Agile Machine Learning

Effective Machine Learning Inspired by the Agile Manifesto

—

Eric Carter
Matthew Hurst

Apress®

Agile Machine Learning

Effective Machine Learning Inspired by the Agile Manifesto

Eric Carter

Matthew Hurst

Apress®

Agile Machine Learning: Effective Machine Learning Inspired by the Agile Manifesto

Eric Carter
Kirkland, WA, USA

Matthew Hurst
Seattle, WA, USA

ISBN-13 (pbk): 978-1-4842-5106-5
ISBN-13 (electronic): 978-1-4842-5107-2
https://doi.org/10.1007/978-1-4842-5107-2

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Joan Murray
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484251065. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

Matthew would like to dedicate this book to his family

—Wakako and Hana.

Eric would like to dedicate this book to his wife Tamsyn.

Table of Contents

About the Authors

Eric Carter has worked as a Partner Group Engineering Manager on the Bing and Cortana teams at Microsoft. In these roles he worked on search features around products and reviews, business listings, email, and calendar. He currently works on the Microsoft Whiteboard product.

Matthew Hurst is a Principal Engineering Manager and Applied Scientist currently working in the Machine Teaching group at Microsoft. He has worked in a number of teams in Microsoft including Bing Document Understanding, Local Search and in various innovation teams.

About the Technical Reviewer

James McCaffrey works for Microsoft Research in Redmond, Washington. James has a PhD in computational statistics and cognitive psychology from the University of Southern California, a BA in experimental psychology from the University of California at Irvine, a BA in applied mathematics, and an MS in computer science. James is also the Senior Technical Editor for Microsoft MSDN Magazine, one of the most widely-read technical journals in the world. James learned to speak to the public when he worked at Disneyland during his college days, and he can still recite the Jungle Cruise ride narration from memory. His personal blog site is `https://jamesmccaffrey.wordpress.com`.

Introduction

This book was born out of a fortuitous meeting. In July of 2012, Eric Carter had just returned to the U.S. following a three-year assignment in Germany launching a shopping search product for Microsoft to the European market. He was sorely disappointed because the effort he had led in Europe was shutting down and so began looking for a new gig. While exploring opportunities in Bing, Microsoft's search engine, he met Matthew Hurst. Matthew had joined Microsoft as a member of Live Labs, an innovation group tasked with exploring novel solutions and applications around search, the cloud and connected technologies. From there he'd worked on various incarnations of maps and local search, often on features connecting text with location. What followed was a complementary partnership that vastly improved the quality of Bing's local search and ultimately led both on a learning journey of how data engineering projects benefit from the application of *Agile principles*.

The Agile Manifesto (or, *The Manifesto for Agile Software Development*, to give it its full title) came into being in 2001 as a collaboration of the seventeen signatories[1]. It is summarized as four values (*individuals and interactions* are valued over process and tools, *working software* over comprehensive documentation, *customer collaboration* over contract negotiations, and *responding to change* over following a plan) and twelve principles. In this book, we examine each of the principles in turn, and relate them to our experiences in working with data and inference methods in a number of projects and contexts.

When the authors met, Bing's local search product was very much a work-in-progress. The quality of the catalog of local businesses was improving, but it was still far behind the Google, market leader at the time. Matthew was on the local search data team and he, along with other members of the team, had been exploring some innovative ideas to better leverage the web and integrate machine learning to dramatically improve the catalog. Eric saw a number of compelling challenges in the local search space as it existed in Bing, and decided to join as the engineering manager of Bing's local data team.

[1]Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas.

At this point in his career, Eric was no stranger to managing teams at Microsoft, having been part of several Visual Studio related products and the now dismantled "shopping search" project. However, it was during his time working with Visual Studio that he discovered the intrinsic value of Agile, and how much more efficient and happy teams were when following Agile principles. Wanting to bring that to his new team, he found himself in a quandary— how do you apply Agile to a team that is more about producing data than producing software? What would it take to bring Agile to a data engineering team?

It wasn't easy. At first, Agile seemed like an invading foreign agent. The team culture was about big ideas discovered through experimentation, long horizon research, and a lot of trial and error science projects—all seemingly contrarian to Agile principles like scrum, iterative development, predictability, simplicity, and delivering working software frequently. With a team focused on producing an exceedingly accurate database of all the businesses in the world, defining "done" was nothing short of impossible. After all, the singular constant in data is that it contains errors—the work is literally never done. Faced with challenges such as communicating to stakeholders how and where the team was making progress, determining whether a particular development investment was worth making, and ensuring that improvements are delivered at a regular but sustainable pace, it became apparent that a modern Agile approach was critical. But how does one apply Agile in a team comprised of data scientists and traditional engineers all working on data-oriented deliverable?

Traditional Agile processes were intended to reduce unknowns and answer questions such as "what does the customer want" and "how can software be delivered reliably and continuously". But in this new project, new world, we already knew what the customer wanted (a perfect catalog of local businesses[2]) but we needed to answer questions such as "What's in the data?" and "What are we capable of delivering based on that data?" We needed Agile approaches, but revised for a modern, mixed talent, data engineering team.

As we navigated through "next-generation" machine learning challenges, we discovered that, without question, Agile principles can be applied to solve problems and reduce uncertainty about the data, making for a much happier and efficient team.

[2]As we will later see in Chapter 3, what the customer wanted wasn't quite as simple as I thought.

Our hope in bringing together this modernized version of Agile methodologies is that the proven guidance and hard earned insights found in this book will help individuals, technical leads and managers be more productive in the exciting work that is happening in machine learning and big data today.

June 2019

Eric Carter

Matthew Hurst

CHAPTER 1

Early Delivery

> ***Our*** *highest priority is to* ***satisfy the customer*** *through* ***early*** *and* ***continuous*** *delivery of* ***valuable [data]****.*
>
> *—agilemanifesto.org/principles*

Data projects, unlike traditional software engineering projects, are almost entirely governed by a resource fraught with unknown patterns, distributions, and biases – the data. To successfully execute a project that delivers value through inference over data sets, a novel set of skills, processes, and best practices need to be adopted. In this chapter, we look at the initial stages of a project and how we can make meaningful progress through these unknowns while engaging the customer and continuously improving our understanding of the data, its value, and the implications it holds for system design.

