are often defined during system building and may be reused for continuous evaluation. In our case, we can easily identify if AI-Alert is perceived by readers as maintaining its traditional high quality or undergoing a significant decline. We examine the frequency of unsubscribes, open rates, and click rates.

Write Scripts that Automate System Testing According to these Metrics

Create regression tests to ensure that the cases the system solved adequately before are still solved adequately in the future. Ideally, one would develop a system that automatically checks quality metrics and alerts stakeholders if there are any significant changes. We do not have such an automated system for our user engagement metrics, although we do check these metrics each week as we prepare to send the next alert.

Keep Logs of All AI Inputs and Outputs if the Data Size Is Not Unbearable, or Keep Aggregate Statistics if It Is

Define alert conditions to detect degrading performance; for example, to detect whether the AI system is unusually producing the same output repeatedly. Because AI algorithms are often quite sophisticated, it can be difficult to diagnose faults when they occur. The input data can have a large impact on the outcome, so whenever possible, inputs and outputs should be logged to assist this diagnosis. This is not always feasible as data may be large and fast moving. We have found it necessary to keep records of all news stories that were filtered by the algorithm to help the editor make minor adjustments to the selected stories each week.

Consider Asking for Feedback from Users, and Aggregate this Feedback in a Place that Is Often Reviewed

Long-term success of an AI system depends on user satisfaction. If users, or in our case, readers ultimately decide the AI system is not worth the trouble, they will abandon it (in our case, unsubscribe or stop opening or clicking stories). We encourage feedback about *AI-Alert* with a contact link at the bottom of each email. We also benefit from passive data collection about user engagement.

In summary, this four-part checklist helps system designers ensure AI is being used appropriately and successfully. The checklist also helps clarify our goals and design decisions about NewsFinder. With this in mind, we next describe NewsFinder's implementation and the experiment we conducted with the new genetic algorithm for story selection. These sections were documented previously in our Innovative Applications of Artificial Intelligence-deployed application paper, "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*" (Eckroth and Schoen, 2019).

Implementation

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. Once news stories are found and enriched 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 next. An overview of these steps is shown in figure 10.

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. Because web pages can be tricky to parse to find the main body text, we use the snacktory library by Peter Karussell[5] and several site-specific rules to handle edge cases with complex formatting. We use the Darwin library by Jony Hudson[6] for the genetic algorithm. Finally, we use the enlive library by Christophe Grand[7] to generate the alert based on a hypertext markup language (HTML) template.

The alert generator runs as a hypertext transfer protocol service that is activated by a POST request submitted by a weekly cronjob. The alert generator queries the search index (running Apache 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

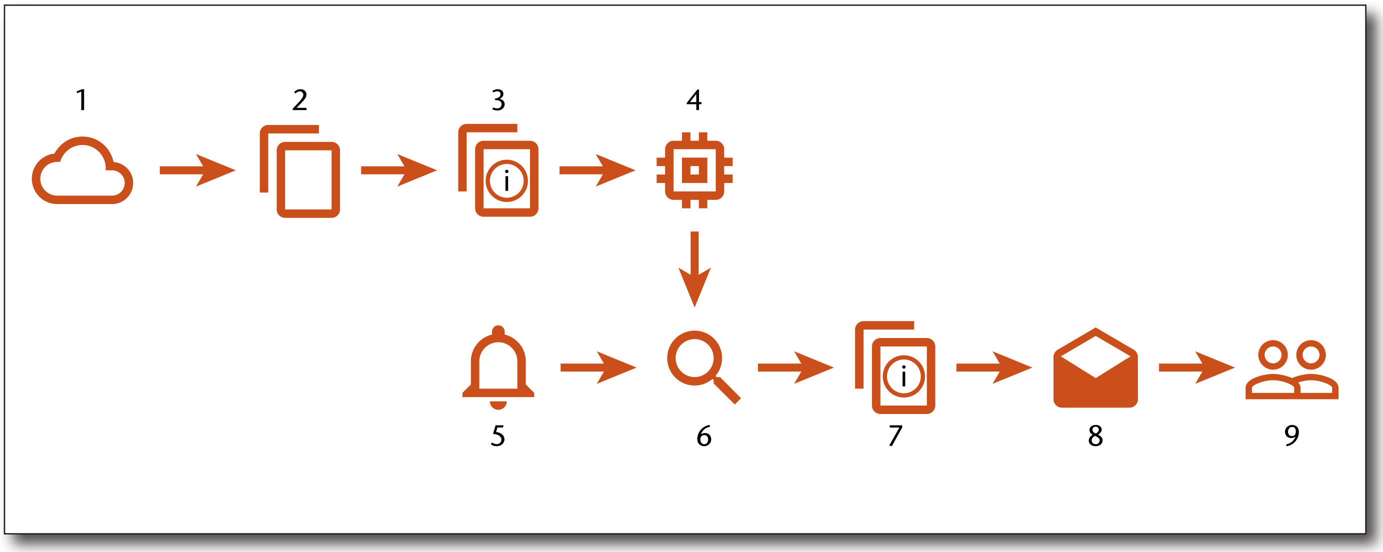


Figure 10. System Design.

The figure shows 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: NewsFinder crawls the web (1) 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 the editors for review (9).

(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. 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.[8] Because 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), although 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 next in more detail.

