

USER MANUAL

1. PROJECT INFORMATION

Project Name: Abstractive Conversation Summarizer With Open Source AI Tools (ACSWOSAT)

Date of Submission: 4/28/2023

Document Version: 1.0

Client: Impact Intell 

Team Member	Role	Document Responsibilities
Akshat Baranwal	Product Owner	Product Edition, Description of the Product, FAQs
Steven Warner	Scrum Master	Operating Instructions, Product Features
Mark O'Connor	Developer	Working of the Product, Maintenance requirements
Cameron Caruso	Developer	Installation and Configuration
Parth Dalwadi	Developer	Contacts
Michael Provenzano	Developer	Production environment, System requirements

2. USER DOCUMENTATION

Table of contents

I. Product Edition (Version)	2
II. Description of the Product	2
III. Technical Documentation	2 - 6
Working of the Product	2 - 4
Production environment	5
System requirements	5
Maintenance requirements	5 - 6
IV. User documentation	7 - 9
Product features	7
Installation and configuration instructions	7
Operating instructions	8
FAQs	8 - 9

I. Product edition (version)

Version Update : This program is currently at version 1.0, which represents the initial release of the software. As such, there are no prior versions or iterations available.

II. Description of the Product

ACSWOSAT is an AI-powered text summarization tool designed for law enforcement officers and departments in the Impact Intell industry. The tool analyzes text conversations from a database, generates brief summaries, and integrates into a Python Django web server. It enables users to communicate efficiently across cases and agencies, combatting criminals' use of information silos to evade law enforcement. The tool's algorithm is designed to mimic human-like summaries and can retrieve text conversations based on a user's specified date range. The tool will be based on open-source machine learning libraries and will be implemented on the client's Python Django based web server.

Intended Users : The intended users of ACSWOSAT are Impact Intell's users, importantly law enforcement officers and departments who need to summarize text conversations to communicate efficiently across cases and agencies. The AI-powered tool analyzes text conversations from a database, generates brief summaries, and integrates into a Python Django web server. The tool aims to combat criminals using information silos to evade law enforcement by facilitating better communication across agencies.

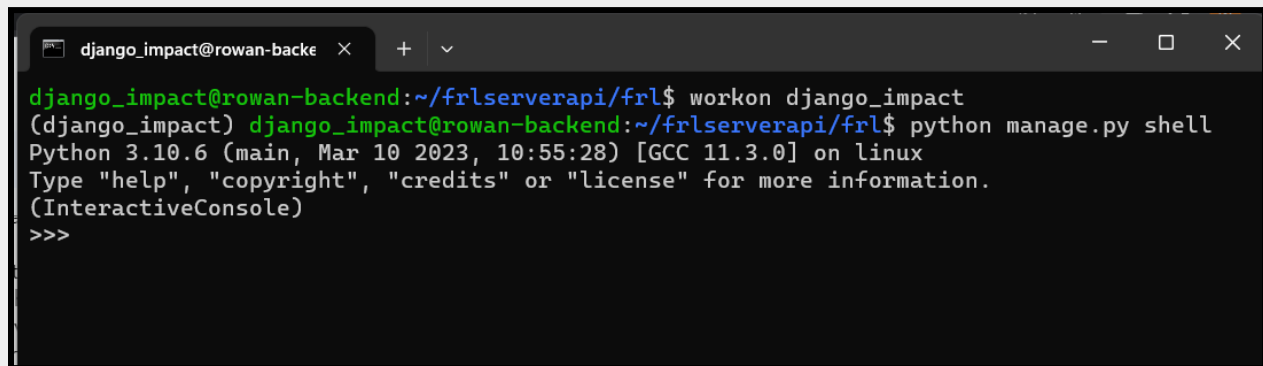
III. Technical documentation

Working of the Product

ACSWOSAT is a machine learning transformer model that has been trained to summarize text conversations in a manner similar to how humans summarize information. The model takes input in the form of text conversations, which are fed into the system as a sequence of tokens. The model then processes these tokens using multi-head attention mechanisms, which help it to identify the key information in the conversation. The output of the model is a summary of the conversation, which is generated using a decoder network that has been trained to produce a compressed representation of the input. This model has been integrated into the web application of the client, allowing users to easily access and utilize the summarization functionality.