To get started, let's take a look at a scenario that introduces the early stages of a project that involved mining local business data from the Web which comes from the authors' experience working on Microsoft's Bing search engine. There are millions of local business locations in the United States. Approximately 50%[1] of these maintain some form of web site whether in the form of a simple, one-page design hosted by a drag-and-drop web hosting service or a sophisticated multi-brand site developed and maintained by a dedicated web team. The majority of these businesses update their sites before any other representation of the business data, driven by an economic incentive to ensure that their customers can find authoritative information about them.[2] For example, if their phone number is incorrect, then potential customers will not be able to reach them; if they move and their address is not updated, then they risk losing existing clients; if their business hours change with the seasons, then customers may turn away.

[1]Based on analysis of local business feed data.

[2]A survey of businesses showed that about 70% updated their web sites first and then other channels such as social media or Search Engine Optimization (SEO) representatives.

E. Carter and M. Hurst, *Agile Machine Learning*, https://doi.org/10.1007/978-1-4842-5107-2_1

A local search engine is only as good as its data. Inaccurate or missing data cannot be improved by a pretty interface. Our team wanted to go to the source of the data – the business web site – to get direct access to the authority on the business. As an aggregator, we wanted our data to be as good as the data the business itself presented on the Web. In addition, we wanted a machine-oriented strategy that could compete with the high-scale, crowd-sourced methods that competitors benefitted from. Our vision was to build an entity extraction system that could ingest web sites and produce structured information describing the businesses presented on the sites.

Extraction projects like this require a schema and some notion of quality to deliver a viable product,[3] both determined by the customer – that is, the main consumer of the data. Our goal was to provide additional data to an existing system which already ingested several feeds of local data, combining them to produce a final conflated output. With an existing product in place, the schema was predetermined, and the quality of the current data was a natural lower bound on the required quality. The schema included core attributes: business name, address, phone number, as well as extended attributes including business hours, latitude and longitude, menu (for restaurants), and so on. Quality was determined in terms of errors in these fields.

The Metric Is the Customer The first big shift in going from traditional agile software projects to data projects is that much of the role of the customer is shifted to the metric measured for the data. The customer, or product owner, certainly sets things rolling and works in collaboration with the development team to establish and agree to the evaluation process. The evaluation process acts as an oracle for the development team to guide investments and demonstrate progress.

Just as the customer-facing metric is used to guide the project and communicate progress, any component being developed by the team can be driven by metrics. Establishing internal metrics provides an efficient way for teams to iterate on the

[3]Be extremely wary of projects that haven't, won't, or can't define a desired output. If you find yourself in the vicinity of such a project, run away – or at least make it the first priority to determine exactly what the project is supposed to produce.

inner loop[4] (generally not observed by the customer). An inner metric will guide some area of work that is intended to contribute to progress of the customer-facing metric.

A metric requires a data set in an agreed-upon schema derived from a sampling process over the target input population, an evaluation function (that takes data instances and produces some form of score), and an aggregation function (taking all of the instance results and producing some overall score). Each of these components is discussed and agreed upon by the stakeholders. Note that you will want to distinguish metrics of quality (i.e., how correct is the data) from metrics of impact or value (i.e., what is the benefit to the product that is using the data). You can produce plenty of high-quality data, but if it is not in some way an improvement on an existing approach, then it may not have any actual impact.

Getting Started

We began as a small team of two (armed with some solid data engineering skills) with one simple goal – to drive out as many unknowns and assumptions as possible in the shortest amount of time. To this end, we maximized the use of existing components to get data flowing end to end as quickly as possible.

Inference We use the term "inference" to describe any sort of data transformation that goes beyond simple data manipulation and requires some form of conceptual modeling and reasoning, including the following techniques:

- Classification: Determining into which bucket to place a piece of data
- Extraction: Recognizing and normalizing information present in a document
- Regression: Predicting a scalar value from a set of inputs
- Logical reasoning: Deriving new information based on existing data rules

[4]An "inner loop" is the high-frequency cycle of work that developers carry out when iterating on a task, bringing it to completion. It is a metaphorical reference to the innermost loop in a block of iterative code.

Structural transformations of data (e.g., joining tables in a database) are not included as inference, though they may be necessary components of the systems that we describe.

Adopting a strategy of *finding* technology rather than *inventing* technology allowed us to build something in a matter of days that would quickly determine the potential of the approach, as well as identify where critical investments were needed. We quickly learned that studying the design of an existing system is a valuable investment in learning how to build the next version.

But, before writing a single line of code, we needed to look at the data. Reviewing uniform sample of web sites associated with businesses, we discovered the following:

- Most business web sites are small, with ten pages or less.
- Most sites used static web content – that is to say, all the information is present in the HTML data rather than being dynamically fetched and rendered at the moment the visitor arrives at their site.
- Sites often have a page with contact information on it, though it is common for this information to be present in some form on many pages, and occasionally there is no single page which includes all desired information.
- Many businesses have a related page on social platforms (Facebook, Instagram, Twitter), and a minority of them only have a social presence.

These valuable insights, which took a day to derive, allowed us to make quick, broad decisions regarding our initial implementation. In hindsight, we recognized some important oversights, such as a distinction between (large) chain businesses and the smaller "singleton" businesses. From a search perspective, chain data is of higher value. While chains represent a minority of actual businesses, they are perhaps the most important data segment because users tend to search for chain businesses the most. Chains tend to have more sophisticated sites, often requiring more sophisticated extraction technology. Extracting an address from plain HTML is far easier than extracting a set of entities dynamically placed on a map as the result of a zip code search.

