Octo Consulting

GSA AI Challenge Methods Writeup

# Solution Overview

## Broad Summary

Our solution for the GSA AI/ML Challenge is an easy to use web application. A user simply navigates to the webpage, uploads a EULA pdf or word doc via drag and drop or via the upload button, the document is processed using our AI/ML pipeline, and then the user is presented with a results page which contains the original document and cards containing the clause, the percent acceptable score, and the classification of acceptable or unacceptable. The user can click on cards and be taken to where that clause appears in the document for further inspection. Refer to the “Demonstration” section of this paper for screenshots or watch our demonstration video for a full presentation of our solution. This solution makes it easy for the user to identify potentially unacceptable clauses in a fast and intuitive fashion, thereby reducing the amount of time needed to analyze EULA documents. Our solution utilizes a pretrained Bidirectional Encoder Representations from Transformers model for training on the provided dataset. Results are promising, with a weighted F1 score 0.87 and a Brier Score of 0.09. With a more balanced and larger dataset, these results could be improved upon to create a state-of-the-art model that dramatically reduces the amount of time a user spends analyzing EULA documents.

## Solution Architecture and In-Depth Description

The user interface of our application has two separate parts. The first is an angular application which uses components to host our html pages. The second part is a flask server which handles REST API requests and starts the parsing and classification of the EULA. The flask server is run through a docker container, which provides easy set up and access. Set up instructions can be found the readme file and on the GitHub wiki page. The picture below shows the architecture diagram for our prototype solution.

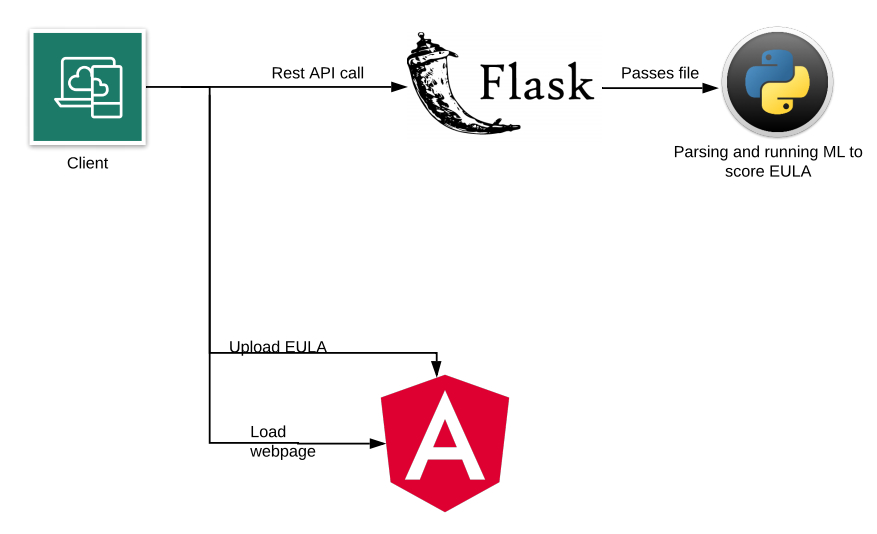


Figure Architecture Diagram

The angular application has three main pages: a file uploader, a waiting screen, and a document viewing screen. The file uploader allows a user to upload a file to the site, which then sends a POST request to our flask server. The flask server checks the file for the correct extension before allowing the file to be processed by the model. Our model is a neural network that performs sequence classification, and our model and associated software is described in more depth later. The server then receives a json file of clauses and scores that the model returns. Meanwhile, the user is shown a waiting screen that gives time for the model to process the EULA. After the circles finish spinning, the user can click on the document viewing screen and see their uploaded document. On the side will be cards with the phrases that were given back to the app by the model, either classified as acceptable (green) or unacceptable (red) along with clause scores. If a user clicks the card, the phrase will be moved into view and highlighted.

# Data

The dataset AI\_ML\_Challenge\_Training\_Data\_Set\_1\_v1.csv provided by GSA was the only dataset used for training. As noted in the challenge requirements, this dataset is limited in size and is not an ideal source. It is an unbalanced dataset, meaning that the percentage of acceptable and unacceptable clauses is not equal. Also, the training example clauses vary significantly in length, with some examples being a single words and others being large paragraphs. In order to overcome the limited training data, each provided training clause was separated into individual sentences. This means that the training set increased from dramatically from 7,977 examples to 23,384. Additionally, the lengths of the original clauses were widely varied, anywhere from one word to a large paragraph. Separating clauses out into individual sentences makes the data more standardized in length, which helps with the training process.