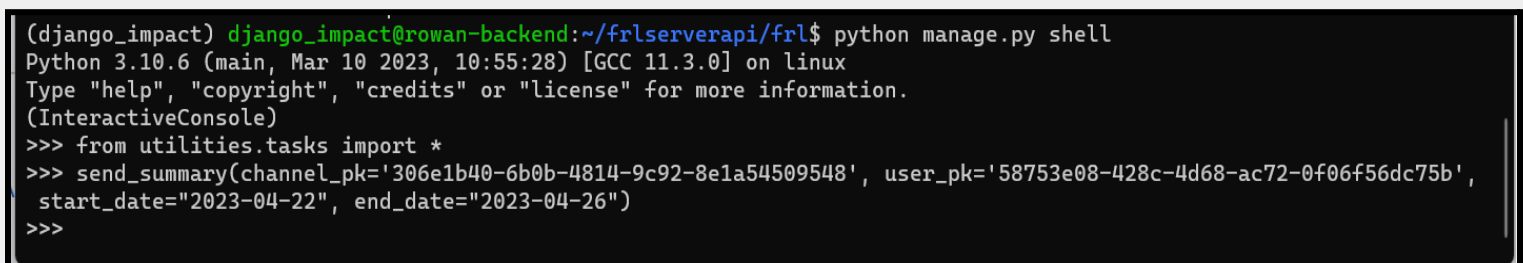
Step-by-step working: This program is currently within the Rowan-Backend server. The function is not currently connected to the web interface so it must be run within the terminal.

1. To begin, start in the “frl” directory.
2. Run the following commands:
 - **workon django_impact**
 - **python manage.py shell**
3. Within the python shell you will need access to the send_summary function. To do this run:
 - **from utilities.tasks import ***
4. Now the send_summary function will be available. The function takes the following parameters:



```
django_impact@rowan-backe x + v
django_impact@rowan-backend:~/frlserverapi/frl$ workon django_impact
(django_impact) django_impact@rowan-backend:~/frlserverapi/frl$ python manage.py shell
Python 3.10.6 (main, Mar 10 2023, 10:55:28) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>>
```

- **channel_pk**: The uuid of the channel you want to summarize.
 - **user_pk**: The uuid of the user who requested the summary.
 - **start_date**: A string that has the start date of the messages in the format of “yy-mm-dd”
 - **end_date**: A string that has the end date of the messages in the format of “yy-mm-dd”
5. To run send_summary:
 - **send_summary(channel_pk, user_pk, start_date, end_date)**

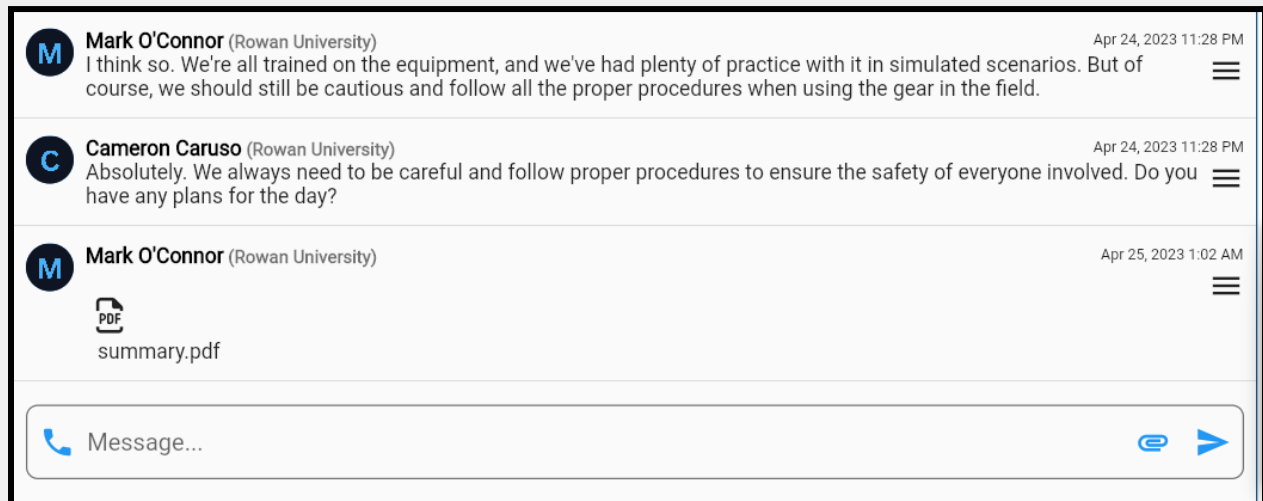


```
(django_impact) django_impact@rowan-backend:~/frlserverapi/frl$ python manage.py shell
Python 3.10.6 (main, Mar 10 2023, 10:55:28) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from utilities.tasks import *
>>> send_summary(channel_pk='306e1b40-6b0b-4814-9c92-8e1a54509548', user_pk='58753e08-428c-4d68-ac72-0f06f56dc75b',
start_date="2023-04-22", end_date="2023-04-26")
>>>
```

