Use machine learning to predict relevant support content based on historical user interactions

PROJECT PLAN

Sddec18 -16
Client: Workiva
Advisers: Neil Zhenqiang Gong
Team Members/Roles
Erin Elsbernd: Communication Coordinator and Machine Learning Lead
Ram Luitel: Project Manager and Software Architect
Faizul Jasmi: Testing and AWS Tech Lead
Taizhong Huang: QA Lead
Christian Chiang: Webmaster and AWS Tech Lead
Khoa Bui: Webmaster and DB lead

http://sddec18-16.sd.ece.iastate.edu/
sddec18-16@iastate.edu

Revised: April 19/Version 3

Table of Contents

List of Figures

List of Tables

List of Symbols

List of Definitions

1 Introductory Material

1.1 Acknowledgement

We feel very thankful and fortunate to be assigned to this project which will help us develop our machine learning skills. The success and outcome of our project require a lot of guidance and assistance from many people and we are extremely privileged to have this input.

We respect and thank Mr. Alex Kharbush for developing this project and giving ISU students the chance to tackle the problem. We are also grateful for the resources and direction he provides throughout the week. We would also like to recognize Dr. Neil Gong for being our faulty advisor and meeting biweekly to provide necessary support and guidance.

We heartily thank Dr. Joseph Zambreno for constant encouragement, support and guidance which will be helpful to us to successfully complete our project work. Also, we would like to extend our sincere esteems to all teaching assistants for their input.

1.2 Problem Statement

Workiva has an application called Wdesk. At the moment Wdesk users utilize a search engine to search for help articles when they are having a problem with the app. However, the search bar is not the most effective at listing specific help articles based on the needs of the user. Searching for a topic requires browsing through many different articles, and then browsing through the contents of each of those articles to hopefully arrive at the solution to the user's problem.

Workiva would like to automate the help article search process by tracking the user's actions while using Wdesk and predicting a help article based on these actions. Currently there are no automated tools that suggest relevant help articles to the Wdesk user. Therefore, if a user cannot troubleshoot an issue using the search bar they will often call a customer support number. With this current setup, if Workiva wants to expand their business, they will need to hire additional customer support staff to handle larger accounts. This is not a sustainable business model. Workiva would like to provide a better customer support experience with their Wdesk app, by using predictive models to help the user troubleshoot instead of humans. This will save provide savings to cost and time.

1.3 Operating Environment

This will not be a stand alone application and will be dependent on the virtual environment provided by Workiva. Our application will eventually need to be part of the larger Wdesk application that Workiva client's use. The end product of this project will need to run in the cloud to simplify infrastructure management, deploy more quickly, lower cost, and gives real time solution. Our end product will be deployed in Amazon web Service(AWS). As our application will be platform independent users can use it on any operating system where Wdesk runs.

1.4 Intended Users and Intended Uses

This product will be used by Workiva customers as a part of the Wdesk application. It may also be used by employees developing Wdesk.

1.5 Assumptions and Limitations

The only cost factor for our project is Amazon Web Service licence which our client will provide us. The assumption we have is that our end product should compile and run on AWS as this is one of the requirements that our client has. If our models have over 70% prediction accuracy than Workiva would like to use our end product.

The limitations we have are technical skills among the team members. Only a few team members have machine learning experience through either classes or internships. We intend to mitigate this issue by setting aside research time for getting familiar with machine learning models and libraries. Some other limitations may include integration of our system with the Wdesk application. As Wdesk is a commercial application we might not have access to it when we are required to integrate our product with the full application. Testing limitations is another challenge as we can only test our model with the smaller datasets provided by our client.

1.6 Expected End Product and Other Deliverables

At the end of the final semester , we should have a fully functional model that can predict a relevant help article given a user's actions using Wdesk with over 70% accuracy. We will also need to host our model on AWS and build an automated data pipeline process around it. This will involve an automated data cleaning, data processing and model updating process. We will also need to deliver all project related documents that include design, code, project architecture documents and potential documents that include instructions on how to use our models. For more detail on deliverable dates please refer to section 2.10.