# Methods

## Data Pre-analysis

A normalized Jaccard similarity score between the accepted and unaccepted clauses that takes duplicate words into consideration is 0.80. This means that around 80% of the words that appear in the entire body of accepted clauses also appear in unaccepted clauses. This is a huge overlap that makes distinguishing clauses extremely difficult. This high similarity score shows that a bag of words type model that does not take ordering of words into account is very unlikely to be successful. For this reason, we used the Bidirectional Encoder Representations from Transformers (BERT) technique developed by Google. The vector for "running" will have the same vector representation in the sentences "He is running a company" and "He is running a marathon" in a traditional NLP encoding structure, but BERT will provide a contextualized embedding that will be different according to the sentence structure.

## Document interpretation with OCR

In order to interpret pdf and docx documents, the Optical Character Recognition Python packages pdfplumber and docx were used. These packages convert the entire document into a text format that is able to be further processed.

## Clause Parsing

After OCR is performed, the uploaded documents have been converted into a single long string of words. In order to separate the document into individual clauses for analysis it was decided that the level of granularity would be based on sentences, as mentioned previously. The Python package nltk or Natural Language Toolkit was used to separate the large body of text into individual sentences. Additionally, nltk provides easy to use interfaces for special character removal, tokenization, stemming, and all other NLP data preprocessing steps.

## Clause Acceptability Assessment – Model Building

This problem is a supervised learning problem because the classification values are given. Our approach is to use a neural network in order to train our model to recognize which sentences are acceptable vs unacceptable. The BERT encoder was used to encode the individual clause sentences, and the BertForSequenceClassification 12 layer model transformer was used from the huggingface transformers library with a softmax classifier top layer. This allows for our model to return a classification and an associated confidence in that classification. Our model used an adaptive AdamW optimizer in order to provide the fastest convergence for our neural network. The training set was separated so that 15% of the data could be used as a test set. A combination of techniques including 5-fold cross validation and heat-mapping of model parameters was used in order to achieve a model with the highest accuracy. As mentioned previously, this resulted in a neural network that is able to predict if a sentence is acceptable or unacceptable for GSA with approximately 78% accuracy. Other metrics are provided in more detail below.

# Software

Angular

Flask

Python

The following Python packages were used in this solution:

* Pdfplumber
* Docx
* Nltk
* Numpy
* Pandas
* Pytorch
* Flask
* OS
* Huggingface transformers
* Google\_drive\_downloader
* Sklearn
* Csv

# Demonstration (including data inputs and visualizations)

The demonstration is provided in an attached video. Screenshots of the user interface are provided here. The angular application has three main pages: a file uploader, a waiting screen, and a document viewing screen. The file uploader allows a user to upload a file to the site via drag and drop or by using the upload button and navigating through their file directory.