Every Task Starts with Data Developers can gain insights into the fundamentals of a domain by looking at a small sample of data (less than 100). If there is some aspect of the data that dominates the space, it is generally easy to identify. Best practices for reviewing data include randomizing your data (this helps to remove bias from your observations) and viewing the data in as native a form as possible (ideally seeing data in a form equivalent to how the machinery will view it; viewing should not transform it). To the extent possible, ensure that you are looking at data from production[5] (this is the only way to ensure that you have the chance to see issues in your end-to-end pipeline).

With our early, broad understanding of the data, we rapidly began building out an initial system. We took a pragmatic approach to architecture and infrastructure. We used existing infrastructure that was built for a large number of different information processing pipelines, and we adopted a simple sequential pipeline architecture that allowed us to build a small number of stages connected by a simple data schema. Specifically, we used Bing's cloud computation platform which is designed to run scripted processes that follow the MapReduce pattern on large quantities of data. We made no assumptions that the problem could best be delivered with this generic architecture, or that the infrastructure and the paradigms of computation that it supported were perfectly adapted to the problem space. The only requirement was that it was available and capable of running some processes at reasonable scale and would allow developers to iterate rapidly for the initial phase of the project.

Bias to Action In general, taking some action (reviewing data, building a prototype, running an experiment) will always produce some information that is useful for the team to make progress. This contrasts with debating options, being intuitive about data, assuming that something is "obvious" about a data set, and so on. This principle, however, cannot be applied recklessly – the action itself must be well defined with a clear termination point and ideally a statement of how the product will be used to move things forward.

[5] "Production" refers to your production environment – where your product is running and generating and processing data.

If we think about the components needed for web mining local business sites to extract high-quality records representing the name, address, and phone number of these entities, we would need the following:

- A means of discovering the web sites
- A crawler to crawl these sites
- Extractors to locate the names, addresses, and phone numbers on these sites
- Logic to assemble these extracted elements into a record, or records for the site
- A pipeline implemented on some production infrastructure that can execute these components and integrate the results

Each of these five elements would require design iterations, testing, and so on. In taking an agile discovery approach to building the initial system, we instead addressed the preceding five elements with these five solutions:

- Used an existing corpus of business records with web sites to come up with an initial list of sites to extract from – no need to build a discovery system yet
- Removed the need to crawl data by processing data only found in the production web corpus of the web search engine already running on the infrastructure we were to adopt for the first phase of work
- Used an existing address extractor and built a simple phone number extractor and name extractor
- Implemented a naïve approach to assemble extracted elements into a record which we called “entification”
- Deployed the annotators and entification components using an existing script-based execution engine available to the larger Bing engineering team that had access to and was designed to scale over data in the web corpus

We discovered as part of this work that there were no existing off-the-shelf approaches to extracting the business name from web sites. This, then, became an initial focus for innovation. Later in the project, there were opportunities to improve other parts of the system, but the most important initial investment to make was name extraction.

By quickly focusing on critical areas of innovation and leveraging existing systems and naïve approaches elsewhere, we delivered the first version of the data in a very short time frame. This provided the team with a data set that offered a deeper understanding of the viability of the project. This data set could be read to understand the gap between our naïve architecture and commodity extractors and those that were required to deliver to the requirements of the overall project. Early on, we were able to think more deeply about the relationship between the type of data found in the wild and the specializations required in architecture and infrastructure to ensure a quality production system.

Going End to End with Off-the-Shelf Components Look for existing pieces that can approximate a system so that you can assess the viability of the project, gain insight into the architecture required, recognize components that will need prioritized development, and discover where additional evaluation might be required for the inner loop. In large organizations, likely many of the components you need to assemble are already available. There are also many open source resources that can be used for both infrastructure and inference components.

Getting up and running quickly is something of a recursive strategy. The team makes high-level decisions about infrastructure choices (you want to avoid any long-term commitments to expensive infrastructure prior to validating the project) and architecture (the architecture will be iterated on and informed by the data and inference ecosystem). This allows the team to discover where innovation is required, at which point the process recurses.

To build the name extractor, we enlisted a distance learning approach. We had available to us a corpus of data with local entity records associated with web site URLs. To train a model to extract names from web sites, we used this data to automatically label web pages. Our approach was to

1. Randomly create a training set from the entire population of business-site pairs.
2. Crawl the first tier of pages for each URL.

3. Generate n-grams from the pages using a reasonable heuristic (e.g., any strings of 1–5 tokens found within an HTML element).
4. Label each n-gram as positive if they match the existing name of the business from the record and negative otherwise – some flexibility was required in this match.
5. Train a classifier to accept the positive cases and reject the negative cases.

If we think about the general approach to creating a classifier, this method allows for the rapid creation of training data while avoiding the cost of manual labeling, for example, creating and deploying tools, sourcing judges, creating labeling guidelines, and so on.

Important Caveat to Commodity System Assembly You may state at some point that the code you are writing or the system you are designing or the platform you are adopting is just a temporary measure and that once you have the "lay of the land," you will redesign, or throw experimental code away, and build the real system. To throw experimental code away requires a highly disciplined team and good upward management skills. Once running, there is pressure to deliver to the customer in a way that starts building dependencies not only on the output but on the team's capacity to deliver more. Often this is at the cost of going back and addressing the "temporary" components and designs that you used to get end to end quickly.

Data Analysis for Planning

Now that we have a system in place, it's time to look at the initial output and get a sense of where we are. There is a lot of value in having formal, managed systems that pipe data through judges and deliver training data, or evaluation data, but that should never be done prior to (or instead of) the development team getting intimate with every aspect of the data (both input and output). To do so would miss some of the best opportunities for the team to become familiar with the nuances of the domain.

There are two ways in which a data product can be viewed. The first looks at the *precision* of the data and answers the question "How good are the records coming out of the system?" This is a matter of sampling the output, manually comparing it to the input, and determining what the output should have been – did the machine get it right? The second approach focuses on the *recall* of the system – for all the input where we should have extracted something, how often did we do so and get it right?

