Industry DS/ML

CONSIDERATIONS IN DEVELOPING CUSTOMER-FACING TECHNOLOGIES

Overview

This talk is meant to be a high-level overview of typical industry processes and procedures when handling a new project, start to finish.

This is not meant to be exhaustive as there is too much variation across business sectors, but should highlight common themes and practical considerations.

- About Me
- What's the Context?
- Humble Beginnings: The Data
- Having Fun: The Solution
- Show and Tell: Productionalizing

About Me

AKA: WHY SHOULD WE CARE WHAT YOU HAVE TO SAY, MR. BEERY?

A Quick Tour of my CV: Sectors and Technologies

- Healthcare (Data Scientist)
  - Modeling: HMMs, MELR, SVMs, MGPS, BCPNN
  - Building: MSSQL, Azure, TFS, Visual Studio
  - Problems: Drug and medical device surveillance (time series), risk analysis, PoS tagging, text classification, graph analysis
- Computational Photography (ML Engineer)
  - Modeling: Graph cuts, KNN, CNNs, MLPs, Nelder-Mead optimization, Bayesian Optimization
  - Building: VS Code, Git, Archlinux
  - Problems: Monocular depth mapping, edge computing, blur detection, hyperfocal analysis, multi-system optimization, image classification

More Sectors and Technologies

- Wide band radio-frequency analysis (ML Engineer)
  - Modeling: DBSCAN, Autoencoders, ARIMA, CNNs, RNNs
  - Building: FPGA, VS Code, Remote Storage Servers
  - Problems: Point-of-Collection Inference, anomaly detection, signal classification, signal tracking
- Document Analysis (Data Science Team Lead)
  - Modeling: XGBoost, CNNs, GNNs, Bayesian Optimization, Siamese Networks, Transformers
  - Building: AWS {Sagemaker, S3, Lambdas, Textract, etc.}, VS Code, Bitbucket (Jira), Git
  - Problems: Layout analysis, object detection, structure recognition, Key:Value extraction, text classification, image classification

What's the Context?

AKA: MOST OF THE EXPERIENCE YOU GET FROM ACADEMIA WON'T HELP YOU

What's the Context?

- Industry applications of ML/AI/DS operate under very different constraints and expectations than most students are used to.
- Collaborative maintenance of code, rapid development cycles, continually shifting data, changing deadlines and high-pressure environments require controlled, principled and flexible solutions.
- Bringing ideas to life for customers follows a model development lifecycle that never truly ends.
  - This means utilizing automation, automating the automation and ensuring that the automated automation is automatically generalizable.

What's the Context? – Cont.

- Many learning institutions will teach singular aspects of popular industry frameworks, e.g. AWS, Google Cloud, Azure, etc.
  - How to use S3 buckets, creating a lambda, etc.
- Most modern ML/AI/DS pipelines involve many more moving components.
  - How do we maintain our massive data stores cost efficiently while allowing rapid access?
  - How do we efficiently train models?
  - What's the best UI/UX for fronting our new model?
  - How do we hand off development efforts to engineering?

Semantik Overview

Creator: Mark Studer | Last Modified: July 14, 2021

Legend

- Data Flow
- Incoming Events
- API Requests

Description

This architecture diagram is a high level overview of the Semantik Platform with links to the subcomponent diagrams where available.

DNS *- Route 53 provides the regions specific routing rules. This includes the routing of users to landing pages and routing of client C-Names like client.ephesoft.net.*

Ingress Filtering *- AWS and Route 53 can be used together to mitigate DDoS attacks.*