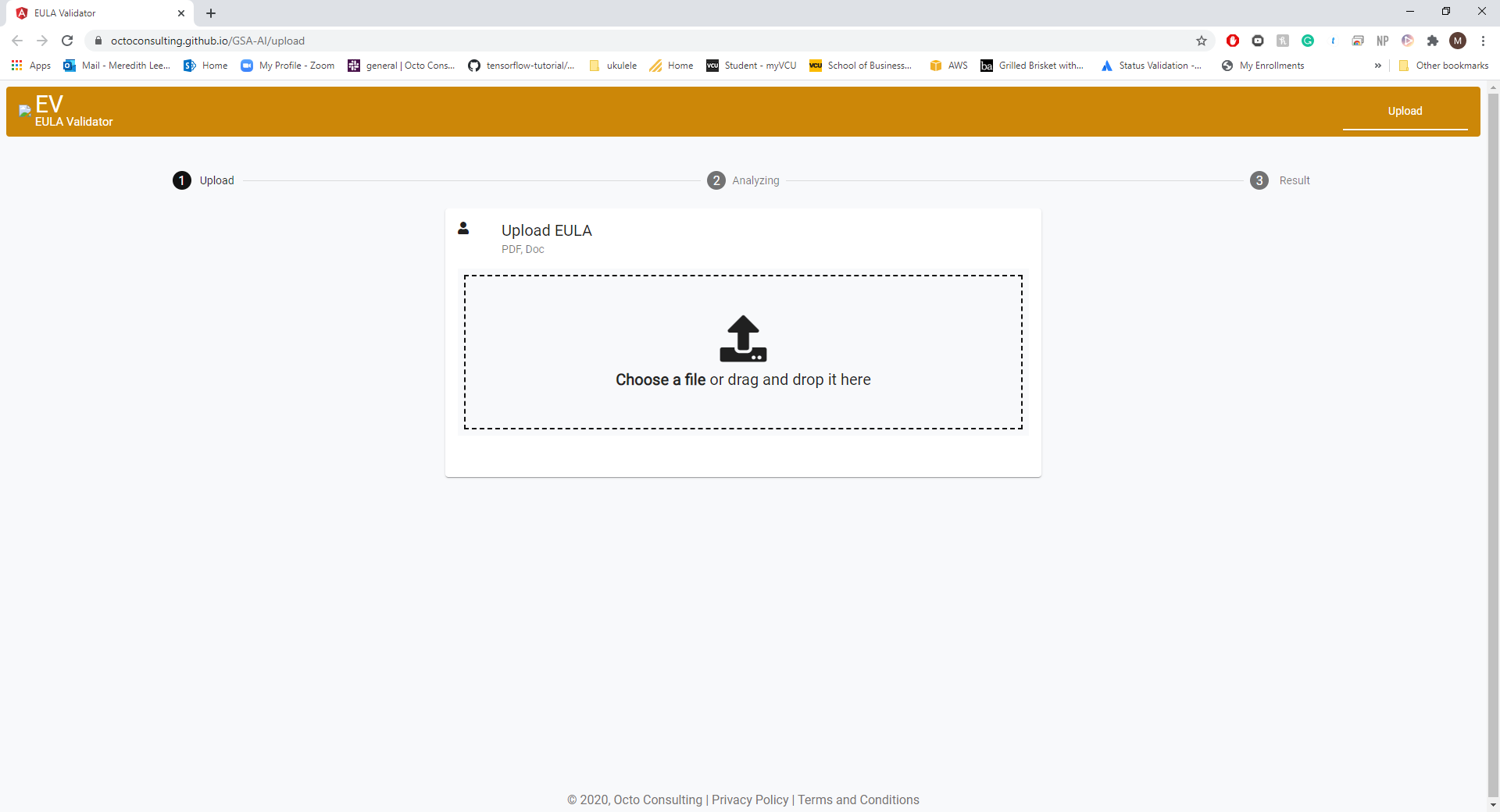


Figure Upload Page

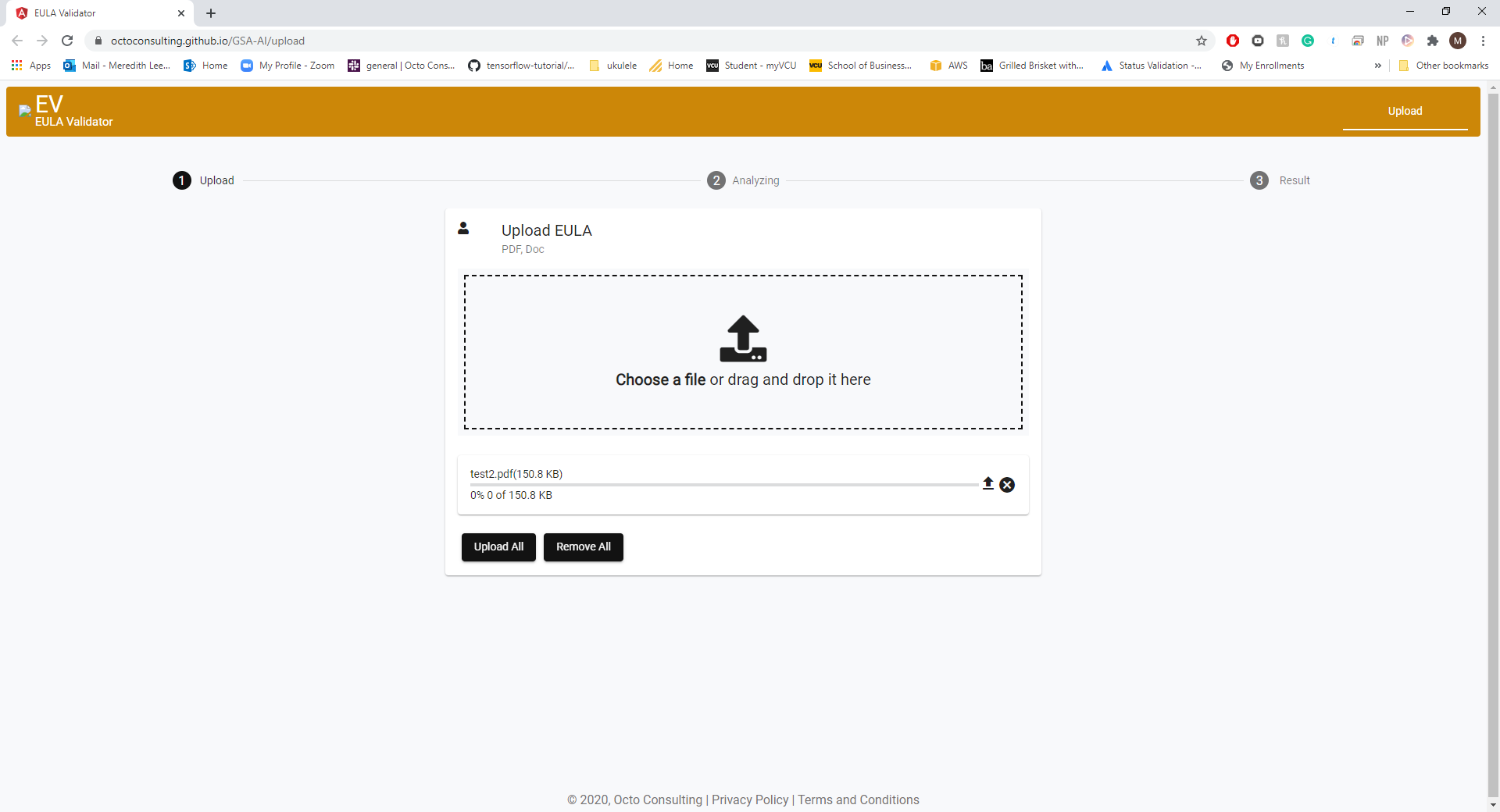


Figure Uploader Close-up

The analysis of the document by our ML model is done in the background, and the user is shown a waiting screen that gives time for the model to process the EULA. After the circles finish spinning, the user can click on the document viewing screen and see their uploaded document.