2 Proposed Approach and Statement of Work

2.1 Objective of the Task

The objective of our project is to create a recommendation system that can predict the most useful article for a given user based on how they have interacted with the Wdesk app. This recommendation model should eventually run on AWS and be able to be updated weekly, if not daily, with new data.

2.2 Functional Requirements

- The recommendation system should be able to make recommendations for help articles and display the recommended article ID.
- The classification model should be optimized to run as quickly as possible.
- The classification model should be able to be updated weekly given new data.

2.3 Constraints Considerations

Our team has little experience with machine learning and recommendation systems so we will have to do a great amount of research and learn quickly to complete the project. Our client would like to have a feasible classification model by May 2018 so that particular deliverable will need to be completed by that time.

We will also need to work closely with our client to ensure that our system is developed in a way where it could be eventually incorporated into the Wdesk app. We also need to take into account that our team is composed of undergraduate students who are working part-time, so some of the timelines may need to be readjusted.

2.4 Previous Work And Literature

Google, Amazon , and Facebook are all working on recommendation systems based on historical data of user behavior. Their advantages are that they have large amounts of data points to detect users' behaviours from different applications and more users' data to test and train. However, their users are complex; they have so many different groups of users who have totally different behavior patterns and knowledge backgrounds. In other words, their data set has a lot of dimensions and features. By contrast, we clearly know our target users; our target users are all using our client's product for work on similar accounting tasks.

Linkedin has a similar recommendation system to what we are developing for our client using a photon ml library.[1] However, Linkedin is using photon machine learning tools so that it can scale to millions of users. The approach they are using to do this is that they capture the user activity and use machine learning to scale resumes. We also want to leverage user activity for our models, but instead have them predict helpful articles. Currently our client doesn't have an automated recommendation system, so our final product will be of great value to the company.

In terms of the models recommendation systems use there are many approaches including collaborative filtering, content-based filtering and context-aware recommendation systems. The two articles referenced at the end of our plan by Zhang, Shuai, and Lina Yao Zhang and Jiacheng, Xu go into more depth on various approaches that harness the power of user data to make good recommendations. While these are all powerful models, they won't work well with the format our data is provided in, namely sequential time series data that can't be represented well in sparse matrix format. Another research paper we found enlightening in broadening our understanding of recommendation models was the one by Singhal, Ayush, Pradeep Sinha, and Rakesh Pant which provided an overview of neural network architectures we could reference for our project.[2] However, we don't have access to as much data as these models require and our sequential data format is also different to the types of data used in the paper. [3]

For our project, the scope is more limited as we will be making predictions for one particular application and have a significantly smaller set of user data to draw upon. There are many machine learning libraries available, e.g. tensorflow and scikit-learn that have models and tools to create a recommendation system. We will work closely with our client to ensure that our system fits the target users of the WDesk app.

2.5 Proposed Design

We are planning to implement several machine learning and deep learning models and select the best performing one for our final model to use in production. To implement our models we will be using scikit-learn and keras with a tensorflow backend. We selected these libraries because our client requested that we use python, and they are the best machine learning tools for python data analysis. Thus far, we have learned to implement basic machine learning and deep learning models on simple datasets. We will expand on these basic models to work with the user data. Our deep learning models will be largely selected based on client recommendations as our client has previous

[1] see LinkedIn post in References.

[2] Singhal, Ayush, Pradeep Sinha, and Rakesh Pant go into using deep learning instead of traditional collaborative filtering approaches.

[3] Zhang, Shuai, and Lina Yao Zhang and Jiacheng, Xu summarize and present more traditional recommendation systems that utilize user behavior data in their models.

experience managing machine learning teams at Amazon and has some idea of the models that may be best for our user data.

To process our data and generate features we plan to use scikit learn and nltk libraries as we plan to treat the data as text data or time series data, depending on the type of model we use. This way we can use bag of words and n-grams approaches to try to capture important action events and sequences of user interactions.

There are several open-source tools for our project. Tensorflow is not the only backend machine learning framework available, but as its provided by google the documentation and ease of use attracted us to that tool. Keras could perform the role of backend we assigned Tensorflow. They are both great frameworks for machine learning. Scikit-learn is another great tool for basic machine learning models and we have been using it for both feature creation and model creation of random forests and SVMs.