The insight that we gained from reviewing the initial output validated the project and provided a backlog of work that led to our decision to expand the team significantly:

- Overall precision was promising but not at the level required.
- Name precision was lower than expected, and we needed to determine if the approach was right (and more features were needed) or if the approach was fundamentally flawed.
- The commodity address extractor was overly biased to precision, and so we missed far too many businesses because we failed to find the address.
- Our understanding of the domain was naïve, and through the exposure to the data, we now had a far deeper appreciation of the complexity of the world we needed to model. In particular, we started to build a new schema for the domain that included the notion of singleton businesses, business groups, and chains, as well as simple and complex businesses and sites. These concepts had a strong impact on our architecture and crawling requirements.
- The existing web index we leveraged to get going quickly was not sufficient for our scenario – it lacked coverage, but also failed to accurately capture the view of certain types of pages as a visitor to the site would experience them.

Now that we had a roughed-in system, we could use it to run some additional exploratory investigations to get an understanding of the broader data landscape that the project had opened up. We ran the system on the Web at large (rather than the subset of the Web associated with our existing corpus of business data records). More about this later.

Data Evaluation Best Practices It is common for data engineering teams early on in their career to use ad hoc tools to view and judge data. For example, you might review the output of a component in plain text or in a spreadsheet program like Excel. It soon becomes apparent, however, that even for small data sets, efficiency can be gained when specialized tools that remove as much friction as possible for the workflows involved in data analysis are developed.

Consider reviewing the output of a process extracting data from the Web. If you viewed this output in Excel, you would have to copy the URL for the page and paste it into a browser, and then you would be able to compare the extraction with the original page. The act of copying and pasting can easily be the highest cost in the process. When the activities required to get data in front of you are more expensive than viewing the data itself, the team should consider building (or acquiring) a tool to remove this inefficiency. Teams are distinguished by their acknowledgement of the central importance of high-quality data productivity tools.

Another consideration is the activity of the evaluation process itself. Generally, there are judgment-based processes (a judge – that is to say, a human – will look at the output and make a decision on its correctness based on some documented guidelines) and data-based processes (a data set – often called a **ground truth set** – is constructed and can be used to fully automate the evaluation of a process output). Ground truth-based processes can be incredibly performant, allowing the dev team to essentially continuously evaluate at no cost.

Both tool development and ground truth data development involve real costs. It is a false economy to avoid these investments and attempt to get by with poorly developed tools and iteration loops that require manual evaluation.

Establishing Value

With the basics of the system in place, we next determined how to ship the data and measure its value, or impact. The data being extracted from the Web was intended for use in our larger local search system. This system in part created a corpus of local entities by ingesting around 100 feeds of data and merging the results through conflation/

merge (or record linkage as it is often termed). The conflation/merge system had its own machine learning models for determining which data to select at conflation/merge time, and so we wanted to do everything we could to ensure our web-extracted data got selected by this downstream system. We got an idea of the impact of our system by both measuring the quality of the data and how often our data got selected by the downstream system. The more attributes from our web-mined data stream were selected and surfaced to the user, the more impact it had. By establishing this downstream notion of impact, we could chart a path to continuous delivery of increasing impact – the idea being to have the web-mined data account for more and more of the selected content shown to users of the local search feature in Bing.

We considered several factors when establishing a quality bar for the data we were producing. First was the quality of existing data. Surprisingly, taking a random sample of business records from many of the broad-coverage feeds (i.e., those that cover many types of businesses vs. a vertical feed that covers specific types of businesses like restaurants) showed that the data from the broad-coverage feeds was generally substandard. We could have made this low bar our requirement for shipping. However, vertical and other specialized feeds tended to be of higher quality – around 93% precision per attribute. It made sense, then, to use these "boutique" vertical data sources as our benchmark for quality.

In addition, we considered the quality of the data in terms of how well that quality can be estimated by measurement. Many factors determine the accuracy of a measurement, with the most important being sample design and size and human error. These relate to the expense of the judgment (the bigger the sample, the more it costs to judge, the lower the desired error rate, the more judges one generally requires per Human Intelligence Task (HIT)[6]). Taking all this into account, we determined that a per attribute precision of 98% was the limit of practically measurable quality. In other words, we aimed to ship our data when it was of similar quality to the boutique feeds and set a general target of 98% precision for the name, address, and phone number fields in our feed.

[6]A Human Intelligence Task (HIT) is a unit of work requiring a human to make a judgment about a piece of data.

Because we had established a simple measurement of impact – the percentage of attributes selected from this feed by the downstream system – our baseline impact was 0%. We therefore set about analyzing the data to determine if there was a simple way to meet the precision goal with no immediate pressure on the volume of data that we would produce. The general strategy being to start small, with intentionally low coverage, but a high-quality feed. From there, we could work on delivering incremental value in two ways. The first involved extending the coverage through an iteration of gap analysis (identifying where we failed to deliver data and make the required improvements to ensure that we would succeed in those specific extraction scenarios). The second involved identifying additional properties of local businesses that we could deliver. Local business sites, depending on the type of business, have many interesting properties that benefit local search users such as business hours, amenities, menus, social media identities, services offered, and so on.

Discovering the Value Equilibrium By establishing the evaluation process, the customer helped frame the notion of value. In many cases, declaring a target – say the data has to be 80% precise with 100% coverage – is acceptable, but a worthy customer would require more. In many projects, the data being delivered will be consumed by another part of a larger system and further inference will be done on it. This downstream process may alter the perceived quality of the data you are delivering and have implications for the goals being set. There is a tension between the lazy customer – for whom the simplest thing to do is to ask for perfect data – and the lazy developer, for whom the simplest thing to do is deliver baseline data. Setting a target is easy, but setting the right target requires an investment. The most important goal to determine is the form and quality of the output that will result in a positive impact downstream. This is the exact point at which the system delivers value. While managing this tension is a key part of the discussion between the customer and the team, it is even better to have the downstream customer engage in consuming the data before any initial goal is set, to better understand and continuously improve the true requirements.

