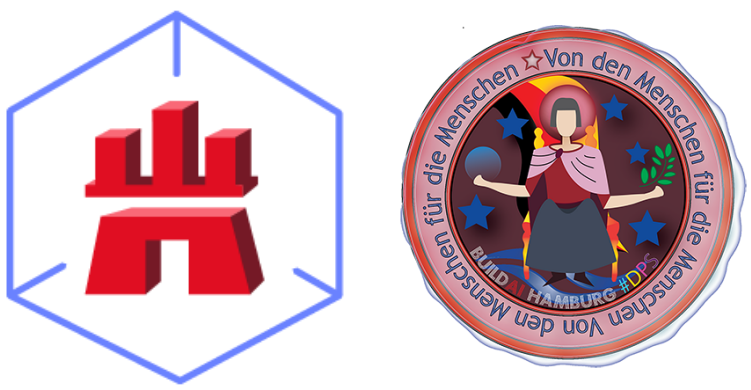
Digital Product School by UnternehmerTUM

Artificial Intelligence Engineer and Software Engineer

Overview:

Digital Product School is an amazing program fostering digital product development to solve real world problems in interdisciplinary teams. My application was accepted in DPS as an AI Engineer for batch#6 but joined batch#11 due to unavoidable circumstances. In Batch#11 at DPS (8 Sept 2020 - 27 Nov 2020), I was an AI Engineer of team BuildAI and our stakeholder was LSBG. We at team BuildAI were responsible to innovate and work on the problem statement - "WIBG: If every citizen could participate digitally in the development of possible variants and concepts for a planning area of a city?" [pdf] | Stakeholders: [Landesbetrieb Straßen, Brücken und Gewässer, Government of Germany]. In this blog, let's discuss about my work efforts in team BuildAI at Digital Product School. We follow : EXPLORATION → IDEATION → PROTOTYPING → VALIDATION. We were getting started with icebreaking and it was amazing to meet digital product distruptors from around the world. I also had the priviledge to design the team BuildAI insignia with Adobe Illustrator:



My team comprised of 1 AI Engineer (me), 1 Product Manager, 1 Interaction Designer and 2 Software Engineers. One of the team member (Software Engineer) left the program due to unavoidable circumstances and I had to fill up the void in my team. I was working overtime both as an AI Engineer and as a Software Engineer for which, I was later honored with dual track certification (AI [certificate] and SE [certificate]) at Digital Product School, Unternehmer Technical University of Munich. Here are my amazing and humble mentors.

Digital Product School is an agile experience with amazing set of digital tools to solve problems at hand. We used [Miro, Mural, Google Docs, Figma, WeTransfer, Github, Wonder/Yotribe, Jamboard, Calendly, Loom, Menti, Typeform, Slack, Asana, Jira, MetroRetro, kahoot, overflow, notion] etc.

We experienced DigiCave at Hamburg, Learned Ideation, Explored Master Portal, DIPAS. We wanted to solve the time-consuming, analog information collection process for LSBG facilitators who run in-person project-planning workshops, because we can offer live-audience digital data collection and analysis with our BuildAI digital participation process tool. We are developing to support, Von den Menschen, für die Menschen in the city of Hamburg!

There was a Hackathon at DPS in the first 2 weeks and our team won the Hackathon (#solutionhackz Winner) [our pitch]. I enjoyed my journey throughout the initial weeks of DPS and learned Figma, did some Adobe After Effects VFX in our pitch as well as Adobe Premiere Pro in our Technical Video. Every DPS batch has a batch song and our batch song was - The Killers - Mr. Brightside (Official Music Video).

My role:

○ Artificial Intelligence Engineer:

As an Artificial Intelligence Engineer:

• Deployed BERT Sentiment Analysis Flask API written in PyTorch to Google Cloud App Engine with Docker.

✓ 4 Pillars of user centric AI products: Collaborative, Interactive, Personalized and Keep Learning.

We explored Fairness with AI - [PAIR], [Google Design], Exciting AI projects - [runwayml], [TensorFlow Serving], [ml5js], [kubeflow for kubernetes], [what-if tool], [Teachable Machine], [Talk to books], [featurestore], [Random 3D City], [Cesium 3D tiles].