An Experiment with a Genetic Algorithm

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, that is, 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 twenty-five alerts generated and sent during the six-month experiment, three distinct subsets, each containing a random twenty percent 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 forty percent 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

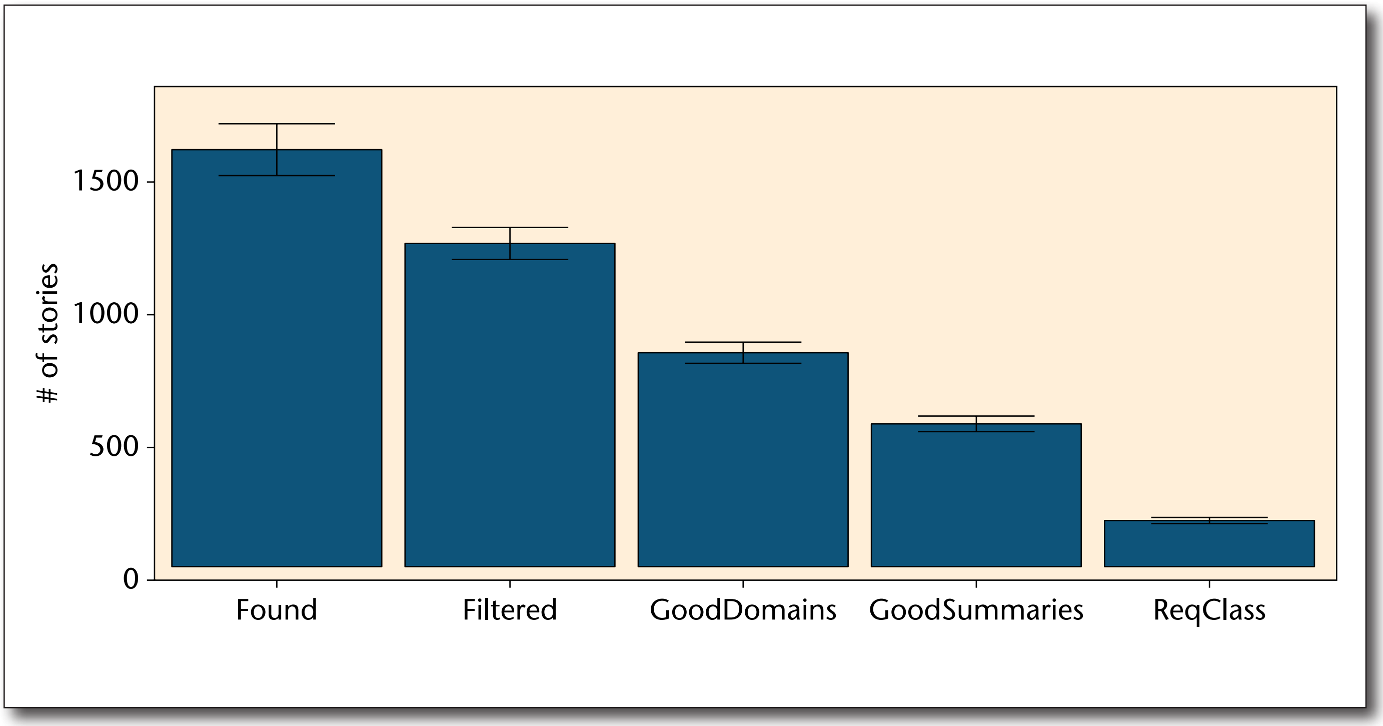


Figure 11. Average Counts of Stories After Each Filter in the Prefiltering Stage.

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 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.

Prefiltering

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 11. Stories are first obtained from the search index by querying on the topic AI from the topic hierarchy. Each week, this yields about 1,500 stories, labeled "Found" in the figure. Next, a set of simple title and web address filters are applied to remove any stories that appear to be blog posts, job postings, press releases, and so on (*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. Because we want stories that have significant AI content, we require that the confidence of at least one AI topic is above eighty percent. Only stories with sufficiently high confidence are retained (*ReqClass*). As shown in the figure, these prefilters reduce the average of 1,500 stories to about 200.

Algorithm 1: Random

Given a set of prefiltered 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

In the first year of our automated *AI-Alert*, stories were selected for the alert according to a scoring algorithm we call "TopClass." The TopClass algorithm scans the prefiltered 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, that is, 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 ten 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 ten 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 ten stories, a random pivot point *p* is selected, between one and nine (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 (nonduplicating) 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), web address 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). To calculate diversity, we first define a function that counts the number of elements in a vector **x** that equal a particular *s:*

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.

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

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 ten have different dates, classes, domains, and/or words in their titles and summaries. The diversity index is defined as:

$$D(\bar{\mathbf{x}}) = 1 - \frac{\sum_{i=1}^{|\bar{x}|} C(\bar{\mathbf{x}}, x_i)\left(1 - C[\bar{\mathbf{x}}, x_i]\right)}{|\bar{\mathbf{x}}|\left(1 - |\hat{x}|\right)} \tag{2}$$

For example, when calculating the diversity of dates, we let $\hat{\mathbf{x}}$ contain 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*(**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 web address domains and scores between 0 and 1 to compute the average domain score for a set of ten stories.