Authentication *- All public endpoints require authentication.*

Encryption *- KMS is used to encrypt all data using SSE.*

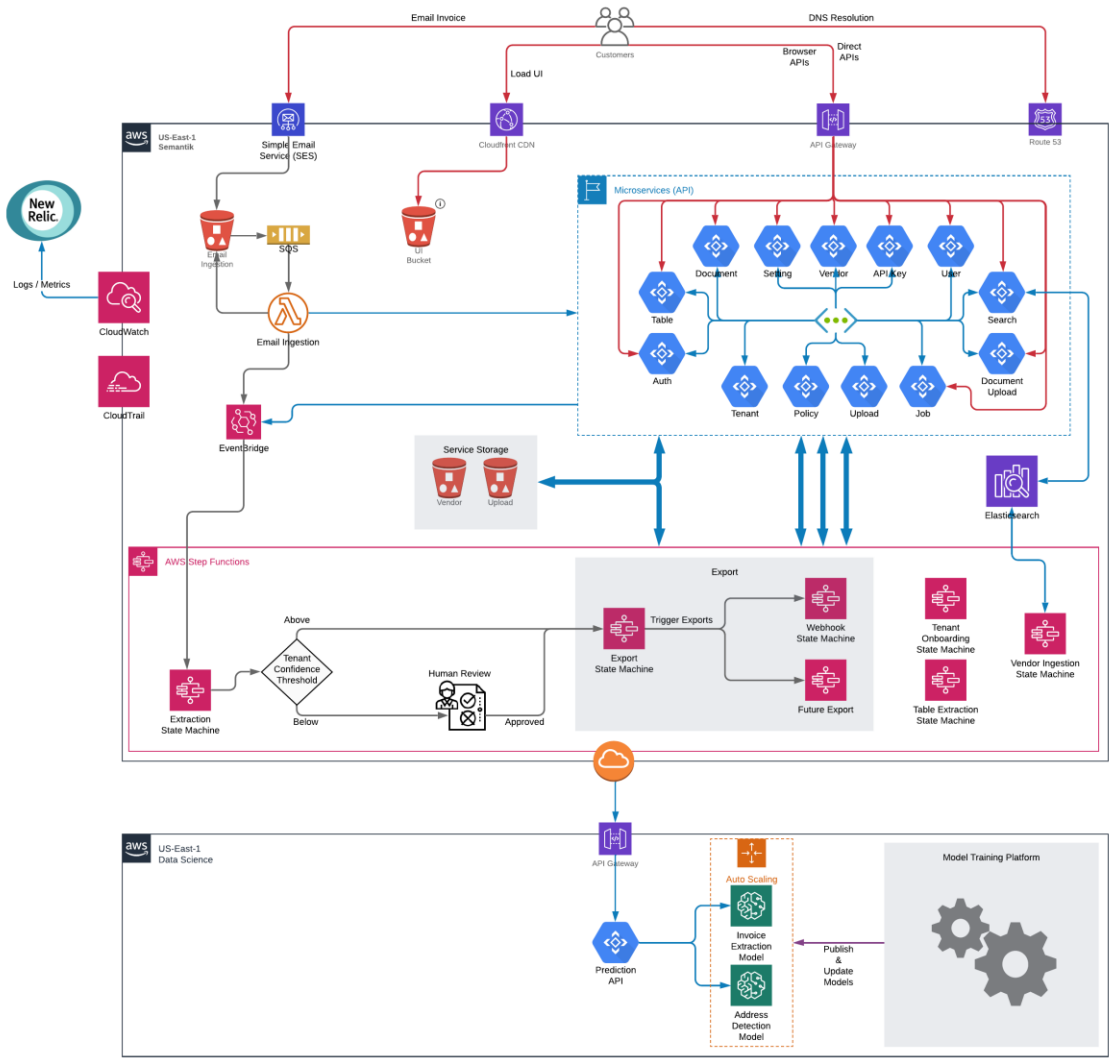
Queueing *- SQS / SNS and EventBridge are used to handle queueing throughout the system.*

Orchestration *- AWS Step Functions are used to orchestrate workflows.*

Data Storage *- Semantik uses DynamoDB, S3 and Neptune Graph datastores.*

Text Extraction *- AWS Textract provides extraction for all documents. In the future we want to provide support for multiple OCR vendors. The key to providing this will be creating adapters to convert OCR output into a common JSON format. This allows output to flow form the extraction state to the rest of the pipeline without worrying about the particular OCR vendor used to extract the data.*

Export *- The export state machine creates the document and table exports and then calls the appropriate export state machine to communicate with the client. The only currently supported export is Webhook.*



Legend

- Data Flow
- Incoming Events
- API Requests
- Storage S3 Access

Description

This architecture diagram shows the flow of an Invoice through the Extraction State Machine and its interactions.

Encryption *- KMS is used to encrypt all data using SSE.*

Queueing *- SQS and SNS are used to handle queueing throughout the system.*

Orchestration *- AWS Step Functions are used to orchestrate workflows.*

Text Extraction *- AWS Textract provides extraction for all documents. In the future we want to provide support for multiple OCR vendors. The key to providing this will be creating adapters to convert OCR output into a common JSON format. This allows output to flow form the extraction state to the rest of the pipeline without worrying about the particular OCR vendor used to extract the data.*