Another big platform for machine learning are Google Cloud and AWS, they both offer services that will facilitate the cloud hosting of our models. They provide machine learning as a service tools for both data processing and actual machine learning models themselves. We have not looked into using any of their data tools yet, but we believe their automated data processing tools may be useful for us next semester when we want to get our model running on AWS and have the entire data ingestion, processing, and model prediction process streamlined. As for this term, our main focus is on creating models that will yield at least 70% of accuracy.

One of the weaknesses of the proposed solution is that we don't take into account choosing a model based on how quickly it could be retrained and updated in a production environment. Our client would ideally like the model to be updated weekly, if not daily, given new data. At the moment, we are choosing a model just based on prediction accuracy, but if we think about the end goal, to have a model incorporated within the WDesk app, we may have a model that takes too long to retrain and update. One way to overcome this is to ensure that our model will have access to a lot of processing power so that if we use a computationally intensive model like an SVM, the training time will not be as big of an issue in a large-scale data production environment.

Another issue that we've had in the first semester is not having access to enough data. In our proposed solution, we assume that we can go through an iterative process of training, testing, tweaking, and retraining a model. However, this assumes that we have a sufficient amount of data points to work with. However, our client has said that he would be able to provide us with more data points over the summer so that we can overcome insufficient data challenges.

2.6 Technology Considerations

We are planning on using primarily open-source tools and libraries. For creating our models we will be using scikit-learn and keras. Eventually we would like to put our development environment in a Docker container so that any deprecations of future versions won't affect the model.

2.7 Safety Considerations

We need to ensure all data is anonymized so that there is no breach of privacy.

2.8 Task Approach

One of the most important aspects of our project is the model to make help article recommendations. Thus, we will be spending the majority of this first semester researching and implementing different models to try out with our data. This will also involve experimenting with different ways to generate features and testing parameters for our models. We will work closely with our client for model selection and tuning based on the needs of the final product.

To start preparing our data, we took a folder of over 1000 csv files on user behavior and condensed them into one dataframe that information on the user, the date, the actions the user took while using WDesk, and the help article they found useful based on this action sequence. The help article is the dependent variable we are trying to predict. The user actions each correspond to some string label as do the help article titles. For ease of working with data, we mapped all of these strings to a dictionary of ints. This process is contained in a python file on a our github, but we plan to tweak it so that we could eventually automate this data preparation task and put it on AWS.

After the data preparation, we begin creating features. By creating text features we have gone from 15% to over 50% accuracy with our random forest model. Thus we have been creating text features, then feeding them to a neural network or random forest model, testing the model and then looking at the breakdown of accuracies for each class label to tweak our features and model. For example, if we find that our model is very good at predicting a certain article that appears a lot, we may have go back and create features that weight certain action sequences more for help articles that are predicted less often. Using confusion matrices and looking at the precision and recall values for each class/help article has been a good guide for this process.

Since we have over 170 help articles to predict and only 690 data points, we have also grouped the help articles by common topics to try to increase the prediction accuracy of the models. Once we have access to more data this may not be something included with our final model, but for now we believe that providing a user with the top five most likely useful articles is better than providing them with one article that is useless.

As for model selection, we have been using random forests and neural networks mainly because their baseline prediction accuracy with our data and little feature processing was higher than other models. We are continually trying to increase the per class accuracy by rigorously tuning and tweaking our features with these models. Model training is an iterative process that involves going back and forth between tuning model parameters, training, testing, and then going back and starting over if we are unsatisfied with the results. See our testing diagram in the below section for a visual description of this process.

Once we have selected a model, then we can work on creating the data pipeline and getting it and the model to work smoothly on AWS, and eventually make the model update given new data. This will involve having to build an entire automated data processing, model updating, training, and prediction process. This is something we will work on primarily in the second semester of this project.

