# Service Desk Automation using Machine Learning

Ryan Rosiak
Intern at JP Morgan Chase
COSC 380-001

# Who is JP Morgan Chase?

- An American multinational investment bank and financial services holding company
  - Multinational
    - Operate in over 60 countries
    - Over 240,000 employees
  - Investment Bank
    - Advisory based regulated transactions on behalf of individuals, corporations, and governments (Mergers, Acquisitions, Trading, etc)
  - Financial Services
    - Manage money
  - Holding Company
    - Owns shares in other companies (Parent Company)
    - Manage assets of owned companies

Fun Fact: JP Morgan Chase is the 3rd largest bank in the world controlling over $3.68 trillion dollars in assets

# My Role

- Consumer and Community Banking Division
  - Banking with the direct public (Chase credit cards, Websites, Mobile Banking, etc)
- Architecture and Data Engineering
  - Tech division
- Photon Framework Team
  - Internal framework that provides a layer of standard features on top of backend architecture (DPL, Database, Auth, etc)
- Team of 4 interns
  - Team name: NP-Compete

# The Problem

- Photon team receives roughly 130 support tickets per sprint
- Framework usage increasing -> Ticket rate rapidly increasing
- 90% of tickets fall under consultation
    - Problems solved by:
        i. **Pointing customer to correct documentation (approx. 82%)**
        ii. 1:1 walkthrough
- 5 engineers on support
    - Do not want to add more!!!
- Less engineers on support -> More engineers developing new features

# The Solution

- Implement service desk automation using Jira bot
  - Support requests submitted through Jira
- Jira bot replies to support requests on Jira
- Two part answers to support requests:
  - **Use NLP to provide a targeted answer (Title, Description, Version(s) Affected)**
  - **Use tagged components from request to provide default answer**
- NLP answer will provide a link to
  - Documentation
  - Previous ticket threads
  - KB articles
- Default answers will provide a link to
  - Documentation

# E2E Project Flow Example

# Team Goals

1. **Add targeted answer functionality to Jira bot using machine learning**
2. Build curated test data to validate model of choice
3. **Provide default answer based on components selected**
4. **Deploy to GAP 3.0 to be used by developers**

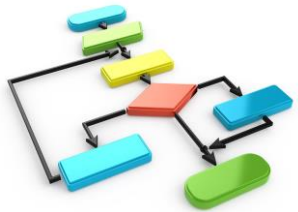
# Timeline

# Researching

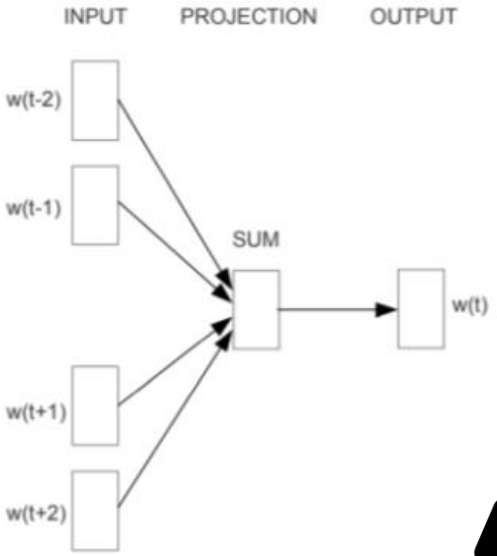
- NLP libraries
  - NLTK
  - Huggingface
  - Gensim
- Machine Learning NLP concepts
  - Word Vectorization
- Machine Learning NLP models
  - **Doc2Vec**/Word2Vec ← Picked this one
  - Bert/ElasticSearch
  - Chatterbot
  - Rasa
- Model algorithms and infrastructure

# The Algorithm/Model

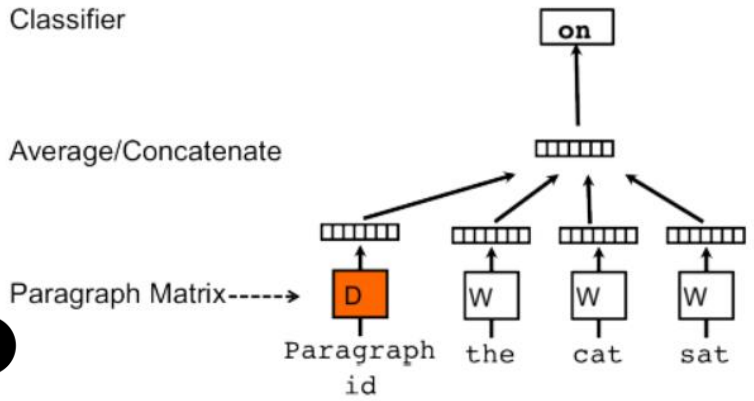
- What is Doc2Vec?
  - Unsupervised ML algorithm trained on unlabeled strings
  - Assign numeric value to document of strings
  - Find “most similar” documents by finding the smallest distances
- Why was Doc2Vec picked?
  - Algorithmic logic already written (Only write business logic)
  - Allowed for easy training and updating of model
  - Performed the best under problem domain (Non-restricted input)
- Parameters: CBOW, Sampling alg, min_count, window_size, vector_size



# The Algorithm/Model Continued




CBOW




# Writing Software

- **All software written in Python**
- Independent modules for interacting with model
  - Creating/Training/Saving/Loading the model
- Generating graphs and metrics for model
- Optimizing code
  - Multithreading
- Unit tests
- Generating bot responses
- Interacting with Jira API and Jira webhook
- Loading modules onto bot
- Writing configuration files
  - Deployment

# Deploying

- What is GAP? (Gaia Application Platform)
  - A platform to run your application asynchronously “forever”
  - Prototype → Production quickly!!!!
  - Creates a container of your app with everything it needs installed
  - Industry grade security
  - Compatibility with most frameworks and database systems
- Updating manifest file
  - Adding dependencies
- Configuring for python project
- **Reading A LOT of logs!!!**
- Configuring code for production environment
- Managing credentials
  - EPV-AIM
  - MariaDB credentials

# Endnotes

# What I Learned

- Python in a real world environment
  - Multithreading
  - Writing tests / Debugging
  - NLP libraries
- Database management (MariaDB)
- Implementing a complex ML model
  - Algorithm analysis
- Adding features to an existing codebase
  - Reading others code
- Industry standard applications
  - Jira workflow, Confluence, Bitbucket
- Writing documentation
- Development practices
  - Whiteboard
  - Kanban boards (Scrum)
- Development environment / Working on a team

# Challenges I Faced

- Learning new concepts
- Debugging
  - Software
  - Logs
- Presenting
  - Photon team
  - Upper management
- Remote workflow
- Managing connections

# Classroom Experience Influence

- Debugging projects (All CS courses)
- Algorithm Analysis (220/320)
- ML fundamentals (311)
- Database (SQL) fundamentals (386)
- Programming fundamentals (All CS courses)
- Problem solving (All CS and Math courses)

# Any Questions?