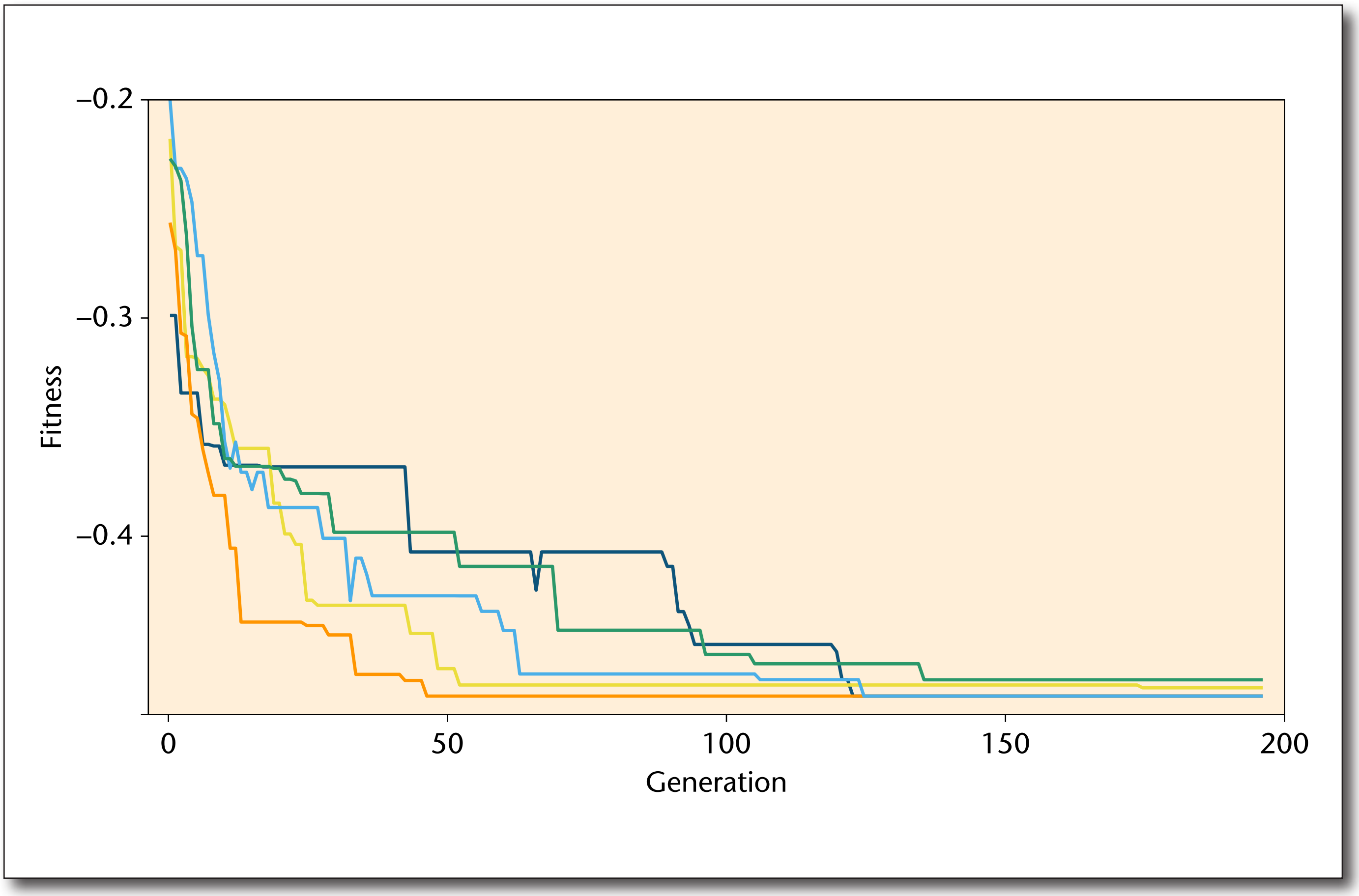


Figure 12. Minimum Fitness Values per Generation for Five Runs with Different Random Seeds.

The fitness function F is defined follows:

$$F = -C_{avg} * D_{avg} * T_{div} * C_{div} * D_{div} * W_{div} * W'_{div} \qquad (3)$$

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

Results

We run the genetic algorithm for 500 generations, but as can be seen in figure 12, about 150 generations suffice. Running 500 generations takes about 1.5 minutes on an Intel Xeon E5 with 96 GB of 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 prefiltered 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 www.wired.com. It is worth noting both stories are about the same event.

The real measure of success is whether 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 13 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