I made a BERT Sentiment Analysis Flask API and deployed it to Google Cloud App Engine using Docker. The BERT model was written in PyTorch and was trained with a 50K IMDB Movie sentiment dataset achieving an accuracy of 91%.

To call the API:

```
https://buildai4sentiment.ew.r.appspot.com/predict?sentence=I like it!
```

The API response:

```
{"response":{"negative":"0.0013362169265747 07","positive":"0.9986638","sentence":"I like it!","time_taken":"0.8116645812988281"}}
```

The sentence, ‘I like it!’ is a positive sentiment and the API responses with a positive score of 99.86%. The instructions to deploy the dockerized API to Google Cloud:

```
$ gcloud init
$ gcloud config set app/cloud_build_timeout 3000
$ gcloud app deploy --log-http
```

BERT : Bidirectional Encoder Representations from Transformers

Resources : (paper) | article, article 2, article 3

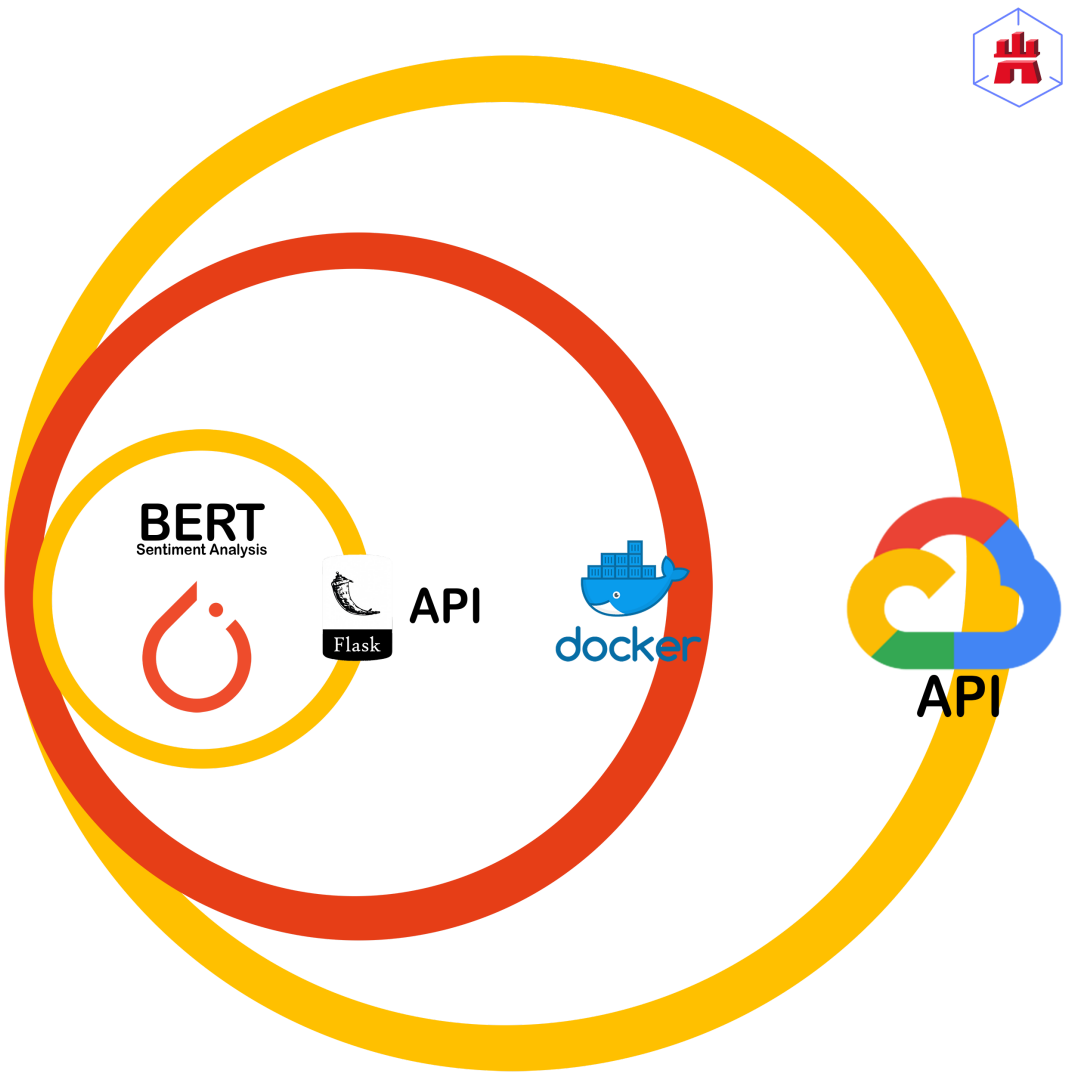
How does BERT works?

