

Documentation

#Project Overview

Title: Ambiguity Checker (ACH) for natural language requirements.

The ambiguity checker web app automatically classifies English natural language requirements as either ambiguous or non-ambiguous.

Context

Requirement engineers need to evaluate and answer customer requirements in a short time. The analysis of the requirements is extensive and time-consuming as it implies analysing hundreds of requirements per project. One of the phases of the analysis is to check for ambiguity in the given requirement text as in real-life contexts requirements can be abstract or ambiguous, i.e., they can be interpreted in more than one way. The overall goal is to support the requirement engineer experts by automatically analysing and classifying the requirements in terms of their ambiguity. Furthermore, working towards explainable AI, the models provide the reasons why the requirements were thus classified.

Key Features

- ***NLP Processing:*** utilized NLP techniques to analyze requirements such as TF-IDF.
- ***AI and ML:*** models are trained and used to classify requirements as ambiguous or not, by providing a level of confidence and an explanation for the classification.
- ***User-Friendly Interface:*** a web application is provided, where users can upload the requirements to check for presence of ambiguity.

How It Works

- ***Input:*** Using the web interface, the users can upload requirement written in English as either excel file or manually type in the requirement to check for ambiguity.
- ***Processing:*** the input entered is processed by employing NLP techniques.
- ***Classification:*** ML/AI models are trained and used to classify the requirements as ambiguous or not. The models have learned from the data in the training phase and are capable of classification on new data (user input).
- ***Output:*** For each requirement, the user is provided with a classification (Ambiguous or Not ambiguous), the classification's confidence level, and an explanation that includes terms that add to the ambiguity. The user has the possibility to provide their own validation directly after the classification and feedback/ comments, which will be used on another version of the model.

Based on the user validation and the model outcomes metrics (TP, TN, FP, FN) can be calculated and saved on the user computer.

A more detailed document representing how the user interface can be used can be found in ***docs/ User_interface_dock.docx***.

#Getting Started

To run the application in your machine the below steps are to be followed:

- **# Prerequisites:**

To implement the ACH was used *Python 3.10.4* and *PyCharm (Community version)* as an IDE. They can be downloaded from the following links:

- Python (<https://www.python.org/downloads/>).
- PyCharm (<https://www.jetbrains.com/pycharm/>): PyCharm is optional, any other python IDE can work.
- For their installation you can follow these following videos:
Python: (<https://www.youtube.com/watch?v=AwIXfaGEN4c>)
- PyCharm (<https://www.youtube.com/watch?v=XsL8JDkH-ec>)
- (<https://www.youtube.com/watch?v=OajNS-WHiUI>)

- **# Get the repository to your local machine:**

Unzip the “Alstom_AmbiguityChecker” folder.


- **# Install project dependencies:**

Guide to installing the required libraries through *PyCharm Terminal*. The required libraries are found the **dependencies.txt** file. Once installed, all the libraries are imported (*Alstom_AmbiguityChecker /Import.py*) and no further actions are needed.

- Go to your project directory: *Alstom_AmbiguityChecker*
- Open it in PyCharm IDE
- In PyCharm Terminal install the dependencies
 - *pip install -r dependencies.txt*

- **# Installation and running**

When you have installed Python and PyCharm and **unzipped** the provided folder “*Alstom_AmbiguityChecker*”, you can go the *Gui.py* to run the application.

- Open the Python IDE (such as PyCharm).
- Locate and open the *Gui.py* file in the IDE.
 - Either directly from PyCharm (File/open/...*Alstom_AmbiguityChecker*) or in the *Alstom_AmbiguityChecker* folder go to *Gui.py*, right click/ openwith /PyCharm.
- Once opened the script, run the *gui.py* in PyCharm by clicking 

▶ Run 'Gui (1)' Ctrl+Shift+F10

- The application will open on your local host, and you are ready to upload your requirements.

NiceGUI ready to go on <http://localhost:8080>, <http://169.254.145.218:8080>,

Detailed structure of the scripts inside the Ambiguity Checker Folder

Color guide: **directory**, **data**, *python script*, *saved scripts (.joblib)*, *document*.

- **data/**: contains the data.
 - **Data.xlsx** – original data from Alstom (530 requirements).
- **docs/**: Project documentation.
 - *Readme.docx*
 - *User_interface_dock.docx*
- **src/**: python source code files.
 - *Imports.py*: file with all the libraries imported.
 - *Processing.py*: all the functions needed for preprocessing.
 - *Main.py* – contains the execution of the program.
 - *Gui.py*- script to be run the application on localhost.
 - *favicon.ico*- small icon that appears in the browser tab (AC with same colors as Alstom logo).
 - **.joblib/**: all files with .joblib extension contain the serialized models (trained models).
 - *Models*
 - *decisiontree_model.joblib* – decision tree model
 - *svm_classifier.joblib* – support vector machine model
 - *vectorizer_comment.job* - Term Frequency-Inverse Document Frequency for comments in the data
 - *vectorizer_requirement.joblib* - Term Frequency-Inverse Document Frequency for requirement in the data
 - *The scores of the models: classifier_score.joblib* – score during training for (Decision Tree, SVM, Naïve Bayes)
 - *Parameters:*
 - *tfidf_svm.joblib*- the best parameters for support vector machine and *tfidf*
 - *tfidf_decisiontree.joblib*- the best parameters for decision tree and *tfidf*