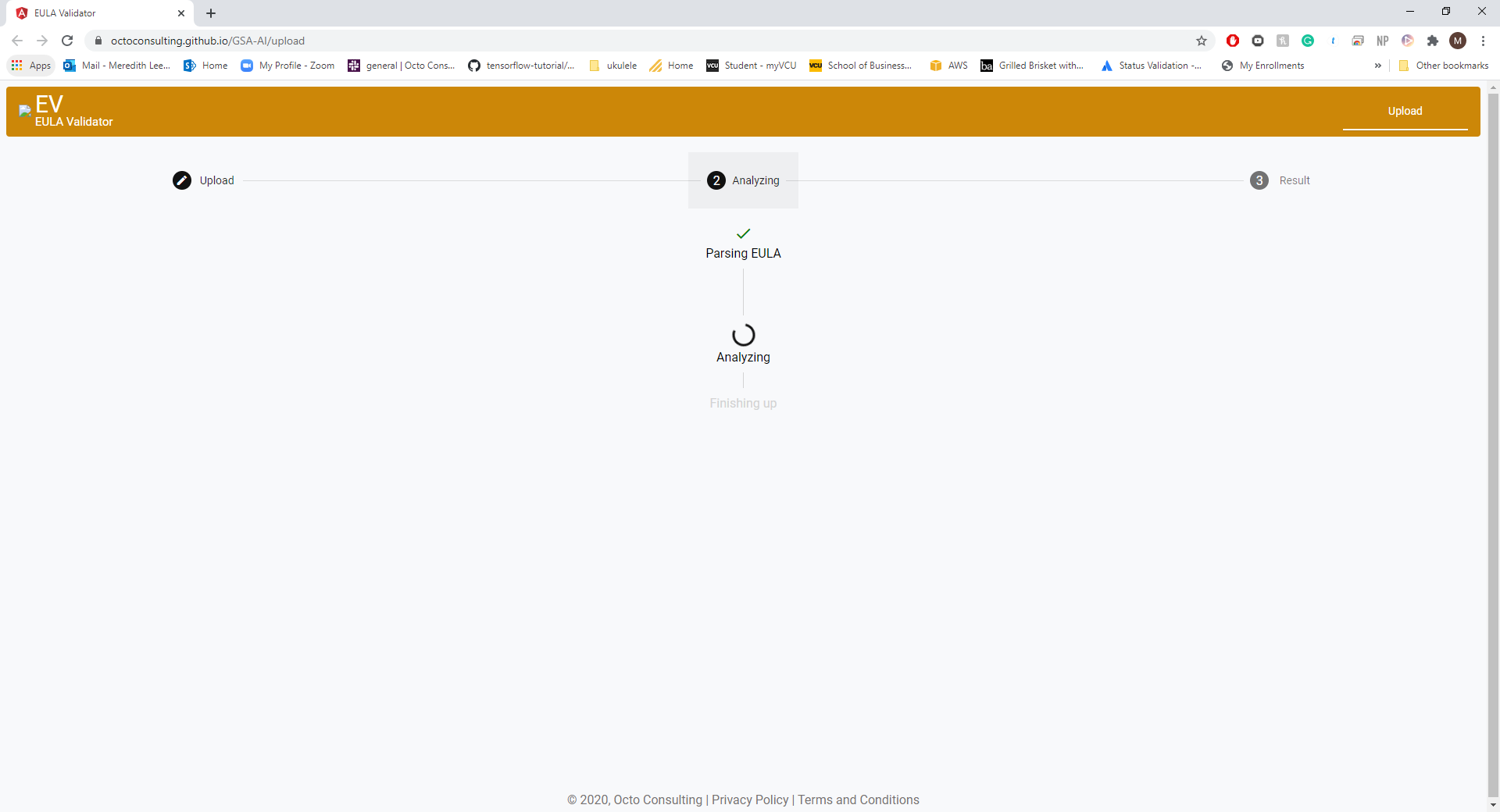


Figure Waiting Screen Close-up

On the left side will be cards with the phrases that were given back to the app by the model, either classified as acceptable (green) or unacceptable (red) along with clause scores. If a user clicks the card, the phrase will be moved into view and highlighted.