Of course, determining the exact point at which value is achieved is an ideal. If the downstream consumer is unclear on *how* to achieve their end goal, then determining the point-of-value for your product would require them to complete

their system. You have something of a data science "chicken and the egg" problem. In some cases, it is pragmatic to agree on some approximation of value and develop to that baseline. After this, the consumer can determine if it is enough for their needs.

There is one situation in which value in a data product can be relatively easily determined. If there is an existing product, then the customer can determine value in economic terms relative to the comparative quality of the data you provide. They may be happy to get slightly lower-quality data in return for lower costs, or pay more to get a significantly better product.

From Early to Continuous Delivery

We left our project, extracting local business data from the Web, open to delivering additional value in several ways:

1. Delivering more entities and covering the greater universe of local businesses in the market
2. Extracting more detailed attributes per business (such as images or reviews) and providing richer facets of information to users of the data
3. Applying the approach to different markets (including different languages), thereby extending the impact globally
4. Applying this approach to different verticals and extracting things like events or product information from a larger universe of web sites
5. Finding additional consumers of the data
6. Considering delivering the platform and tool chain itself as a product either internally or externally

Each avenue offers an opportunity to continuously deliver value to both the existing customer and parent corporation. The more we deliver to our immediate customer, the more we can positively and efficiently impact the product and our end users; the greater the ROI.

More Entities

Once we had established the feed as a viable data source for our first-party customer, we set about attacking the coverage problem from a couple of analytical angles. On one hand, specific implementation details and presentation formats of the web sites could make it easier or harder to extract the desired information. Take the address for example. If it was present in a single HTML node (such as a div or span) on the site and the site was presented in as a static HTML file, then extraction would be relatively simple. If, on the other hand, the site dynamically loaded content or used an image to present the address, then a more sophisticated technology would be required. These challenges necessitated successive investments in what we termed *technical escalation*. In other words, to get more data, we needed to increase the sophistication of our extraction stack.

Another consideration that our data analysis presented was the complexity of the underlying business. A small business with a single location would generally not present too much confusing information on their site. It would have a unique address, a single phone number, and so on. Now consider something marginally more interesting – a local bakery that participates in farmers' markets during the weekend. Now the site might legitimately include additional address information – the addresses of the markets where it would appear in the next few weeks. Fully exploring business complexity, we discovered that multi-brand, international chains not only present with multiple brands at many – potentially thousands – locations but through necessity leverage dynamic and interactive web site architecture to implement search engines for their customers to locate a chain location.

How to Look at Data It is always important to view data in a form identical to that of your system's view, especially on the Web. If your system is processing web pages, reviewing example pages in a browser will hide many details, for example, there may be invisible portions of text in the underlying HTML. Viewing such a page in a browser will hide these, but your inference system will read these as being equivalent to the visible data. Only when your team has a full understanding of the underlying "natural" form of the data should you employ views that transform the data in some way.

With these two rich (and not entirely independent) dimensions to explore, we approached the planning by first prioritizing the potential investments in terms of expected impact and estimated cost. Second, we iterated on solutions to those areas in a tight loop: implement, evaluate, and improve.

In this way, we continuously improved our product by delivering more impact through coverage while iterating both on our overall architecture and on individual components.

The relationship between the concepts being modeled and the system requirements can be illustrated by our need to adapt the system to handle chain web sites. A chain, technically, is any business that has multiple locations for which the customer experience is interchangeable. An example would be McDonald's, because you get essentially the same experience at any location. Chains often make their location data available to site visitors through a search interaction, for example, you enter a zip code in the Starbucks store locator and get a response with a list of coffee houses in or near that location. To crawl and extract from these pages, we needed to emulate the user searching, dynamically render the result page, and follow the links from that page to individual store detail pages.

More Attributes

Another way we iterated on the value the system delivered was through additional attributes. Early on, we chose a design for the core of our system that included a simple interface for functions we called *annotators*. An annotator is the computational analogy of a human annotator who looks at a page and highlights spans of text that denote a specific type of thing. We created name annotators, address annotators, phone annotators, and so on. Each could be implemented in complete isolation, tested, evaluated, and deployed to the pipeline through a configuration – we call this approach a plugin architecture (see Figure 1-1 for an overview).

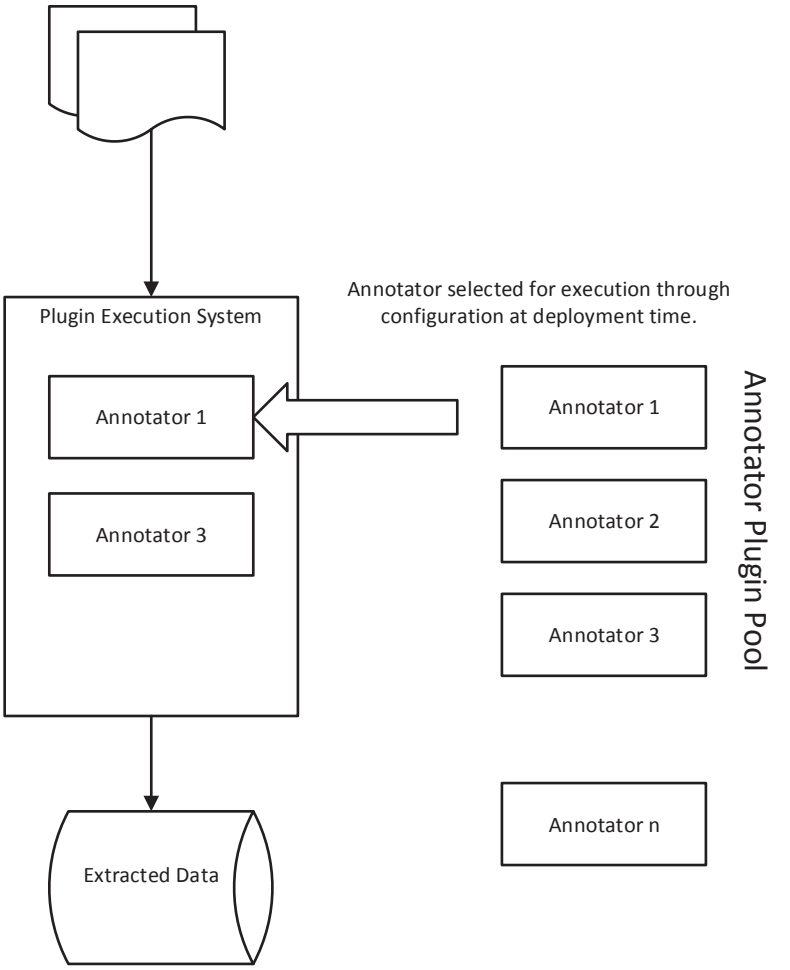

Figure 1-1. *In a plugin architecture, components – in this case, annotators – can be developed, tested, and managed independently of the runtime. A configuration is used to deploy components to a pipeline as required.*