BERT’s key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT’s goal is to generate a language model, only the encoder mechanism is necessary.

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads

the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

• I was also working on Automatic Report Generation with Natural Language Generation (NLG).



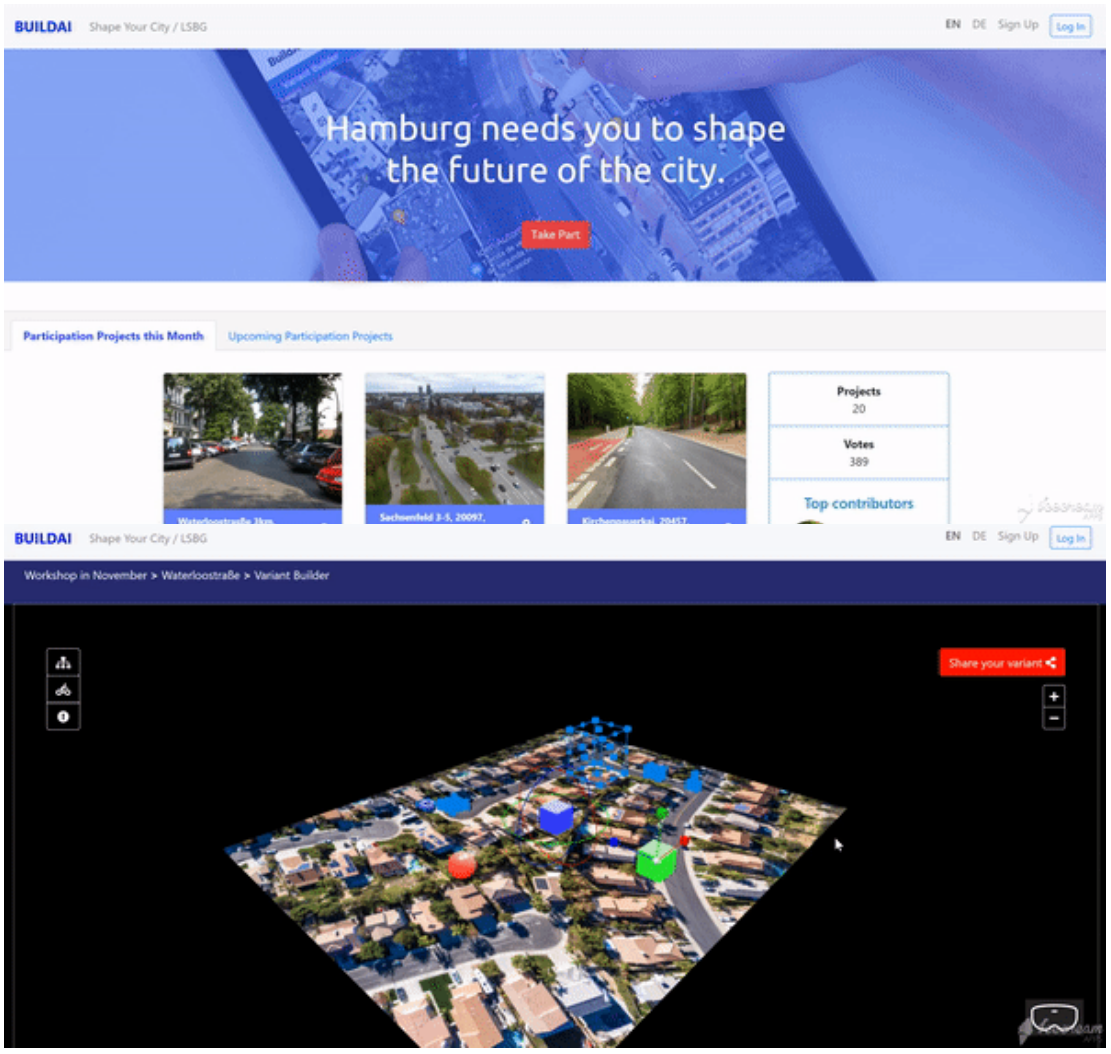
○ Software Engineer:

As a Software Engineer:

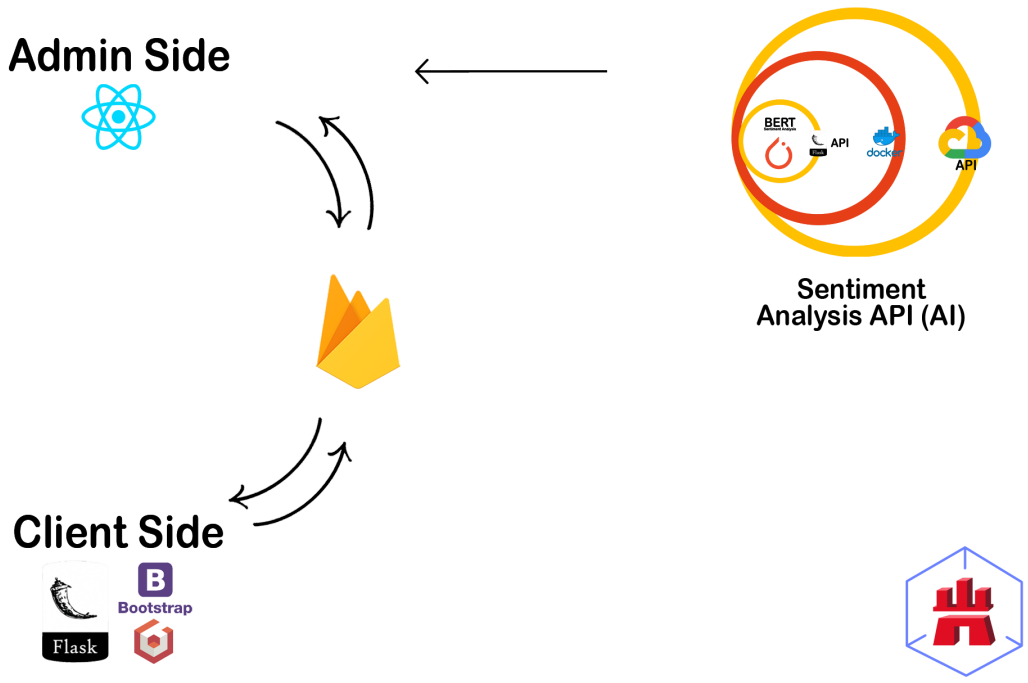
• Built client side web-application with Flask, Bootstrap and React JS.

• Built variant builder 3D Mapping with CesiumJS, TerriaJS as well as variant builder platform with BabylonJS.

buildai.herokuapp.com is the client side project gallery / web application, built as a Flask server and deployed to Heroku. The frontend is formatted with bootstrap and the backend has jQuery. buildai.herokuapp.com/project1 is a project page from project gallery and is about the "Waterloostraße Project" in the City of Hamburg. The web application is responsive and is suitable to be used via laptop, mobile and/or ipad. Citizens can participate in the development work in the City of Hamburg by proposing their own variant of the project and sharing their views/comments as well. The project page holds information about the project - history, motive, video, images etc and gives an extensive overview of the project. These infromations are fetched from Firebase. Citizens can share their opinion/views regarding the project in the comments section which are then stored in Firestore → Firebase. Citizens can propose their own variant of the project by clicking on "Go to Variant Builder".