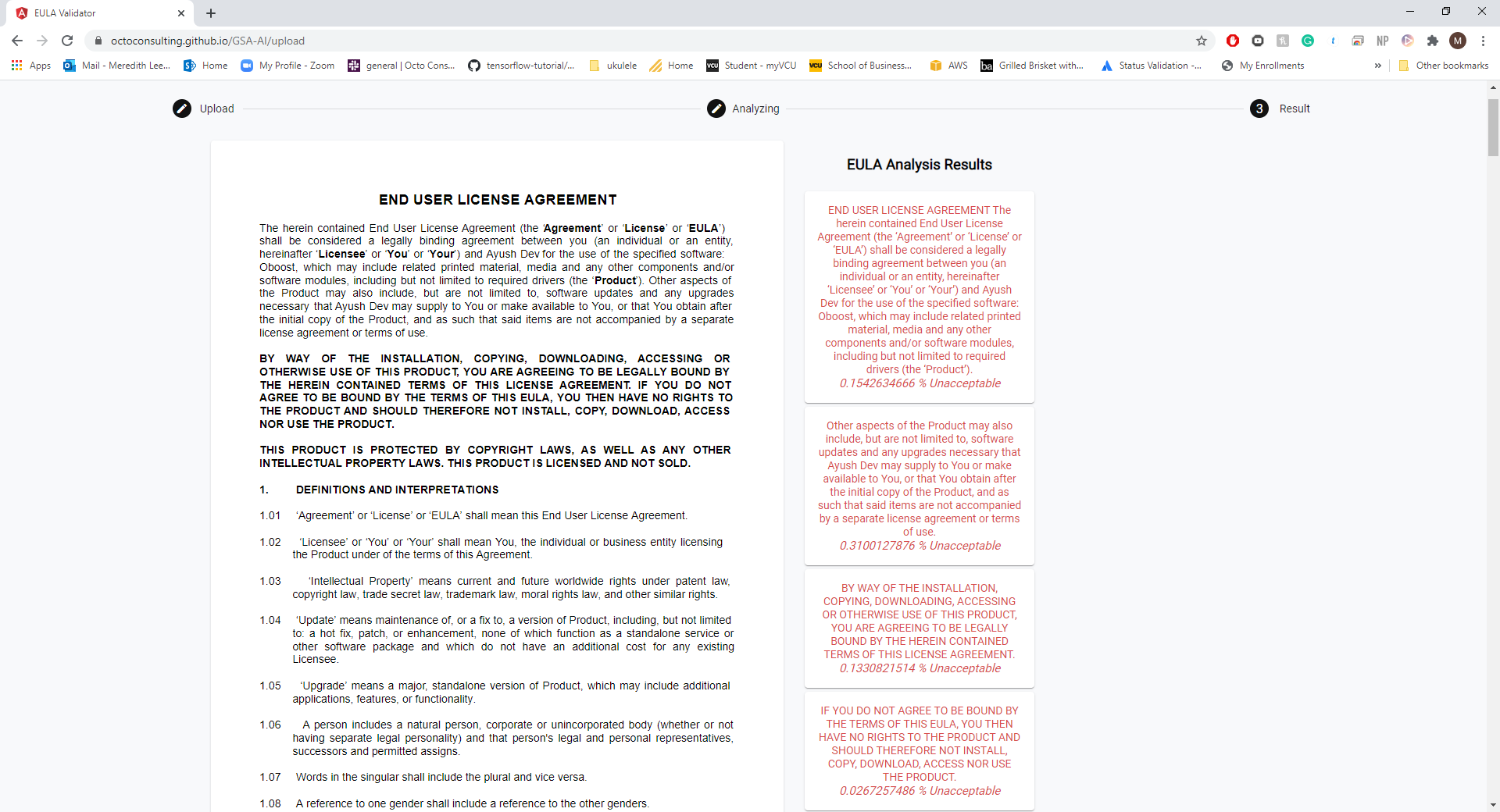


Figure Results Page