Isolation allowed us to work on an annotator, without dependency on other components, and deploy with no additional testing to other components. In this way, we could expand from our original three properties – name, address, phone – to an arbitrary set of properties. Perhaps the most interesting of these was the business hours annotator. Understanding how this was developed provides some interesting insights into the relationship between the nature of the entities in the real world, the manner in which they present themselves on the Web, and the requirements of a system designed to extract that data in the context of a local search engine.

It is possible to model business hours semantically by intuition – many businesses have a simple representation of open and closed times, one for each day of the week. But it requires looking at the data, that is, actual business web sites, to truly understand the sophistication required to both model the domain and build the right extraction technology. Businesses may use inclusive and exclusive expressions ("open from 9 to 5 but closed from 12 to 1 for lunch"), they may be vague ("open until sundown"), and they may describe exceptions ("except for national holidays"). In more sophisticated sites, business hours may present in the moment, that is, in a way that only makes sense in the context of the date the site is visited ("Today: 9–5, Tomorrow: 9–6, etc.).

Studying the data led us to think hard about how we collect data in the first place through crawling the Web. While the general web index was fine as a starting point, if just 1% of businesses in our corpus were changing their business hours every week, we would need to refresh our data far more frequently than that of a large-scale index which would not update most business pages as quickly as it would update a more dynamic page like that of a news site or social site. In reality, the number of sites required to service a local business extraction system is many orders of magnitude smaller than that required for the general web search engine. And so, it made perfect sense for us to remove our dependency on this initial approach and invest in our own crawling schedule.

Finally, business hours are not linguistically similar to names, addresses, or phone numbers, having far more internal structure and variance of expression. This meant that we had to invest more heavily in the specific annotator to deliver business hours data. Again, studying the data informs the next iteration of architecture, requirements, and technology choices.

More Markets

As the project progressed, we began connecting with other teams that were interested in the general proposition – leveraging the Web as an alternative to licensed feeds for local search. This led to partnerships with teams tasked with delivering local data for other markets (e.g., a country-language pair). By taking the pluggable approach to architecture and partnering with other teams, we transitioned from being a *solution* for a single scenario to a *platform* for a growing number of scenarios.

Different markets presented different motivations and variables. In some cases, less budget for licensed data incentivized the team for that market to seek out more automated approaches to delivering data; in others, there simply weren't any viable

sources of data approaching the quality that the end product demanded. To continue delivering on the investment at this level required us to fully support the operational side of the system (from deployment automation to DRI[7] duties). We provided approaches to deliver the pluggable components in a way that could leverage the experience in the original instantiation, but was also flexible enough to meet the specific needs of those markets, and a commitment to creating, maintaining, and improving tools for every part of the developer workflow.

More Quality

Expanding the coverage of a system requires constantly assessing and maintaining the quality of the output. In some cases, this can involve adjusting and evolving the current method. In others, it requires investing in a switch to a more general approach or some sort of hybrid or ensemble approach. In our journey with local extraction from the Web, we encountered many instances of this.

We wanted to switch the method we were using for address extraction. The initial commodity solution, which was giving us reasonable precision but lacked recall, had to be replaced. We opted for a machine-learned approach (based on sequence labeling) and an efficient learning method similar to active learning. To make the transition, we created a regression set and continuously improved the model until we were satisfied that it was a viable replacement. From there, we could take advantage of a more general model to continue to extend our recall and maintain quality.

Another very real aspect of shipping industrial inference systems is managing errors that impact the current performance or are otherwise of great importance to one of your consumers. A typical problem with extracting from the Web is that the distribution of problems in the population changes over time. For example, while the system might have happily been delivering data from a specific site for some time, if the site makes a change to a presentation format that wasn't present or was rare in the development period, the system may suddenly stop producing quality results, or any results at all. If the consuming system was, for some reason, very sensitive to the data from this site, then it would be pragmatic to implement either some form of protection or some form of rapid mitigation process.

[7]A DRI is an acronym for "Designated Responsible Individual." This is the person who is on call on a particular week to resolve any live site issues that may come up with the running product.

The Platform as a Product: More Verticals and Customers

Having expanded the contributions made by the investment in web mining through coverage, attributes, and markets, the future for additional value lies in transforming the platform itself into a product. Getting to this level requires a decision around the business model for the product. While our original goal was to deliver business data, expanding to ship the platform as a general-purpose capability meant that we would be shipping software and services. This path – from solution to platform to product – is not uncommon and represents orders-of-magnitude jumps in value of investment at each step.

Early and Continuous Delivery of Value

Projects that aim to extract value from data through some form of inference are often, essentially, research projects. Research projects involve many unknowns and come with certain types of risk. These unknowns and risks tend to be markedly different from those in traditional application development. Consequently, we need to rethink our notions of the customer, the core development iteration, what form our product takes, and how we deliver.