2.9 Possible Risks And Risk Management

There is a risk that we may not finish our project. To avoid this we need to make sure we have a realistic project plan and schedule that we follow. We need to ensure we give ourselves enough time to research and execute our tasks. We also need to make sure that if we are not getting good accuracy with our models that we reach out to the faculty advisor and client for help. There is a risk that one of the open source libraries that we use may no longer be available or supported through the year. To mitigate this we need to research several machine learning libraries for use with python so that we have alternative options if the problem arises.

2.10 Project Proposed Milestones and Evaluation Criteria

Milestones	Description	Planned Date *(tentative)

Research/Get experience	1.Scikit-Learn 2.Keras with Tensorflow	2018-02-05
Experiment with Classification and Recommendation Models	1. Test out previously used recommendation algorithms and feature generation models with our data. 2. Ensure we understand how the models work.	2018-02-28
Get at least 70% accuracy on at least one model	1.Develop our own models based on the data and preexisting ones to predict help articles.	2018-03-15
Testing	1. Test algorithm and tune more as needed.	2018-03-31
Run model on AWS	1. Get model ready to deploy and run with data on AWS.	2018-04-20

Table 1

2.11 Project Tracking Procedures

We will use Trello to assign tasks into two week sprints. These tasks will be assigned according to team roles, what has been accomplished in the past weeks, and our goals moving ahead. We will try to split up tasks such that they can be completed within a two week time period. Our team will meet weekly both as a team and with either our client or faculty advisor to make sure we are on track with our sprints and deliverables. We will be utilizing a gantt chart to track our schedule and progress.2.12 Expected Results and Validation

Research/Get experience

1. Learn how to create basic ML models with scikit-learn and Keras.

Experiment with Classification and Recommendation Models

1. Elicit input from client and faculty advisor in researching how different machine learning algorithms work and which ones we may like to prototype with our data.
2. Select different models for final prototyping and get preliminary results using our data.

Get at least 70% accuracy on at least one model

1. Continue testing and tuning models until we can get 70% accuracy on a model.
2. Elicit feedback from faculty advisor and client on how to best tune models to achieve this threshold.

Testing

1. Continue testing and tuning model as client requests.
2. Configure model to update given new data.

Run Model on AWS

1. Get model to run on AWS.
2. Test functionality of model on AWS.

2.12 Standards

We are working with a company for our project, and being a software project there are no required labs or physical equipment we must use. Since the work is done for a company there are no institution standards, there are only the ones Workiva has set-- the primary one being the non-disclosure agreement signed at the beginning of the semester. In this NDA we are legally required not to talk about the details of the project outside our development environment, that means not sharing project information with the public.

For standards in terms of terminology we will be referencing the IEEE Standard Glossary of Image Processing and Pattern Recognition Terminology. This standard defines common terms along with definitions in the area of image processing, pattern recognition, and machine learning. Many of the terms and definitions in the glossary will be useful for our project. The standard definitions allow for knowledge to be shared consistently among team members, client, and advisor. For example, feature is a term that is in the glossary. While many people from a computer science/ML background may immediately know this term, an individual from a statistics background would only know feature as a parameter. We hope that using standard terms will allow our project processes to be conveyed clearly to all interested parties.

The IEEE Standard 730-2014 Software Quality Assurance Processes would be helpful in assisting us to evaluate the quality of the software side of our project. It will also be helpful in allowing us to determine whether the recommendation software we are creating meets our client's' expectations and allow us to do acceptance testing for our model and recommendation system before we deliver them to the client.

Working with Workiva most of the standards applied to software companies are present in the project as well. Documentation is important as it helps track any progress we have made and also allows overseers of the project see where we stand; therefore, every progress made is recorded and well documented Deadlines should be met according to the timeline, proposed goals should be done to meet with Workiva's standards. Lastly, one of the most important standards in the software industry would be that all code must be original and come from the team, there should no plagiarism nor should we be taking credit for work that isn't our own.

2.13 Test Plan

We will be testing the models throughout the process of development. We will use python for data processing, and for creating and testing our models. Therefore, we will use PyCharm, Ipython Notebook and other Python IDE to develop our project. Our test cases will then be able compile and execute in any of these environments.