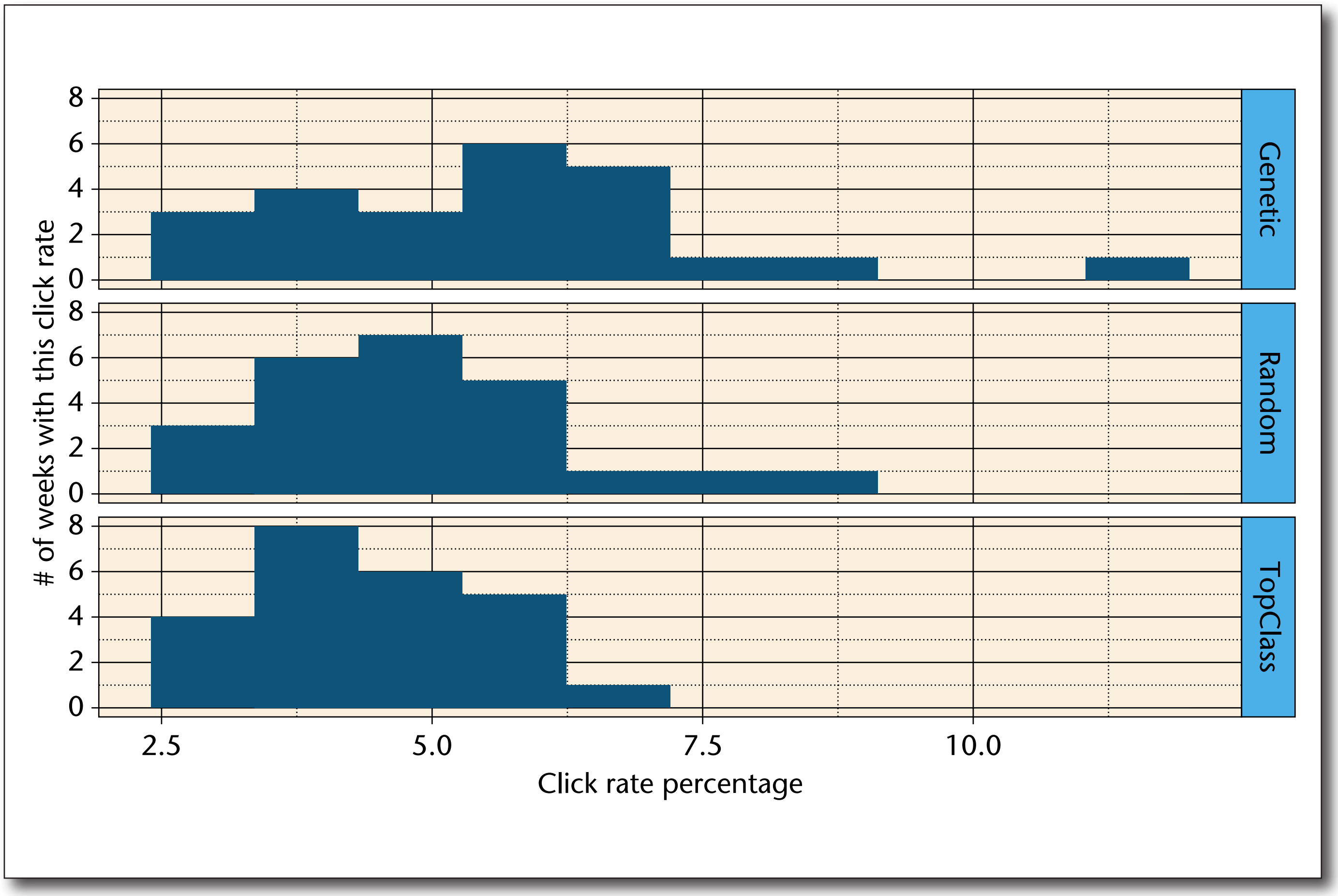


Figure 13. Frequency Plot of the Number of Weeks that Received Various Ranges of Click Rates (Number of Distinct Alerts).

The results from each story selection algorithm are separated as vertical facets.

algorithms. We see that the genetic algorithm has the highest rate in all the statistics (minimum, mean, median). Furthermore, we can calculate statistical significance of these differences in click rates, as shown in table 3. Using pairwise *t* tests, we calculated 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; because there was no information provided to the reader about which version of the alert they received each week, and because 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 percent per week, but this decline was not significant. The unsubscribe rate also declined by about 0.0003 percent 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 prefilters is specific to stories about AI; we expect our approach would work equally well on stories about any subject.

Maintenance

Our six-month experiment successfully validated our belief that the genetic algorithm approach is a

Algorithm	Minimum	First Quartile	Median	Mean	Third Quartile	Maximum
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 (Percent) for Emails Generated by Each Algorithm over the Experimental Period.

Measure	Algorithms 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

The measured variable was click rate (percent).

Table 3. Results of Pairwise t Tests with 23 Degrees of Freedom.

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*. 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 multipurpose suite of technologies that support the alert, the AITopics website, and the Society of Petroleum Engineers' research portal,[9] 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 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 prefiltering stage of processing. These blacklists give bad web address domains and bad words in titles and summaries (not just offensive words, but also words that indicate opinions, rumors, and so forth).

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.

Overall, NewsFinder's code requires little maintenance. However, independent of NewsFinder's particular algorithms, we have seen a gradual evolution of the nature of the alert.

Evolution

Since the end of our experiment, our open rates and click rates have remained mostly constant, as seen in figures 14 and 15. This is a good indicator that readers continue to find value in the alert.

We have seen an interesting change over the years. It seems AI and ML are becoming ever more popular for journalists. But companies are also posting more press releases; bloggers are writing more opinion pieces; and more tutorials, events, and software releases are appearing on social networks. Generally speaking, this is all good for the field. Years ago, AITopics acquired stories only from a select set of news sources such as BBC News, *The New York Times,* and several others. We could generally trust that any story in which our technology found a strong AI aspect was a relevant story to the alert's readers.

However, as the field has grown, we wanted to bring in a broader range of sources. The Twitter hashtag #artificialintelligence is a great source for such broad