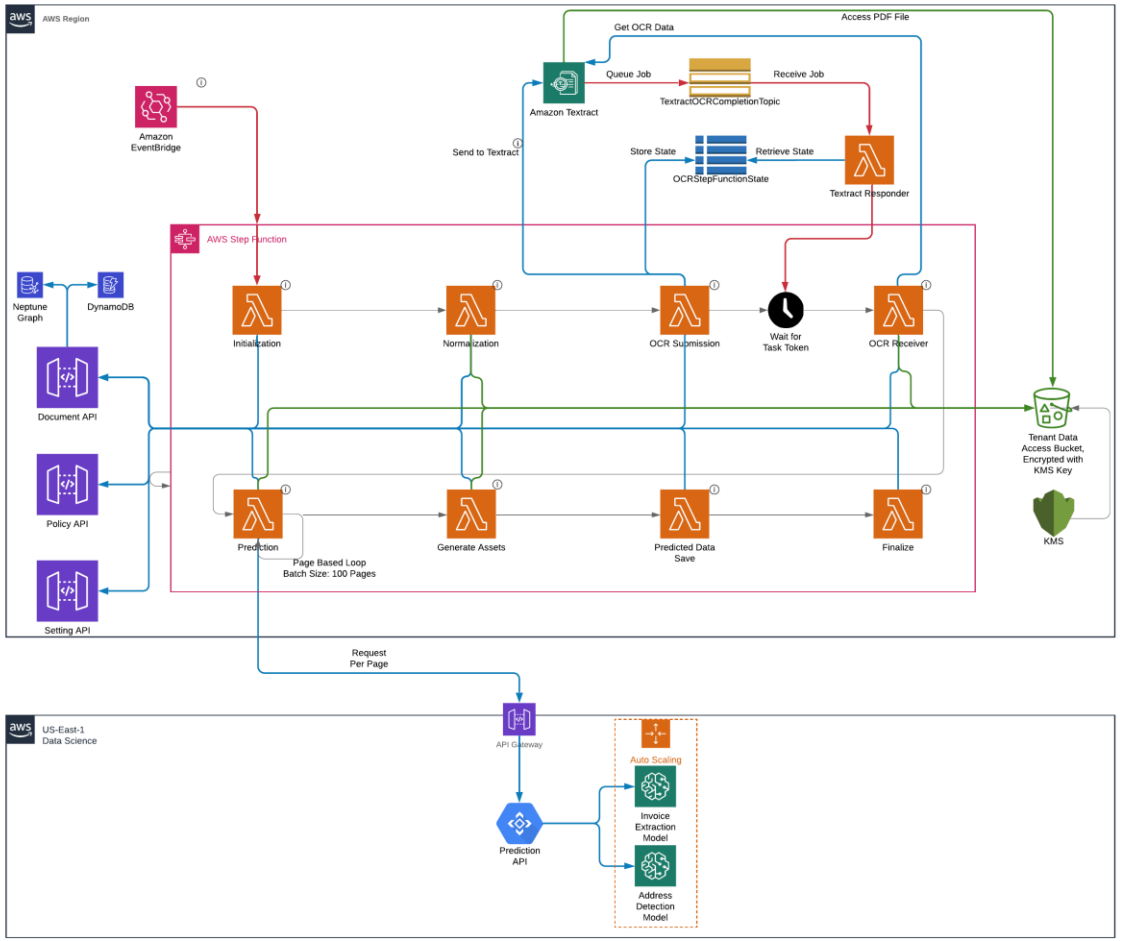
Prediction *- Sagemaker hosts Prediction API's in the Data Science Account that are used for doing page level predictions.*

Resource Types

SNS Topic, DynamoDB Table, KMS, S3 Bucket, Lambda, Step Function, API Gateway, Microservice, Neptune Graph Database, DynamoDB Key-Value Database, SageMaker, Textract

SM-Extraction

Creator: Mark Studer | Last Modified: July 14, 2021



Q&A

Humble Beginnings: The Data

AKA: HOW YOUR FIRST STEPS BUILD A STRONG FOUNDATION OR LEAD TO EMBARRASSING CONVERSATIONS WITH LEADERSHIP

Is There a Problem, *Sir*?

- If not, there should be! A good problem statement is key to coming up with a deliverable solution.
  - Industry **YY** spends **$XX** each year doing **ZZ**. There is no existing system that can take input **AA** and provide output **BB** which would save an estimated **$CC.**
- With a strong problem statement we identify the business opportunity as well as understand what obstacles our technology is hoping to overcome.
- Equipped with this point of view, we can determine the exact type of data we need to address the problem.

Evaluating the Data

- Once we know the type of data we need, we have to take stock of what data is available. There are three common scenarios here that you'll run into:
  - Sometimes the data we need is unavailable or unreasonable to obtain. This is a non-starter and as data scientists we need to understand there will be times that we need to focus efforts elsewhere.
  - Sometimes the data is available, but unlabeled, too messy, almost unusable. These cases require either an incredibly innovative solution or an immense initial cleansing effort.
  - Sometimes the data is available and labeled, or easy to label, or a good candidate for assistive tooling.

Evaluating the Data – Cont.

- Once we've determined the state of the data, we also need to determine if it can solve the problem statement with it.
  - For example: the problem we're trying to solve is determining the likelihood of success of a particular cancer treatment on a patient-by-patient basis. We have secured records of 10,000 patients and they are fully coded and digital. We meet all HIPAA regulations and are ready to go. But… The records are all from diabetes clinics.
  - No matter how good the data is, we will never be able to make an inferential system for cancer based on diabetes records.
- Evaluating your data to ensure it can solve the problem is key to success.