At the core of the types of projects we are interested in is some type of inference over potentially large data. Your team must have the ability to intelligently explore the search space of data and inference solutions in the context of an overall product and well-defined business need, so that

- The customer is engaged, and their expectations are managed – an engaged customer will ensure that the details of the final product align with the business case requirements. The customer will see the implication of the project – especially how the nature and potential of the data can influence the final result and be best consumed.
- Progress is articulated appropriately and continuously – including prototypes, preliminary improvements that are below the required standard for a Minimal Viable Product (MVP), and eventual success in delivering and maintaining the data product. An ideal customer will even get ahead of delivery and assist in the success of the project by building stubs or prototypes of the consuming system if they are not already available.

- The customer fulfills the role of helping the team to establish and prioritize work. The customer understands that involvement is a vital and equal voice that helps guide, shape, and land the project.

Projects, both traditional and data, tend to have the following phases:

1. Initialization
2. Delivery of a baseline product, or Minimal Viable Product (MVP)
3. Delivery to requirements
4. Maintenance
5. Expansion

In most of the phases, the central development loop, that is, the process by which your team will deliver some sort of inference capability, will be characterized by the workflow illustrated in Figure 1-2.

- Requirements generation is an iterative process involving the customer and the development team, refining the initial tasks through analysis of data representative of the population targeted by the system.
- An approach to evaluation is determined and implemented to allow progress toward the targeted data quality to be tracked.
- The development team collects data to be used in developing inference components, including training data and evaluation data for their inner loop.
- Evaluation, data analysis, and error analysis are used to improve the models but also, where needed, to improve the evaluation process itself.

Our goal in this book is to help you navigate delivery through this workflow and to understand how the contexts of each process influence the overall success.

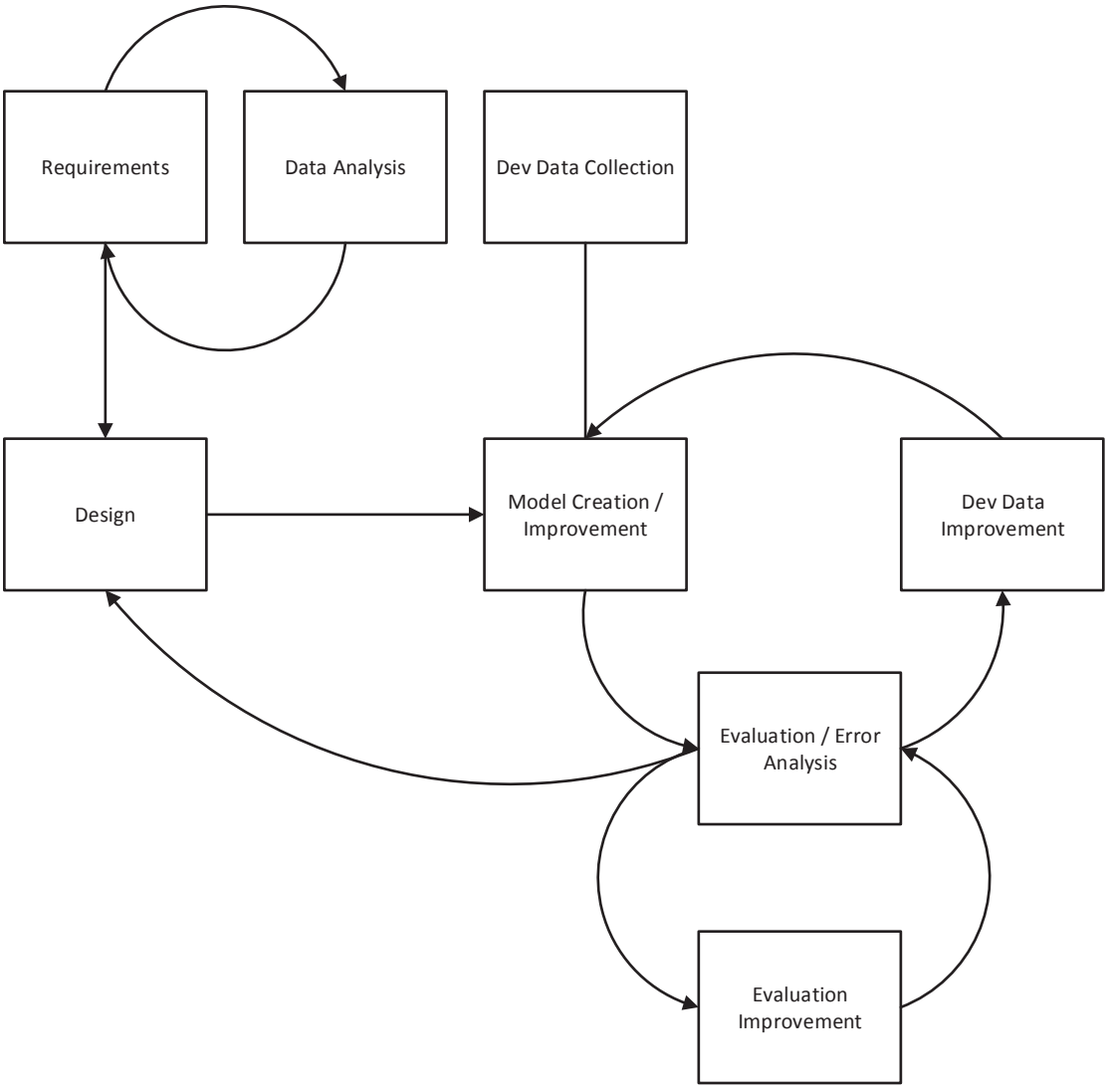

Figure 1-2. *The process by which a team delivers inference capability*

While a lot of focus is given to advances in the core models and training techniques in the inner loop, in industrial contexts, it is also important to acknowledge the position your contribution will have in the overall workflow of the deployed system. As shown in Figure 1-3, you may be responsible for a simple, single-component system or be building

a larger, multicomponent system or developing a subset of such a system. In any of these cases, you may have one or more internal and external customers.[8]