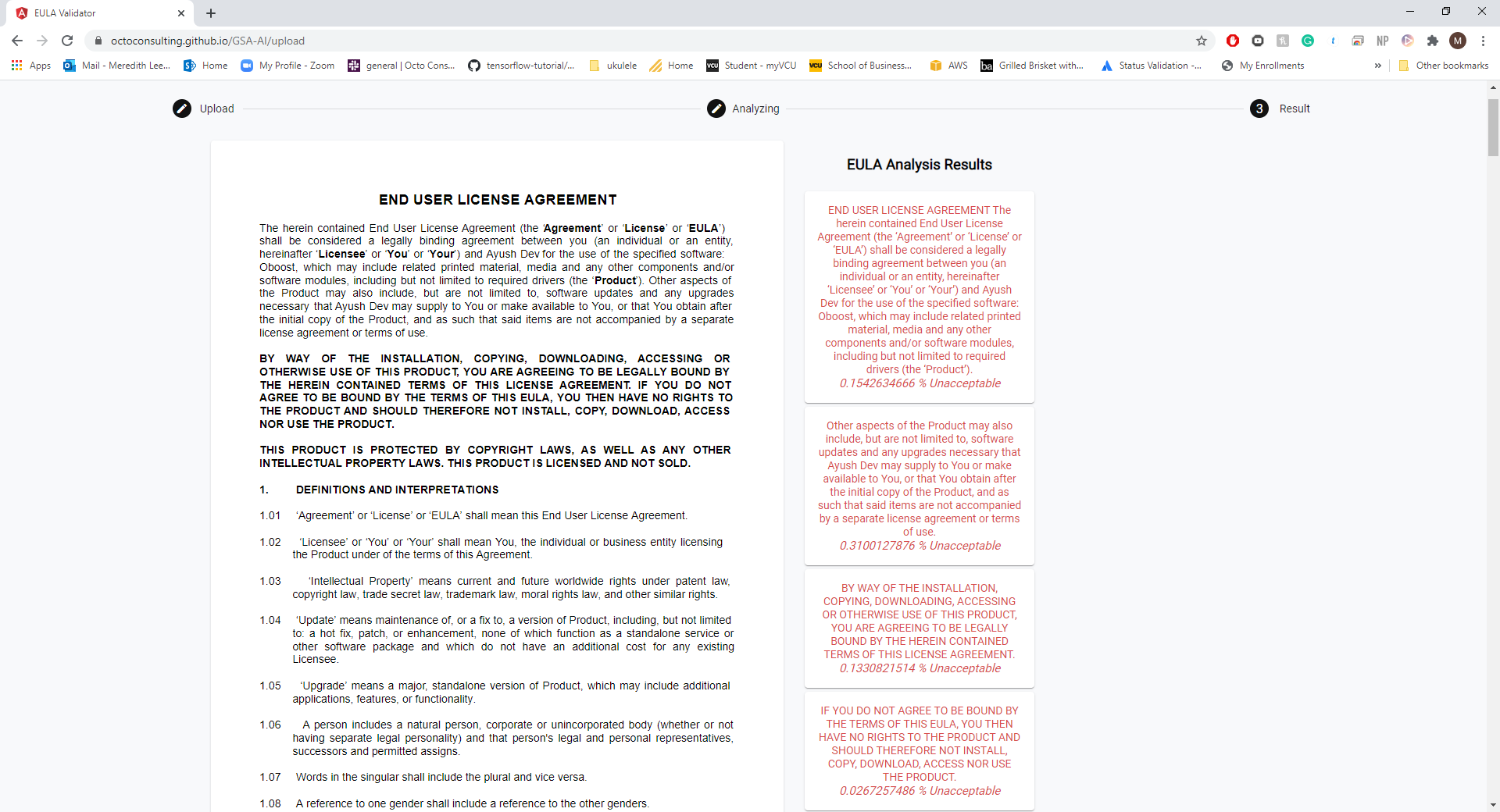
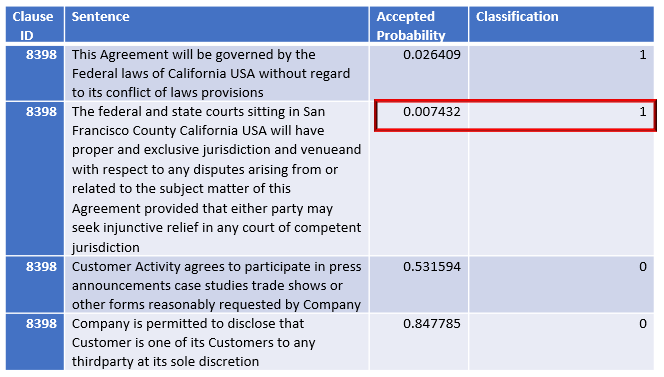