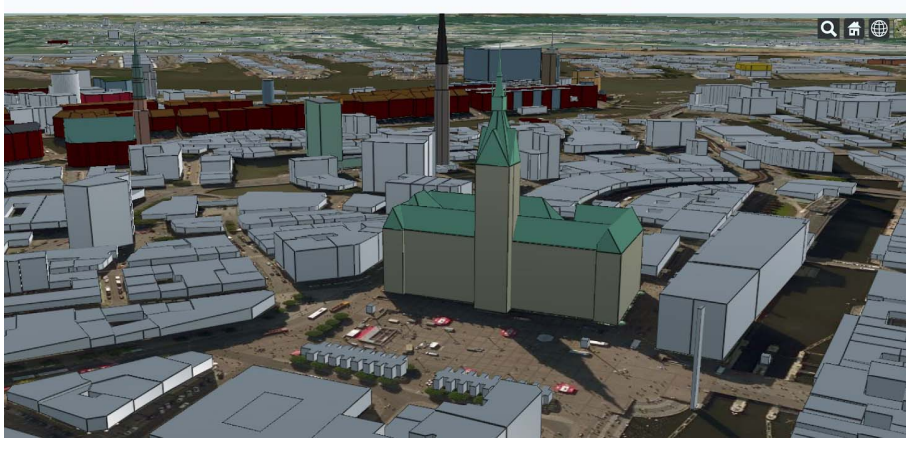


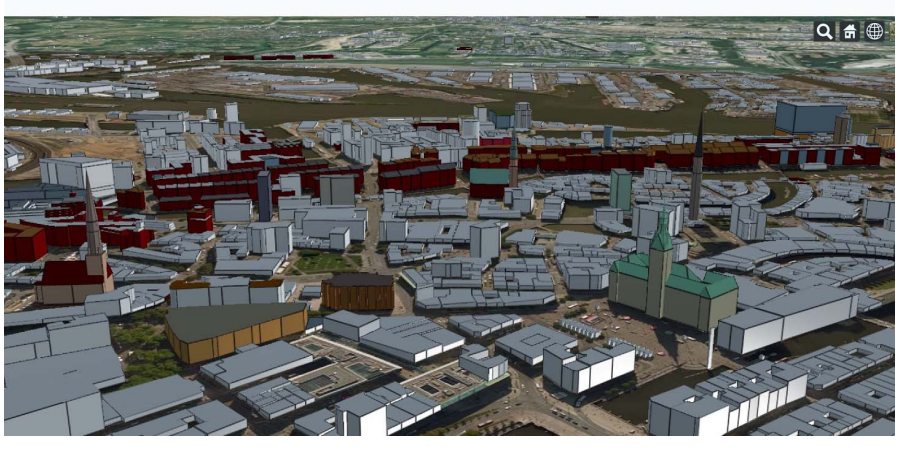
The variant builder is built with BabylonJS and has a VR mode as well. Citizens will be able to build their own variant of the project and submit it to the city administration. Once data is sent to Firebase, the admin side fetches the data from Firebase and calls the sentiment analysis API. The comments in the project are then analysed for sentiment with our API in the admin side and an overall sentiment of the project is found out. There are several analytics - dashboard, pie charts etc in the admin side to give an overall impression of the project from the citizens of Hamburg. The admin side is developed with ReactJS and Material-UI. The entire architecture of the system:



○ Experimentation:

Tech Stack: Python, PyTorch, Google Cloud Platform, Docker, Flask, Bootstrap, ReactJS, Material-UI, node.js, BabylonJS, CesiumJS, Firebase, CI/CD, CRUD, POSTMAN, Exploratory Data Analysis (EDA), Git.
I experimented with Cesium JS for 3D mapping of a project area in Hamburg. The 3D OSM building layers helped us render the city of Hamburg in 3D. Cesium JS is quite a powerful tool with an amazing set of features, to list a few - underground visualization, object animation etc. Here are some 3D OSM buildings around St. Peter's Church area in Cesium JS which could be imported in "Variant Builder":





I worked on another 3D visualization library called Terria JS which is built over CesiumJS and has a lot of more amazing features than its base library. TerriaJS is opensourced and loads gltf/glb 3D geotagged buildings and objects with better visual textures. TerriaJS is promising and has amazing useful features inbuilt, to name a few - Story Mapper, Distance Measure, Before/After slider etc. The possibility of having a true 3D variant builder by mapping TerriaJS and BabylonJS by GeoJSON is closer than ever.

Final:

Product Video : BuildAI Hamburg Product | **Solution Video** : BuildAI Hamburg Solution | **Technical Video** : BuildAI Hamburg Technical
I really enjoyed my time at DPS and very grateful to my mentors and batchmates | Signing off! amartya.saikia@dpschool.io

Memories