Preparing the Data

- If you want to be successful on a team with data, here are a few crucial pieces of advice:
  - Decide how to annotate data only AFTER you research possible models you could use
  - Define annotation instructions and edge-cases BEFORE annotation work begins
  - DON'T keep local copies of data that only you work on
  - Make sure you MONITOR and TRACK changes to the data
  - DON'T give write permissions to your training/validation/test sets except for updating

Q&A

Having Fun: Building Solutions

AKA: DON'T JUST HAVE FUN. THAT $5 MILLION CONTRACT DEPENDS ON YOU DELIVERING A WORKING SOLUTION.

DS/ML is Just Digital Carpentry

- At the end of the day, the most important thing you can have as a data scientist is a good toolbox, stuffed with tools for every job.
- Part of your responsibility is knowing (and continuing to learn) all the various ways of solving a problem, their strengths and their weaknesses.
  - You can't use a table saw as a hammer, but you should know that you have wooden mallets, rubber mallets, 12oz hammers, sledge hammers, etc at your disposal.

Peeking is Allowed

- The best part of being an industry data scientist is we can look just as smart as the academics without any of the hard work. :P
  - Publications come out at a ridiculous pace these days. In machine learning, nearly all these publications are freely available and often have accompanying code.
  - Many academic solutions require serious code re-writes and modifications to work in a real-world environment, but the core principles and theory driving the advancement are already done.
- You should always consider what has already been done before deciding what to do. (Don't reinvent the wheel)

Don’t Bring a Gun to a Pillow Fight

- It’s often tempting to use the newest, biggest, most powerful model you can find to solve a problem. After all, it gives the best accuracy, right?
  - If the success metric is only maximizing accuracy or F1, this approach is fine.
  - But often there are other considerations: what hardware is available when performing inference? Are there budget constraints that limit system performance? Are we competing for system resources with other software?
- You should fully understand the operational context of any system you are developing. Developing a solution with 100% accuracy doesn’t matter if it can only produce one prediction per minute in a high volume setting.

This is Not the Never-ending Story

- Many data science and machine learning projects fall prey to poorly defined success criteria.
  - It's easy to always have an eye on that "one more improvement" or the "I'll try one more thing to boost performance"
  - It's possible to spend months on the same model or algorithm, fine-tuning, tweaking, experimenting, but ultimately there is no business benefit from doing so.
- The data science development cycle should always include success criteria that determine when initial work will stop. Once a performance objective is met, the model can be moved to production for customers.
- It's easy to revisit and improve a model later, but you can't go back in time and work on another project that needed to get done.

Wax On, Wax Off

- In order to master the karate of data science, we have to spend a lot of time mastering the more day-to-day basic tasks:
  - Using source control, learning good programming practices and assisting the team through PRs, proposal feedback, etc.
  - I want to really stress the importance of mastering the programming language used at the company. Maintaining a function with 200 lines of code is a nightmare. Mastering a system with 1000 lines of code across 40 functions and 3 classes is easy.

Q&A

Show and Tell: Productionalization

AKA: YOU CAN'T PLEASE EVERYONE, SO YOU'VE GOT TO BUILD ROBUST PRODUCTION PIPELINES

The Good

- Today there are a myriad of ways to serve data science solutions to customers:
  - Cloud-based systems
  - On-premise solutions
  - Edge-computing (think IoT)
- Each method comes with its own constraints and challenges, but building solutions for them is common and well-documented
- Depending on the industry, you should be familiar with how to serve models in any of these contexts

The Bad

- Getting a model into production is just part of the process. Making sure each iteration works as expected and keeping customers happy is a longer, more arduous journey.
  - It's not fun, but writing robust unit testing and integration testing prevent a lot of problems by identifying bugs and model weaknesses before it reaches customers.
  - Periodic model failure analysis based on customer feedback is also important to identify failure modes, add testing and target high-priority improvement areas.

The Ugly

- Even a well-structured and automated model development lifecycle generates a lot of details that need to be handled:
  - Each model iteration needs to be versioned, benchmarked and either rejected or deployed. In a CI/CD workflow, this can lead to dozens of versions produced each month.
  - Models need to be continually monitored for drift – when the real-world distribution of data is no longer captured by the training/test data.

Concluding Thoughts

- Being successful as a data scientist requires being methodical in every step of your development lifecycle.
  - Build strong foundations with data.
  - Learn how to maximize the potential of each tool in your toolbox.
  - Remember that getting something into production isn't the end, it's just the beginning of a different cycle.
- Don't just be competent in a programming language. Really make efforts to master it. Efficient, fast and reusable code is your best friend.
- Always keep an eye on the end goal: customer satisfaction. You'll always build better solutions when you understand why.

Thank you!

And now for more Q&A…

Q&A