ICSI 499 Team 4: Milestone IV

Michael Paglia, Proshanto Dabnath, Joseph Regan, Michael Alexander Smith

March 2024

1 Introduction

1.1 Project Statement

The goal of the project is to develop software for PySpady. This free-to-use Python library enables users to leverage novel and classical sparse encoding algorithms and methodologies to analyze spatial-temporal data. The library will contain features including missing value imputation, future value prediction, and more. The design and implementation of this software will contain the ability to determine the best combination of dictionaries/models/coding optimizers for a dataset based on some learning score.

2 Proposed Solutions to Functional Requirements

2.1 Data Management System

- Implement a system allowing user to input datasets for Temporal Graph Signal Decomposition (TGSD) and save any resulting outputs.
  - Solution 1: Implement a Database Management System, such as SQLite [10] or MySQL [6], to handle data storage. Use SQL queries for data retrieval and management. Implement security measures for the integrity of the data.
  - Solution 2: Utilize a config file where users can specify data paths, allowing for processing and storage of data locally.
  - Preferred Solution: Solution 2. This solution is a more manageable solution and allows us to not to have worry about compatibility issues between the Python library and a DBMS in the future. This solution also stays more inline with the core objective of developing a Python library instead of deviating to unrelated features separate from the library.

2.2 Graphical User Interface (GUI)

- Develop an intuitive GUI accessible via web or local interface for interaction with the software and its visualization capabilities.
  - Solution 1: Develop a web-based GUI, using frameworks such as Flask [3] or Django [2], providing users with an interactive interface to the TGSD library.
  - Soltuion 2: Implement a command line interface GUI for ease of interaction with the Python library.
  - Preferred Solution: Solution 2. The project is first and foremost a Python library, so the first version of a GUI we will be implementing will be via an interface closely related to the usage of Python libraries. In the future, given sufficient time and resources, a web-based GUI can be explored.

2.3 TGSD

- Implement TGSD, a scalable algorithm offering solutions in matrix decomposition, imputation of missing values, temporal interpolation, clustering, period estimation, and rank estimation. [5]
  - Solution 1: Implement TGSD in Python using libraries such as NumPy [7] and SciPy [9]. (Utilize parallel processing techniques with libraries such as Dask or multiprocessing for scalability. Optimize the algorithm performance via libraries such as numba or Cython.)
  - Solution 2: Implement TGSD in MATLAB using built-in functions for matrix operations. (Utilize MATLAB's parallel computing toolbox for scalability. Optimize algorithm performance by using MEX functions and C/C++ code.)

  - Preferred Solution: Solution 1. The original plan for the project was to develop a Python library. Python is the goto language for data processing and data analysis in the field and as such has a large audience and a diverse set of libraries and tools available for use.

2.4 Auto-configuration

- Develop an auto-configuration system built into the software. The system should have default parameters for the TGSD algorithm, an estimated time of completion, and capabilities of producing explanatory figures.
  - Solution 1: Utilize a machine learning based techniques such as grid search [8] or Bayesian optimization to search for optimal parameter configurations. Provide an interface for users to fine-tune configurations if necessary.
  - Solution 2: Use a rule-based system that provides configurations based on predefined rules. Design a system to define rules for different dataset types and use cases. Provide an interaface to override suggested configurations if necessary.
  - Preferred Solution: Solution 1. We plan to implement a form of grid search to find an optimal or near optimal configuration of hyperparameters. This will allow the library to be general to a variety of datasets and use cases.

2.5 Outlier Detection

- Use the TGSD algorithm to design algorithms to find communities via clustering methods.
  - Solution 1: Apply distance based methods, such as Euclidean distance or magnitude-based distance, to identify outliers. Provide configurable thresholds to adjust the quantity of detected outliers. methods, such as Z-score or interquartile range, to identify outliers. Provide configurable thresholds to adjust the sensitivity of outlier detection.
  - Solution 2: Apply statistical methods, such as Z-score or interquartile range, to identify outliers. Provide configurable thresholds to adjust the sensitivity of outlier detection.
  - Preferred Solution: Solution 1. Distance based methods are a generalizable and intuitive way to detect outliers and perform outlier inference. Less statistical knowledge is required to understand the thresholds of statistical based methods compared to distance based methods.