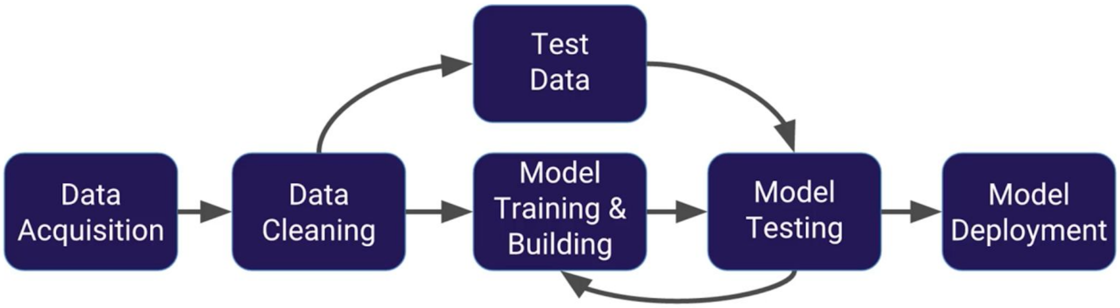

Figure 1: Testing Process Diagram

The process of developing our prediction models involves the following steps.

- In the future, user activity will be monitored and stored. This data will be processed in real time as the user of Wdesk uses the application. However, at the moment we are using a static dataset and do not need to worry about real time data processing and storage.
- Our model will take in raw data and process it as required by the prediction algorithm we want to use, i.e generate features, create a padded sequence for a neural network etc.
- The processed data will be tested that it still holds value, i.e. if we map strings to ints, we test that this is actually the case.
- The generated features will be tested for their value using cross-validation.
- The model will be built and trained using training data.
- The model will be tested using our testing data. Based on the outcomes of the testing step, we may have to go back to the data processing phase and repeat until we get over 70% accuracy with a given model. See function and non-functional testing for more specifics on model testing.
- After we get the desired performance of our model we can deploy it for use in production.

2.13.1 Functional Testing

Due to the nature of our project we will not being overly reliant on unit tests. We will use visual and statistical testing methods to gauge the performance of our data features, model hyperparameters, and model performance.

- Test to make sure the data is not incomplete or mislabeled.
- Use F1 scores, ROC curves, and confusion matrices etc. to gauge the accuracy and prediction competency of our models. This involves testing to ensure a model has over 70% accuracy.
- Use cross-validation to test and tune features and model hyperparameters.

2.13.2 Non-Functional Testing

The following list includes testing for non-functional requirements

- Performance: Test that the prediction model should be able to predict the helpful article within several seconds of the initial query.

- Scalability: Test that the model should eventually be able to process any size of data
- Extensibility: Test that the model should be able to compile and run on AWS.
- Usability: Acceptance testing by the client would check that the model we develop should be easy to understand and use for Workiva developers so that they can integrate it without any problems with their current application.

3 Project Timeline, Estimated Resources, and Challenges

3.1 Project Timeline

Figure 2: Project Timeline

The given Gantt Chart shows our projected timeline for both semesters. We decided to make this chart as detailed as possible as it would help us to track every tasks and deliverables effectively.

On the first semester, we would spend about a week to gather requirements and decide team roles & responsibilities to set us in the right foot. From thereafter, we start developing models which is divided by two parts. The first part is when the model is trained with self generated data and the second starts when Workiva starts to give us real data periodically for us to train our developed models. Between trainings and results gathering, we would do parameter tuning and further improvements or additions to our models.

On the second semester, we would spend a week to conclude our models design and implementation that we have been doing throughout summer break. Minor changes and improvement might be done during that time. After that, we would move to porting it to AWS which is a step to make our project ready for production.

3.2 Feasibility Assessment

The team has worked together in the past in previous classes and has always completed the work assigned. We are all enthusiastic about machine learning and we got one team member who has previous internship experience using machine learning. Others in the team have also had internships in multiple companies and have a broad perspective of what it is to work in the “real world”. The project proposed by Workiva is feasible as long as we can generate good features for our models given the data. They are asking for a article suggestion model for their WDesk application depending on how the user interacts with the program. As long as we process our data well, and carefully select our model with input from our client, we believe this project is feasible.