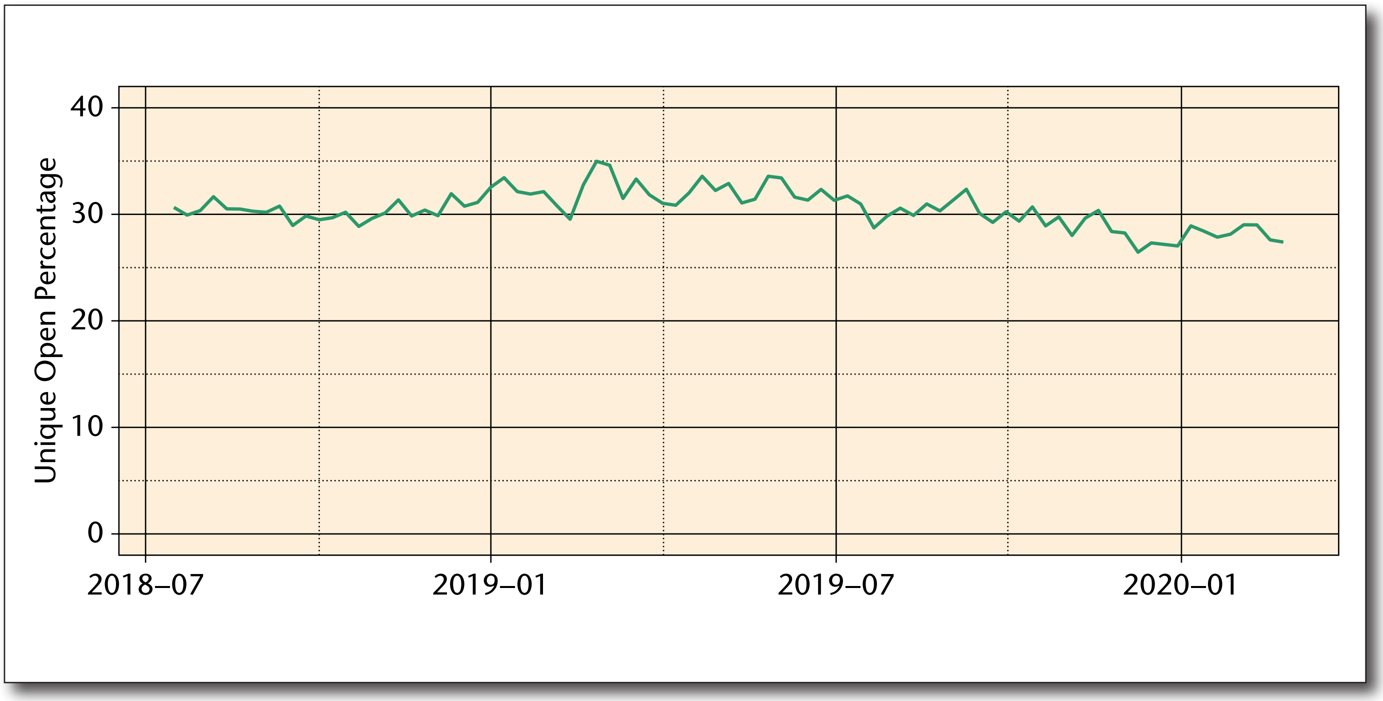


Figure 14. Percent of Delivered Alerts that Are Opened at Least Once by a Reader.

Tracking this metric requires that the reader's email client loads images, which not all email clients do by default. Industry average is 19 percent (MailChimp, 2018).

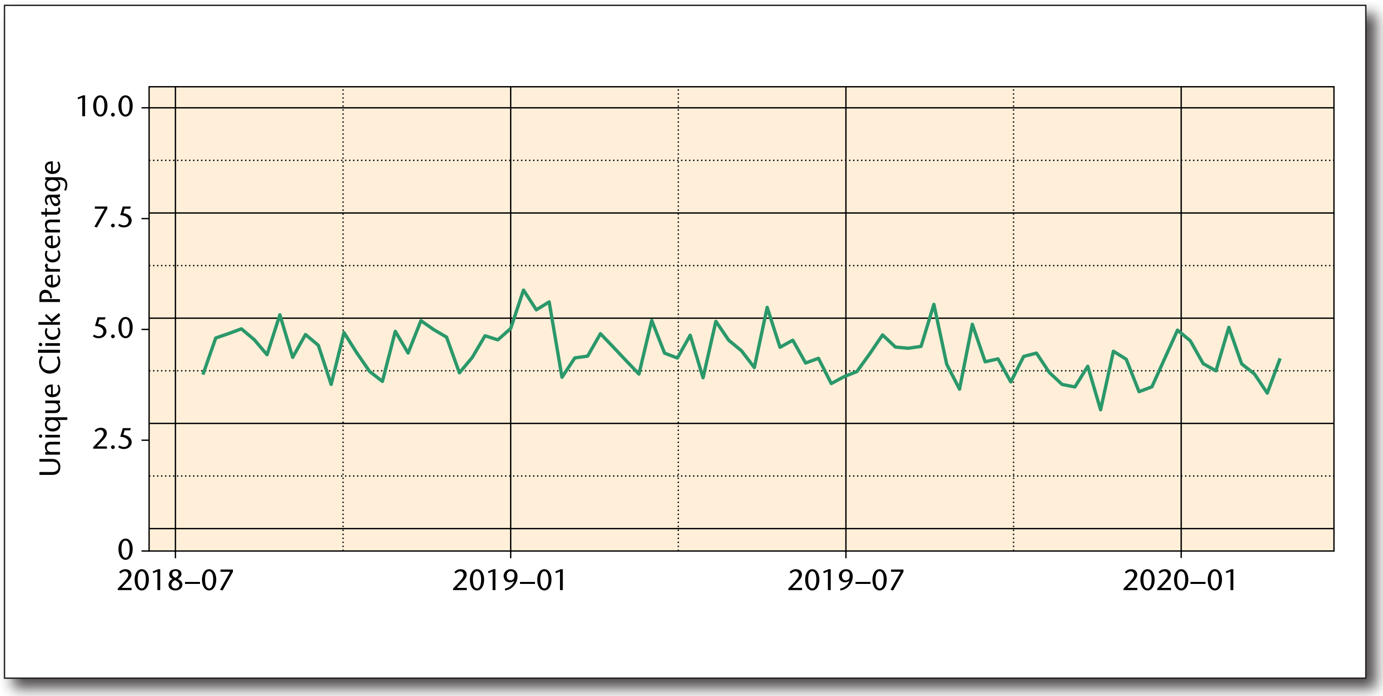


Figure 15. Percent of Readers Who have Clicked at Least One Story or an AITopics Topic Link in a Week's Alert.

Industry average is 2.0 percent (MailChimp, 2018).