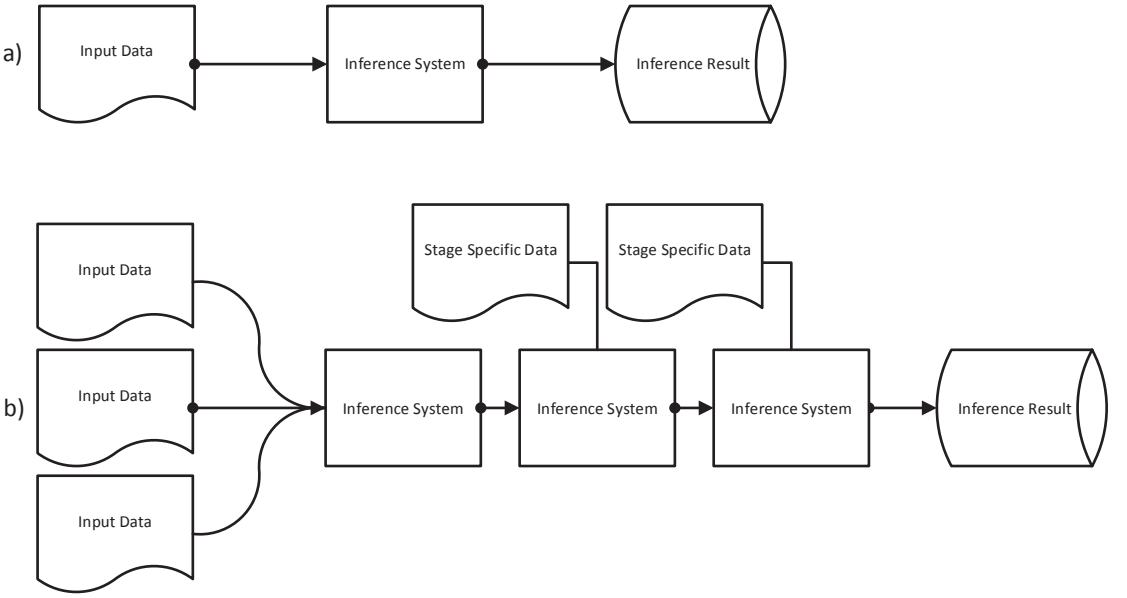

Figure 1-3. *Simple end-to-end system (a) vs. a multi-data, multicomponent system (b)*

Finally, there is the selection of an appropriate strategy of progress (Figure 1-4). Given a domain of data and an inference problem, we can aim to cover the domain entirely from the beginning and make improvements overall to the quality of our output, or we can partition the problem into sub-areas of the data domain (with distinct characteristics that allow for narrow focus in problem solving) and achieve high quality in those sub-areas before moving on to include additional tranches of the domain. For example, if we were building a system to identify different types of wildlife in video, we could start with building a data set for all types of animals and then training our system to recognize them. For a fixed cost, we would have a small amount of data for

[8]The notion of internal and external is somewhat arbitrary, especially in large corporations. The reality is that some customers are closer to you organizationally and some are at a greater distance.

each animal, and our recognition quality might be low. For the same cost, we could alternatively create labeled data for just a small number of animals and get quickly end to end with high quality.

Another example of this is in the domain of information extraction from the Web. A "quality first" strategy would focus on simple web sites where there is little variation between sites and few technical hurdles (e.g., we could limit our input to plain HTML and ignore, for the time being, problems associated with dynamically rendered pages – images, flash, embedded content, etc.). A "breadth first" strategy would attempt to extract from all these forms of input with the initial system and likely end up with lower-quality extractions in the output.

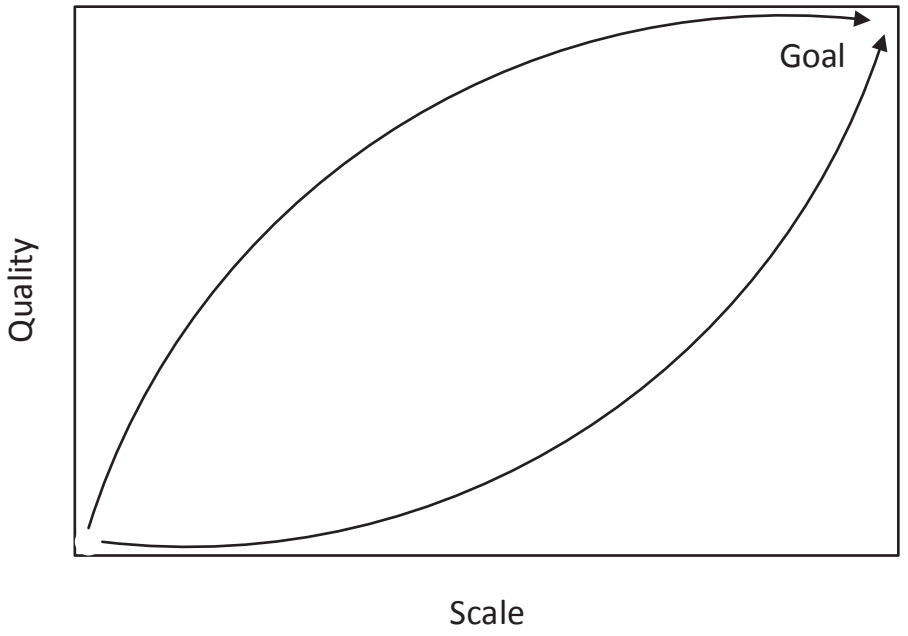

Figure 1-4. *Strategy for progress in large-scale inference projects. The top line represents prioritizing quality first and then turning attention to scale. Here, scale means the ability to handle more and more types of input. In a document processing system, for example, this would mean processing many genres of document in many file formats. The bottom line represents prioritizing scale first and then working on quality – in other words, investing in handling a large variety of documents and later driving for quality. In an agile process, the top line is preferred as it is the shortest route to delivering value to the customer.*

Conclusion

In Chapter 1, "Early Delivery", we looked at ways in which a team can get up and running on a new inference project. We touched on the central role of metrics and a culture that is biased to action to enable quick iterations using data analysis to inform the evolution of all aspects of the project.

In Chapter 2, "Changing Requirements", we will discuss how you can build systems that are designed for change.