One risk that we realized is that at the end of this project, the product that we have is not highly reliable to be put out for production where it has low percentage of giving the right help article. If such case happen, all of the resources that we have used would be a waste. It would not cost Workiva much as they could easily hand the project to full-time employees but, it would cost us that we spent a year with a product that failed to reach production.

3.3 Personnel Effort Requirements

Task	**Estimated Hours**	**Description**
Research	120	The project consist of more than just machine learning. As there are many areas to focus, each team member has been assigned a specific topic: AWS, Data Management, Machine Learning and much more.
Architecture Design	350	The core fundamental of any system is its design. A good architecture will make the development of the project much easier.
Machine Learning Models	300	In this task we will focus on training our models and see how they perform, comparing the models and choose the best performing one.
Data Processing	100	Using the data provided to us by Workiva we will test which formats and features will work the best for each different model.

Testing Functionality	150	Testing will happen throughout the whole project. We will do this in sprints checking that every component works well.

Table 2

3.4 Other Resource Requirements

- Amazon Web Services
- Datasets of User Interaction on Wdesk
    - Large training set
    - Intermediate testing set for comparisons
    - Smaller testing set

3.5 Financial Requirements

Most of the resources that we use and need are open sources materials and softwares. Moreover, our client has not emphasized any budget or materials he would need to provide us at a cost. In the future, most resources will be provided by the client.

4 Closure Materials

4.1 Conclusion

Throughout these two semesters we will work on creating a predictive model that can provide a help article recommendation based on user app-interaction history. Thus far we have done significant research into model selection and feature generation for our data. We have also considered how we would test our models and features, and gauge their efficacy. With our current results, we do think that using text features and neural networks may be a promising area of focus given our discussions with our faculty advisor, client, and our own research into these topics.

We believe our testing approach, as noted in the Testing Process Diagram, is well-suited for the project. This will involve a circular process of data cleaning, feature generation, model selection, model tuning, and model testing. We will continuously repeat this process until we can gain at least a 70% accuracy rate or higher on a model. This testing approach was created after researching how Data Scientists and Machine Learning Engineers typically structure their data analysis projects.

In the next semester, we will work on automating our data processing, feature creation, model updating and training process. We will be using AWS and Lambdas to complete this automation process. We will also need to refactor our python code and modularize it so it can be deployed as part of a larger automation process.

To achieve such a goal, the plan is divided into 2 phases.

First Semester (Phase 1): In phase 1, we would first gather information and requirements from the client. The client will also give us data for us to work on and progress towards designing a model for machine learning. From thereafter, we would constantly do testing, analyzing, improving, parameter tuning and learning of the models that we have designed.

During this phase, the goal is to come up with the best possible model that generates the most accurate results.

Second Semester (Phase 2): Continue working on model refinement and tuning as needed. Get model to deploy on AWS and re-engineer it to be able to update weekly given new data. Ultimately, we will need to create an automated process on AWS for data ingestion, data cleaning, data processing, model training and tuning, and model prediction.

4.2 References

Amazon Web Services (aws) - Cloud Computing Services. https://aws.amazon.com/

Comma-separated Values. https://en.wikipedia.org/wiki/Comma-separated_values

Jiacheng, Xu. "Family Shopping Recommendation System Using User Profile and Behavior Data." *arXiv preprint arXiv:1708.07289* (2017).

KDnuggets. (n.d.). Retrieved from https://www.kdnuggets.com/2017/10/linkedin-personalized-recommendations-photon-ml.html

Python Data Analysis Library. https://pandas.pydata.org/

Scikit-learn: Machine Learning in Python - Scikit-learn 0.16.1 Documentation. http://scikit-learn.org/

Singhal, Ayush, Pradeep Sinha, and Rakesh Pant. "Use of Deep Learning in Modern Recommendation System: A Summary of Recent Works." *arXiv preprint arXiv:1712.07525*(2017).

Tensorflow. https://www.tensorflow.org/

Zhang, Shuai, and Lina Yao. "Dynamic Intention-Aware Recommendation System." *arXiv preprint arXiv:1703.03112*(2017).