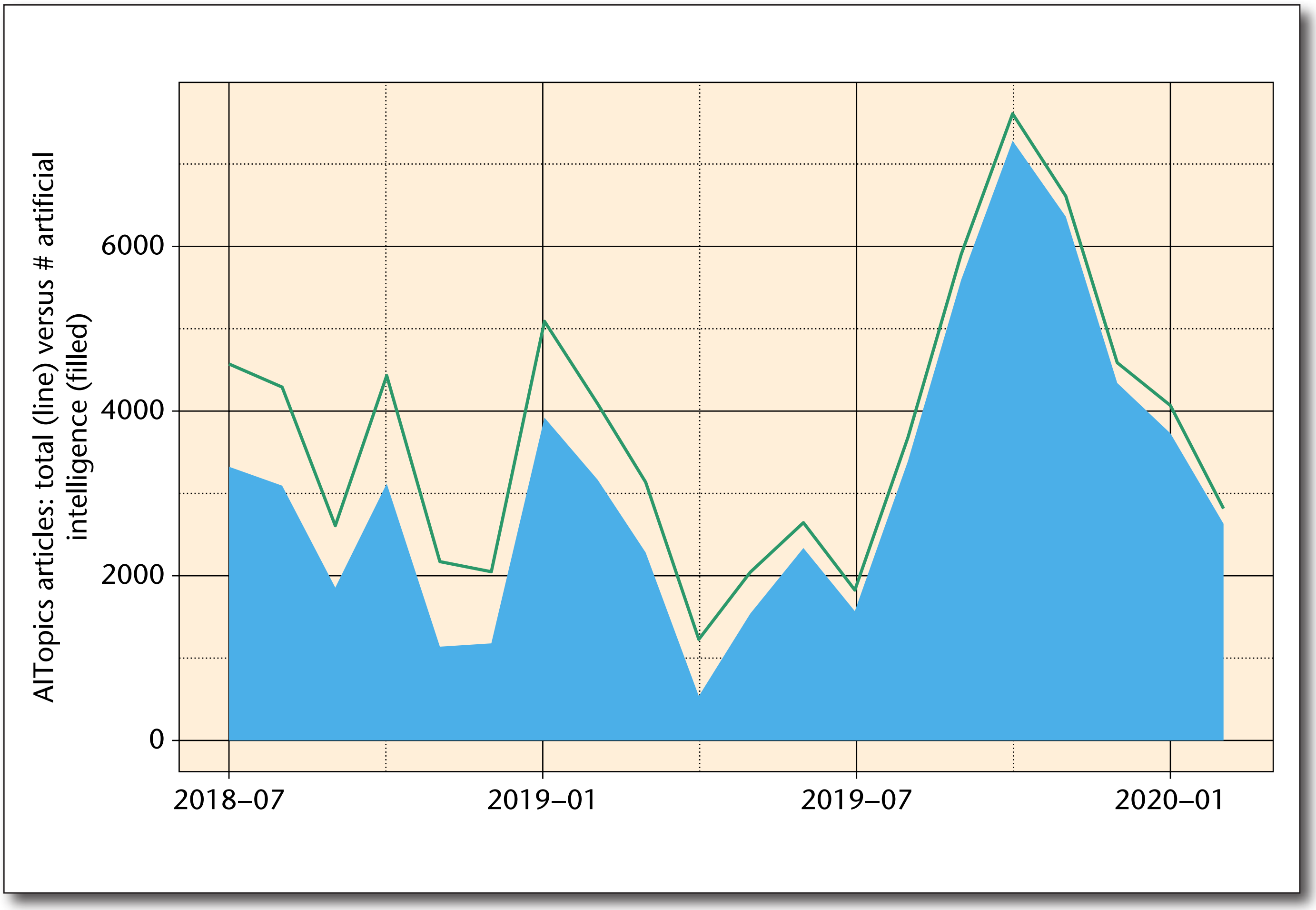


Figure 16. Number of News Stories per Month Acquired on AITopics.

From all news sources, excluding conferences, journals, and preprints (solid line) and stories acquired from #artificialintelligence Twitter hashtag (filled area).

sources. NewsFinder examines any link posted with that hashtag and, if that link has a significant connection with AI according to our document classification engine, it is a candidate for selection by the genetic algorithm.

The popularity of the #artificialintelligence Twitter hashtag has grown so much that now AITopics' news content is strongly dominated by stories posted on Twitter. Figure 16 shows this trend. We emphasize again that overall this is an encouraging outcome. We want an abundance of stories from diverse sources; our document classifiers already ensure the stories injected into AITopics are about the field of AI. Users can visit AITopics and filter content as desired to find the kind of stories they are after. If we continued to only acquire stories from specific hand-selected sources, AITopics would have fewer relevant stories.

The difficulty with the strong influence of #artificialintelligence is that the alert more and more often includes nonnewsworthy content that must be filtered out before the alert is sent. Thus, editorial time has gradually increased since our experiment. Today, the alert must be generated numerous times (between one and five times), and each time the editor filters out stories that are irrelevant (press releases, blog posts, and so forth). These actions add up to about 10 to 20 minutes each week.

It is not clear if there is an AI solution to this problem. As elucidated by the checklist above, we would have to be able to define a specific goal and find a technology that can realize that goal. We would first have to define what kinds of stories are appropriate for the alert. Much of the task can probably be solved with natural language processing techniques such as using word embeddings to identify documents that contain unacceptable words (and their synonyms) such as *tutorial*, or *register* (for events). We would also wish to detect and filter out clickbait stories. Luckily, work has already begun in this area, as indicated by the cleverly-titled AAAI-16 paper, "8 Amazing Secrets for Getting More Clicks: Detecting Clickbaits in News Streams Using