Figure Results Page Close-up

This interface is intuitive and easy to use, and makes it much easier for the user to determine which clauses are unacceptable.

# Reasons for predictions in Validation Data File

As described earlier, each clause presented in the training set was split into individual sentences for training and testing. This procedure was also followed for the provided validation data file. As seen in the image below, each clause is separated into individual sentences and then our model returns the accepted probability and classification. However, for grading purposes we want to return only a single probability and classification for each clause, and so per clause id the minimum accepted probability and the associated classification was chosen as the returned value. This can be seen in the diagram below. Clause 8398 is separated into the four individual sentences, each sentence has an accepted probability and classification returned by our trained model, and the lowest accepted probability and classification is chosen as our returned value.



This makes sense because many of the sentences within an unacceptable clause are in fact acceptable. There are usually only one or two sentences within that clause that are truly unacceptable and so make the entire clause unacceptable. Our solution attempts to overcome the fact that many of the sentences within an unacceptable clause are acceptable by increasing the granularity of assessment. Within our user interface each card shows a single sentence with the accepted probability score and classification, and so the user can quickly identify exactly which sentences within a clause are unacceptable.

# Self-Reported Metrics of Solution (Brier Score and F1 Score)

The training set was separated so that 15% of the data could be used as a test set, and the results from training indicate a 77.67% accuracy on the test set. The F1 score on the test set is 0.72, and the weighted F1 score which considers the unbalanced dataset is 0.87. The Brier score is 0.09. All of these values are compiled into the table below for easy viewing. These results are promising and given a larger more balanced dataset it is likely that we would be able to achieve a state-of-the-art model.

|  |  |
| --- | --- |
| **Metric** | **Score** |
| F1 Score | 0.72 |
| Weighted F1 Score | 0.87 |
| Brier Score | 0.09 |
| Test Set Overall Accuracy | 77.67% |

# Compiled Model

The compiled Pytorch model is too large to host on github, and so is hosted on Google Drive. The shareable link to access the model is <https://drive.google.com/file/d/1r-B3D-KvPrQqORoAF6cNyESBDhCofqxL/view?usp=sharing> and when running the solution the model is automatically downloaded.