4.3 Appendices

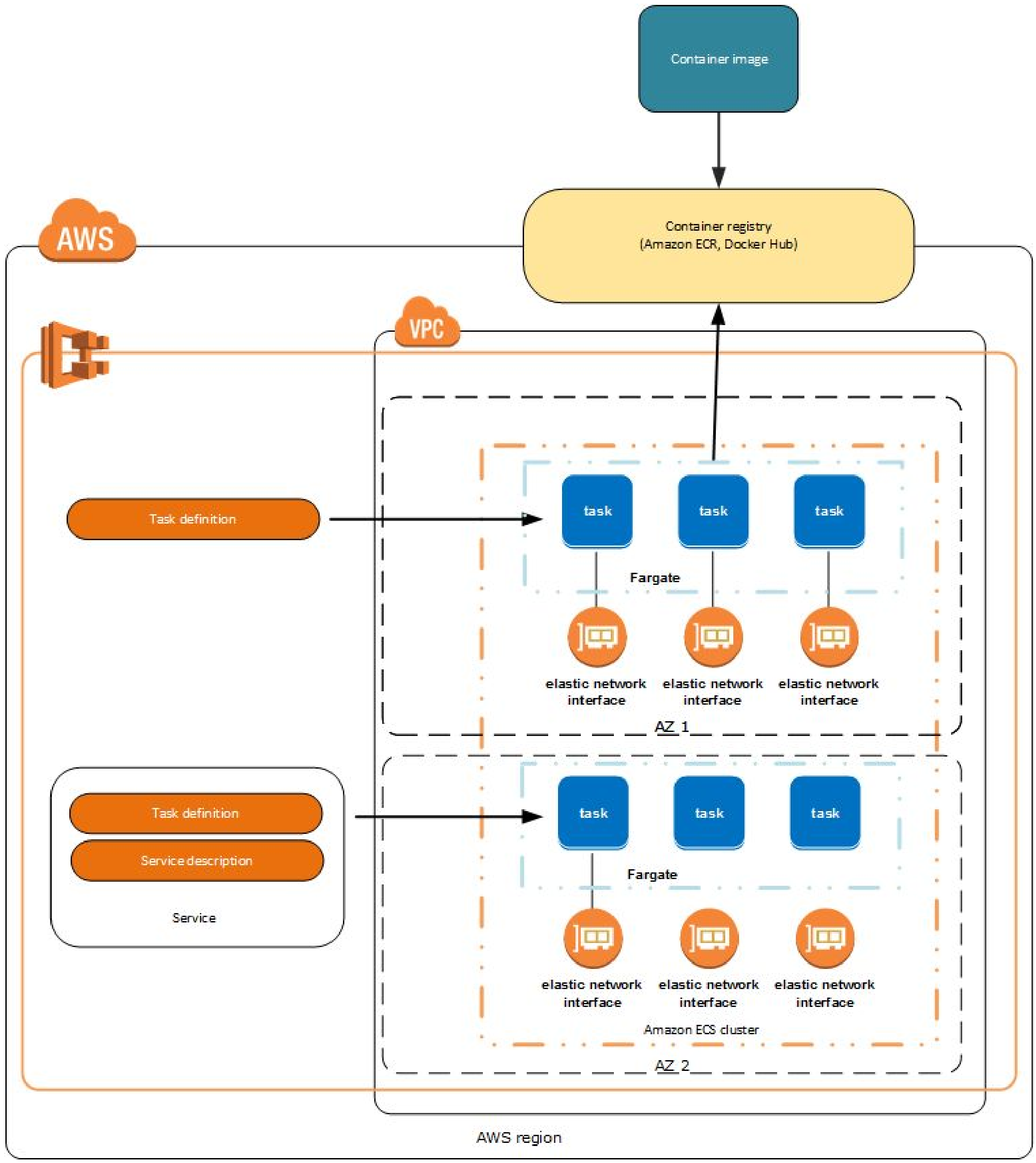

Figure 1.2: The structure of AWS we may use.

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html