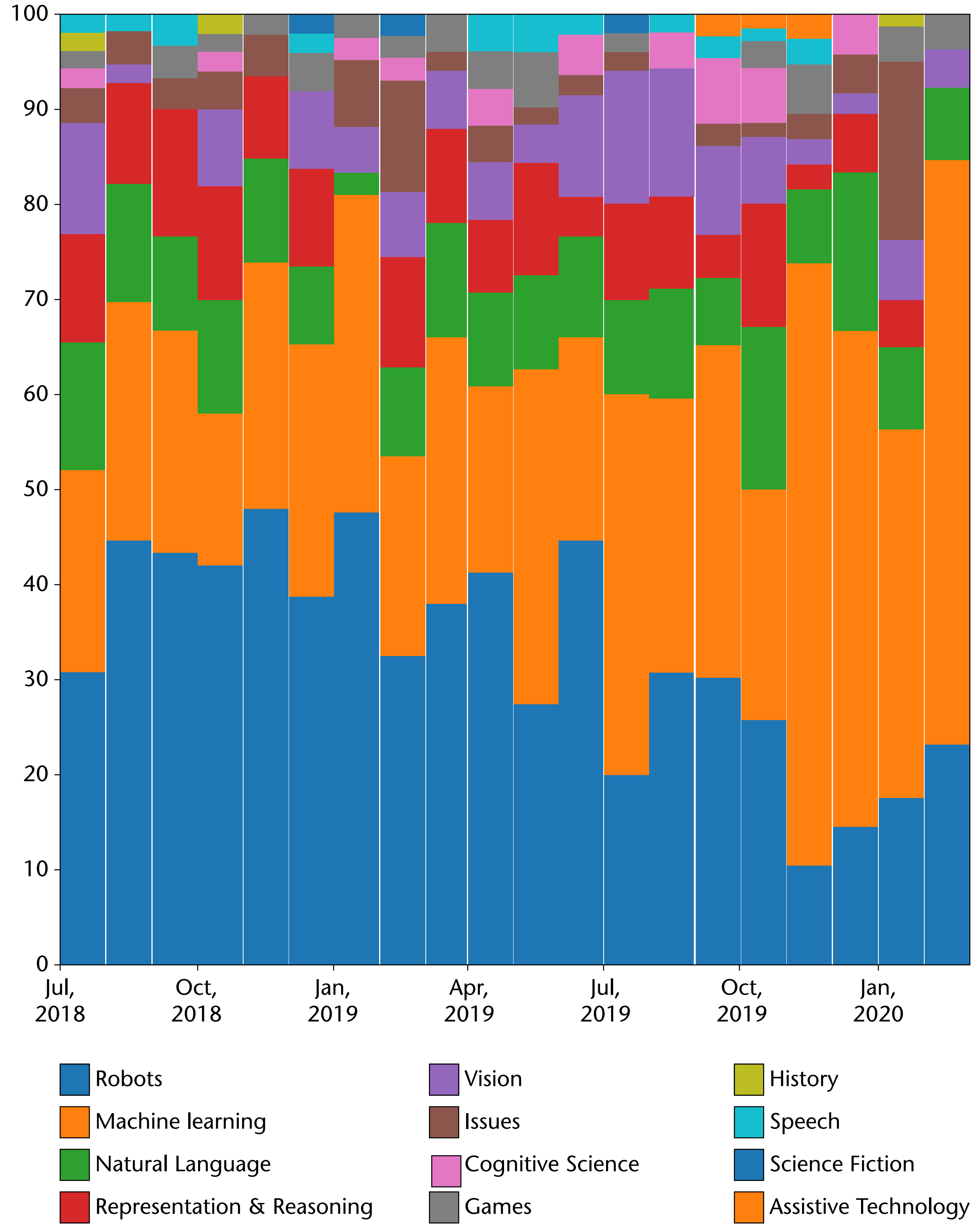


Figure 17. Ratio of Technology Topics Represented by Stories from Each Month of AI-Alert.

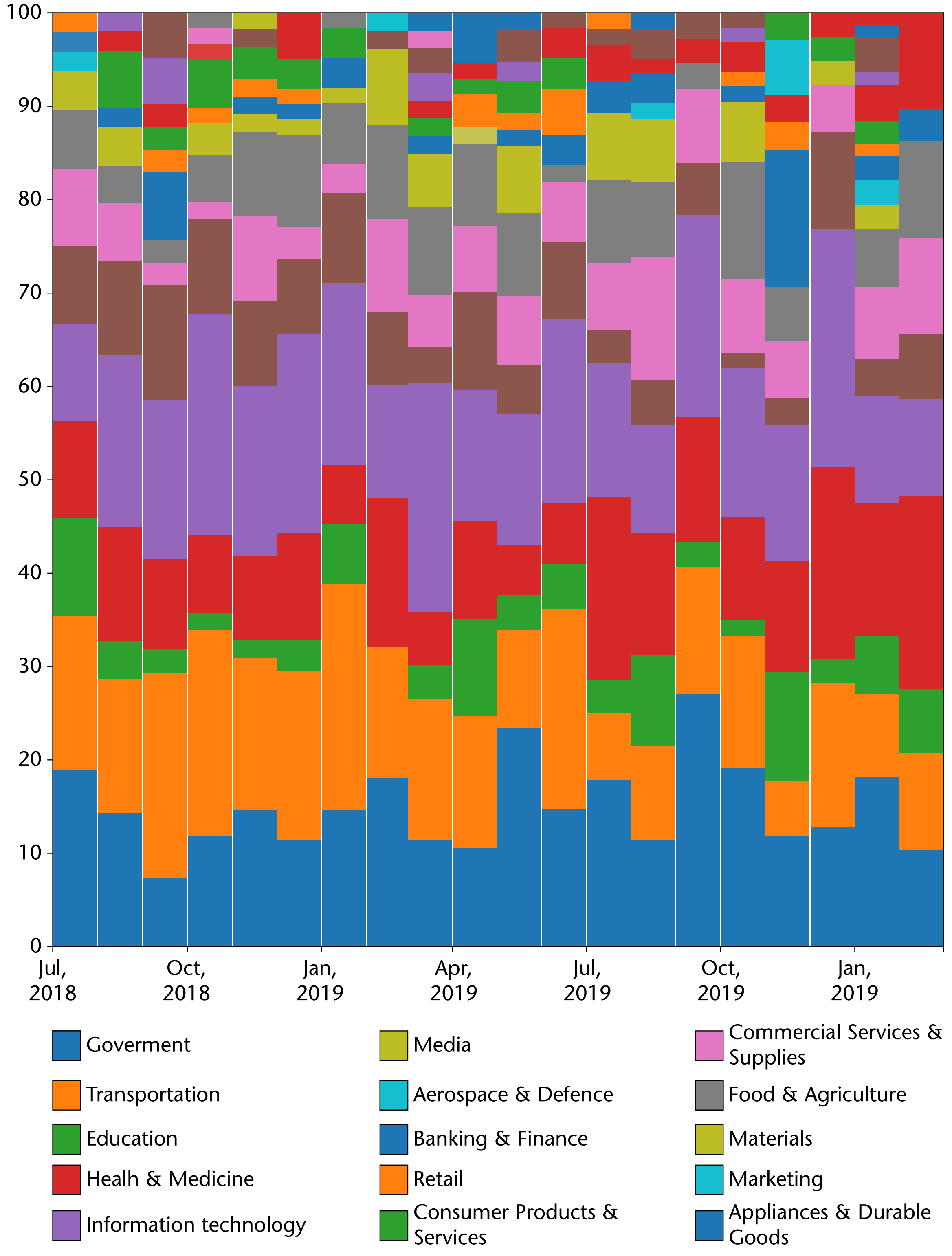


Figure 18. Ratio of Industry Topics Represented by Stories From Each Month of AI-Alert.

