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

DESIGN DOCUMENT

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: March 2/Version 1

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

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 requires 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 and Project 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 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 lot of time and money.

1.3 Operational Environment

This will not be a stand alone application and will be dependent on the virtual environment provided by Workiva. Our application will eventually 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, to ensure a lower cost, and give a 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 Uses

Our end product is a model that will be used by developers at Workiva to integrate with the wDesk app. However, the model will directly impact how Workiva customers use and interact with the wDesk app.

1.5 Assumptions and Limitations

The only cost factor for our project is Amazon Web Service licence which our client will provide to 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 best performing model. 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 Deliverables

At the end of the next 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 deliver all project related documents that include design, code, project architecture documents and potential documents that include instructions on how to use our models.

2. Specifications and Analysis

2.1 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 have not been able to test them on real user data because we do not have the complete user data and help article labels for training yet.

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.

2.1.1 Functional Requirements

- The recommendation model should be able to make recommendations for help articles and display the recommended article ID.
- The model must be able to run on AWS.

2.1.2 Non-Functional Requirements

- The recommendation model should be optimized to run as quickly as possible.
- The recommendation model should be scalable with large and real-time data.
- The model should be written in Python.

- The model should be easily readable by Workiva developers for future integration with WDesk app.

2.2 Design Analysis

To begin our design analysis we researched various supervised machine learning models. After input from our client and faculty advisor, we also researched neural networks and time series models because we are ultimately working with time series data. At the moment our team is subdivided into two sub-teams. One team will be creating machine learning models using random forests and ARIMA models. The other team is experimenting with neural networks and markovian networks. The selection of these models came after research into supervised learning classification and working with time series data. Even though random forests are used for classification, we also wanted to try working with classic machine learning models and see how well they could perform against neural networks given good features. Though we don't have training data to give preliminary results yet, we think the LSTM neural networks may be a strong performer given they can work well with time series data and text data. This means they will work well if we process our data like a time series model, or if we process our data in terms of text features.

For generating features, we plan to use text feature generation methods including bag of words and n-grams. With these features we won't necessarily need to use time series models but can still get good predictions. We decided to use text feature generation methods with our data so it could better capture important user action sequences.

3 Testing and Implementation

3.1 Interface Specifications

We will use python for data processing, for creating and for testing our models. Therefore, we will use PyCharm, Ipython Notebook or other Python IDE to develop our project.

We will use the following python libraries for model creation and testing: Scikit-learn, Keras with Tensorflow and Pandas.

We will mainly test if our machine learning models can predict the testing data correctly after being trained. We will select the best performing model as our final model.

If our final model finally gets over 70% accuracy, we will set it up on the AWS cloud and test.

3.2 Hardware and software

CSV: In computing, a comma-separated values (CSV) file stores tabular data (numbers and text) in plain text. We will use python to parse the data from CSV files given to us for testing.

We will test if our data is processed to specifications by using a Python script before analyze it.

> Scikit-learn: Scikit-learn (formerly scikits.learn) is a free software machine learning library for Python. It is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. We will use the open source code and models from Scikit-learn to create our models, e.g. 'RandomForestClassifier()'.
>
> TensorFlow: TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and also used for machine learning applications such as neural networks. We will use it to build our deep learning models.
>
> Keras: Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow. It is designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. We will use it to support our deep learning model for TensorFlow.
>
> Pandas: Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. We will use Pandas to parse our data from CSV files.

Since they are all open source libraries, each version update may cause methods being changed. We will test if every method works as what we expect and develop in docker containers to avoid version conflicts.

> Amazon AWS: Amazon Web Services (AWS) is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis. We will upload our final project to AWS cloud for scaling.

In conclusion, we will test if we input our data and use open source code correctly. We will test our models use scikit-learn features. We will also test if our code can run and scale on AWS as well.

3.3 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.

I. Test to make sure the data is not incomplete or mislabeled.
II. 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.
III. Use cross-validation to test and tune features and model hyperparameters.

3.4 Non-Functional Testing

The following list includes testing for non-functional requirements

I. Performance: Test that the prediction model should be able to predict the helpful article within several seconds of the initial query.
II. Scalability: Test that the model should eventually be able to process any size of data
III. Extensibility: Test that the model should be able to compile and run on AWS.
IV. 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.

3.5 Process

Our design process will include several different models. However, the design and implementation details of all models is the same, the only difference is the features involved, the data input, and the algorithm.

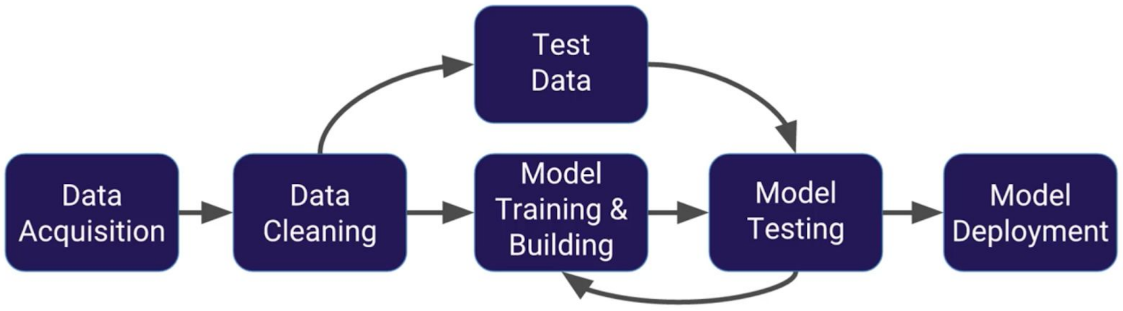

Figure 1: Design Diagram

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 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.
- After we get the desired performance of our model we can deploy it for use in production.

3.6 Results

3.6.1 Implementation Issues and Challenges

During the initial development phase of our project, we have had some issues with merging code with different versions of python, scikit-learn, tensorflow, etc. We plan to overcome this problem through creating a Docker container and having everyone work within the same environment.

Another issue we potentially foresee is training models with a small amount data. At the moment, the total number of data points we have is 1200, and this may not be sufficient for a neural network model. We want to overcome this challenge by focusing heavily on creating good features from our data. We will also need to do more research into using neural networks with smaller datasets.

Our client would like our final model to scale well and run on AWS. Since we don't have experience with building models that can scale and update daily, this will pose a significant challenge to us as we begin this process in the second semester.

4 Closing Material

4.1 Conclusion

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. To best approach the project, we are subdividing into two teams who will focus on specific models and test their performance. Though we don't have any initial results yet, 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 design approach, as noted in our Design 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 Design Diagram and approach was created after researching how Data Scientists and Machine Learning Engineers typically structure their data analysis projects.

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

James, Gareth et al. *An Introduction To Statistical Learning: with Applications in R*. Springer Science+Business Media. 2013.

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

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

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

4.3 Appendices