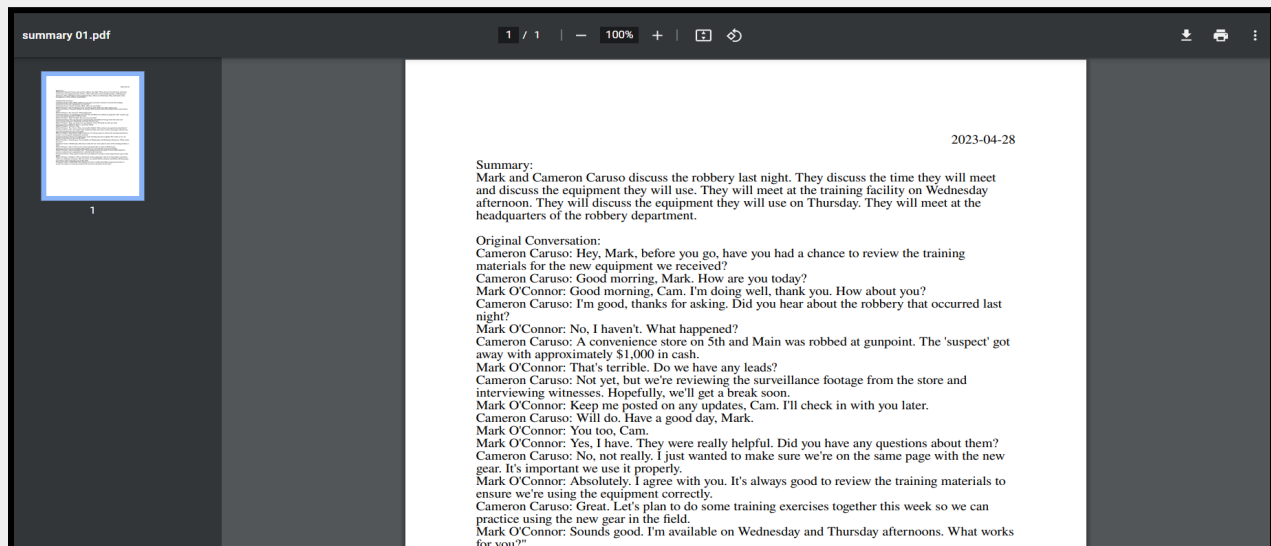
(After receiving input of the channel and the date range, the function starts collecting messages within the specified parameters. These messages are then reformatted and concatenated into a single string. The concatenated

string is then fed into the decision tree algorithm which performs filtering of irrelevant text from both the chat and the transformer. The final output of this process is a summarized version of the original conversation. This summary and the original conversation are then written into a temporary HTML file, which is then converted into a PDF and subsequently posted back into the original channel.)

6. Summary posted to channel.



7. Resulting Summary in .pdf format.



Production environment (deployment and configuration)

The ACSOWAT runs as part of a Django Python backend that utilizes Celery to asynchronously queue the task. The Django software is hosted on a Linux Ubuntu 22.04 server that Impact Intell Solutions provides.

Deployment & set-up procedures: The application exists on a development server branch and must be merged with the live server branch for deployment. Configuration for end-users includes the creation of an interface on the front-end of the web application.

System requirements

Minimum hardware and software requirements: This program needs the folders summaryDataFolder, profiles, saveFiles, utilities, and decisionTree to run the program. The program must also be running on a Django Python backend, that is utilizing Celery for task queueing. Minimum requirements for hardware include at least 16GB of RAM if training on default settings. It is recommended by Huggingface to utilize a GPU when possible as it is the most efficient method for intensive tasks such as training.

Capacity & space requirements: The total storage capacity required for all files and libraries, excluding training datasets, is 5 gigabytes.

Maintenance requirements

The transformer model can undergo more training to achieve better results. To do so you will need to use the training.py file located in utilities/t5.

- To start you will need to have two separate json files containing for datasets. Each file will need to be in the format of:

```
{
  "data": [
    {
      "id": "1",
      "summary": "",
      "dialogue": ""
    },
    {
      "id": "2",
      "summary": "",
      "dialogue": ""
    }
  ]
}
```

- Both of the json files will act as inputs for the dataset variable. You can set it as :

```
dataset = load_dataset("json", data_files={"train": "pathToTrainfile/train.json",
"validation": "pathToValidation/validation.json"}, field="data")
```

- The next step is to initialize the model :

```
model = T5ForConditionalGeneration.from_pretrained("t5-base")
```

- Setting it to “t5-base” will have the model start training from its default model. If you would like to train it using a saved model you already have you will need to pass it the path of the saved configuration.

```
model = T5ForConditionalGeneration.from_pretrained("path/savedModels/my_model10")
```

Two training parameters that will need to be adjusted are batch size and epoch.