Article Informality" (Biyani, Tsioutsiouliklis, and Blackmer, 2016).

Another way of looking at the evolution of the alert is by examining the diversity of topics represented by the stories included in the alerts. All *AI-Alert* stories are archived on AITopics, and AITopics provides interactive topic visualizations. In figure 17 we see how topics in the Technology hierarchy are represented over the weeks of alerts. The ratio of Robotics stories has decreased over time, mostly overtaken by ML stories (typically Deep Learning). Various other topics such as Natural Language Processing continue to be represented. The second important topic hierarchy, Industry, shows greater diversity (figure 18). During our editorial oversight, we eliminate stories that do not appear to be news. However, our editorial actions do not include filtering out or including stories for the sake of influencing the representation of Technology or Industry topics. Thus, the diversity seen in these two visualizations shows the contribution of the genetic algorithm. The algorithm is designed to balance multiple criteria, one of which is topic diversity. These visualizations show it is doing a good job in that respect.

Conclusion

AI-Alert has evolved over twenty years to become a nearly fully automated weekly email. The alert contains the top stories about AI each week from myriad sources. Over recent years, the ways that journalists and other writers produce content about the field of AI has changed. Today, there are many stories produced each day, as well as tutorials, blog posts, software releases, and more that are not good candidates for inclusion in the alert. We invented a genetic algorithm to generate the alert each week from more than a thousand candidate stories, and demonstrated its quality through six months of experimentation. In this way, we are using AI to generate alerts about what's happening in the field of AI; and all of this work is sponsored by the premier professional society for AI. This article documented our history, the design and implementation of the automated system, and lessons learned from deploying this system in production. We expect *AI-Alert* and its parent website and document database, AITopics, to continue to evolve over the next twenty years as the field of AI and ML continues to grow and mature.

Acknowledgments

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

Notes

1. aitopics.org/class/AI-Alerts
2. github.com/JonyEpsilon/darwin
3. www.theguardian.com/technology/2017/mar/25/google-youtube-advertising-extremist-content-att-verizon
4. www.nytimes.com/interactive/2020/03/02/technology/youtube-conspiracy-theory.html
5. github.com/karussell/snacktory
6. github.com/JonyEpsilon/darwin
7. github.com/cgrand/enlive
8. twitter.com/aitopics
9. search.spe.org/i2kweb/SPE/search

References

Biyani, P.; Tsioutsiouliklis, K.; and Blackmer, J. 2016. 8 Amazing Secrets for Getting More Clicks: Detecting Clickbaits in News Streams Using Article Informality. In *Proceedings of the Thirtieth Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, 94–100. Palo Alto, CA: Advancement of Artificial Intelligence (AAAI) Press.

Buchanan, B. G., and Glick, J. 2002. AI Topics: A Responsibility to Celebrate AI Responsibly. *AI Magazine* 23(1): 87–94.

Dong, L.; Smith, R. G.; and Buchanan, B. G. 2011. Newsfinder: Automating an Artificial Intelligence News Service. In *Proceedings of the Twenty-Third Innovative Applications of Artificial Intelligence Conference*, 1581–8. Palo Alto, CA: Advancement of Artificial Intelligence (AAAI) Press.

Eckroth, J. 2018. *AI Blueprints: How to Build and Deploy AI Business Projects*. Birmingham, UK: Packt Publishing Ltd.

Eckroth, J.; Dong, L.; Smith, R. G.; and Buchanan, B. G. 2012. NewsFinder: Automating an AI News Service. *AI Magazine* 33(2): 43–54. doi.org/10.1609/aimag.v33i2.2406.

Eckroth, J., and Schoen, E. 2019. 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. In *Thirty-First Association for the Advancement of Artificial Intelligence (AAAI) Conference on Innovative Applications of Artificial Intelligence*, 9357–64. Palo Alto, CA: Advancement of Artificial Intelligence (AAAI) Press. doi.org/10.1609/aaai.v33i01.33019357.

Jost, L. 2006. Entropy and diversity. *Oikos* 113(2): 363–75. doi.org/10.1111/j.2006.0030-1299.14714.x.

Luhn, H. P. 1958. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development* 2(2): 159–65. doi.org/10.1147/rd.22.0159.

MailChimp. 2018. *Average Email Campaign Stats of MailChimp Customers by Industry*. Atlanta, GA: The Rocket Science Group, LLS. mailchimp.com/resources/research/email-marketing-benchmarks.

Sculley, D.; Holt, G.; Golovin, D.; Davydov, E.; Phillips, T.; Ebner, D.; Chaudhary, V.; and Young, M. 2014. Machine Learning: The High Interest Credit Card of Technical Debt. Paper presented at Software Engineering for Machine Learning (SEAML) (Advances in Neural Information Processing [NIPS] 2014 Workshop), Montreal, Quebec, Canada, December 8–13.

Smith, R. G., and Eckroth, J. 2017. Building AI Applications: Yesterday, Today, and Tomorrow. *AI Magazine* 38(1): 6–22. doi.org/10.1609/aimag.v38i1.2709.

Joshua Eckroth is an associate professor at Stetson University, the chief architect at i2k Connect, and editor-in-chief of AITopics. He holds a PhD in Computer Science from the Ohio State University.