Batch size is the number of examples that the model will take at a time to train on. The greater the batch size the better but this will be limited by processing power.

Epoch is the number of cycles through the dataset the model will do. 1 epoch means going through the dataset a single time.

- The last part of training is saving the model. We recommend that you save the model into a different directory each time you train. This allows you to have the initial model saved. **Note:** each model has multiple configuration files and the path needs to be to the directory that holds them.

```
trainer.save_model("path/savedModels/my_model16")
```

- We also recommend commenting out this line after every session to prevent accidentally overwriting.
- To change which model is being used in production you will need to modify summaryGenerator.py. This file is located in the utilities directory.

All that is required is to change the path to the model.

```
model = T5ForConditionalGeneration.from_pretrained("/path/savedModels/my_model10")
```

IV. User documentation

Product features

Features overview:

- Text summarization of Impact App chat conversations
- Integration with a Python Django web server
- AI-powered transformer model for summarization
- Decision tree for removing fluff and irrelevant content
- Ability to process conversations within a specified date range
- Conversion of summary into PDF format
- User-friendly interface for easy interaction
- Efficient communication across agencies and cases for law enforcement officers.

Installation and configuration instructions

ASCOWAT is a machine learning-based summarization tool that can be integrated into the web application of the client. To install and configure ACSOWAT, the client needs to follow the steps mentioned below:

1. Download and place the ASCOWAT source code into a specific directory. Files include :
 - a. decisionTree.py
 - b. dataCollector.py
 - c. summaryGenerator.py
 - d. training.py
2. Ensure that all the required libraries and dependencies are installed in the system.
3. Integrate the send_summary function provided in the source code into the client's web application.
4. Configure the function with the required input parameters such as channel ID, user ID, start date, and end date.
5. Run the function from the terminal or command prompt to generate the summary in PDF format and post it back into the original channel.
6. Test the functionality thoroughly to ensure that it is working as expected.
7. Following these steps will enable the client to integrate the ASCOWAT summarization tool into their web application and use it to generate summaries of conversations within their channels.

Operating instructions

The operating instructions for the ACSOWAT Summarizer are essentially the step-by-step working process of the program. These instructions provide clear and concise guidance on how to run the program through the terminal, including necessary commands and input parameters. Following these instructions will enable users to utilize the summarizer to collect, filter, and summarize messages within a specified date range, and generate a summarized version of the original conversation in PDF format.

The steps are:

- Start in the "**fri**" directory.
- Run the command "**workon django_impact**".
- Run the command "**python manage.py shell**".
- Within the Python shell, import the send_summary function using the command "**from utilities.tasks import ***".
- Use the send_summary function with the following parameters:
 - channel_pk: The UUID of the channel to summarize.
 - user_pk: The UUID of the user who requested the summary.
 - start_date: A string with the start date of the messages in the format "yy-mm-dd".
 - end_date: A string with the end date of the messages in the format "yy-mm-dd".
- The function collects messages within the specified date range, reformats them, and concatenates them into a single string.
- The concatenated string is filtered using the decision tree algorithm to remove irrelevant text.
- The resulting summary and original conversation are written to a temporary HTML file, converted to PDF, and posted back to the original channel.

FAQs

Error messages and error handling:

In ACSWOSAT, error messages will be generated in a specific format that includes information about the method or function that originated the error and the corresponding class or filename. This format is designed to provide detailed and specific information about the nature of the error, helping developers to identify and address the underlying issue. By providing this level of technical detail, ACSWOSAT can facilitate efficient and effective troubleshooting and debugging, enabling developers to quickly and accurately resolve any issues that arise.

Troubleshooting:

- Verify your login credentials - Make sure you are using the correct username and password to log in to the Impact Intelligence frontend.
- Check the database connection - If the backend is unable to retrieve the conversation from the database, check the connection to ensure it is properly configured.
- Verify the date range - If you are not receiving the desired results, double-check the date range specified in your request to ensure it is correct.
- Check the input format - ACSWOSAT requires text conversations in a specific format. If you are having issues, ensure that your input is in the correct format.
- Contact technical support - If you have tried the above troubleshooting options and are still experiencing issues, contact technical support for further assistance.

Contacts :

ACSWOSAT Technical Support Discord Link - <https://discord.gg/tZEsEX3n>

Team Member	Role	Individual Contact
Akshat Baranwal	Product Owner	baranw63@rowan.students.edu
Steven Warner	Scrum Master	warner87@students.rowan.edu
Mark O'Connor	Developer	oconno65@rowan.students.edu
Cameron Caruso	Developer	caruso89@students.rowan.edu
Parth Dalwadi	Developer	dalwad44@students.rowan.edu
Michael Provenzano	Developer	proven24@students.rowan.edu