2.6 Community Detection

- Use the TGSD algorithm to design algorithms to find communities via clustering methods.
  - Solution 1: Apply centroid-based clustering methods, such as k-means [4], to visualize relationships between graph nodes.
  - Solution 2: Apply density-based clustering methods, such as DBSCAN [1], to visualize the communities between the graph nodes.
  - Preferred Solution: Solution 1. Centroid-based methods are a common way to perform clustering. K-means is a simpler, more scalable, and easier to understand form of clustering compared to DBSCAN.

3 System/Data Flow Diagrams

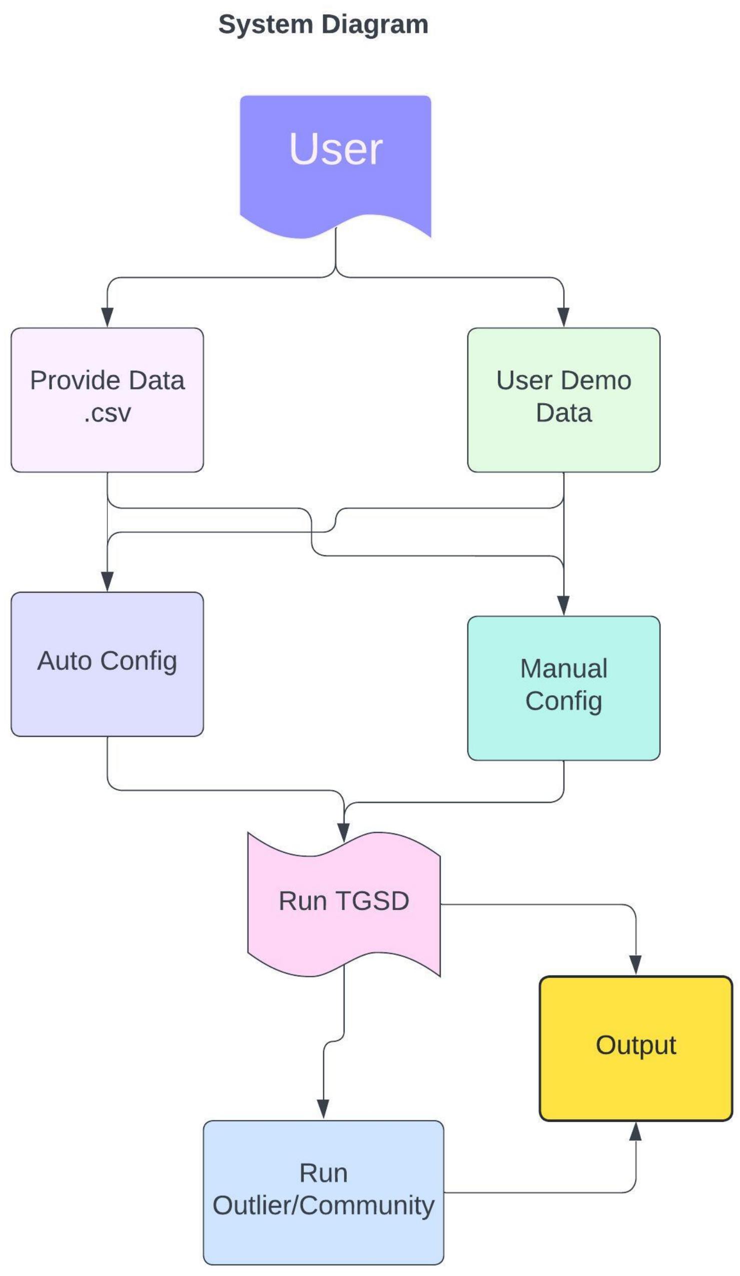

Figure 1: System Diagram

- The user can use demo data or input data.
- The user can use manual config or auto config.
  - (*) The user can run TGSD.
  - (*) The user can run outlier detection.
- Stared points (*) denote final output.

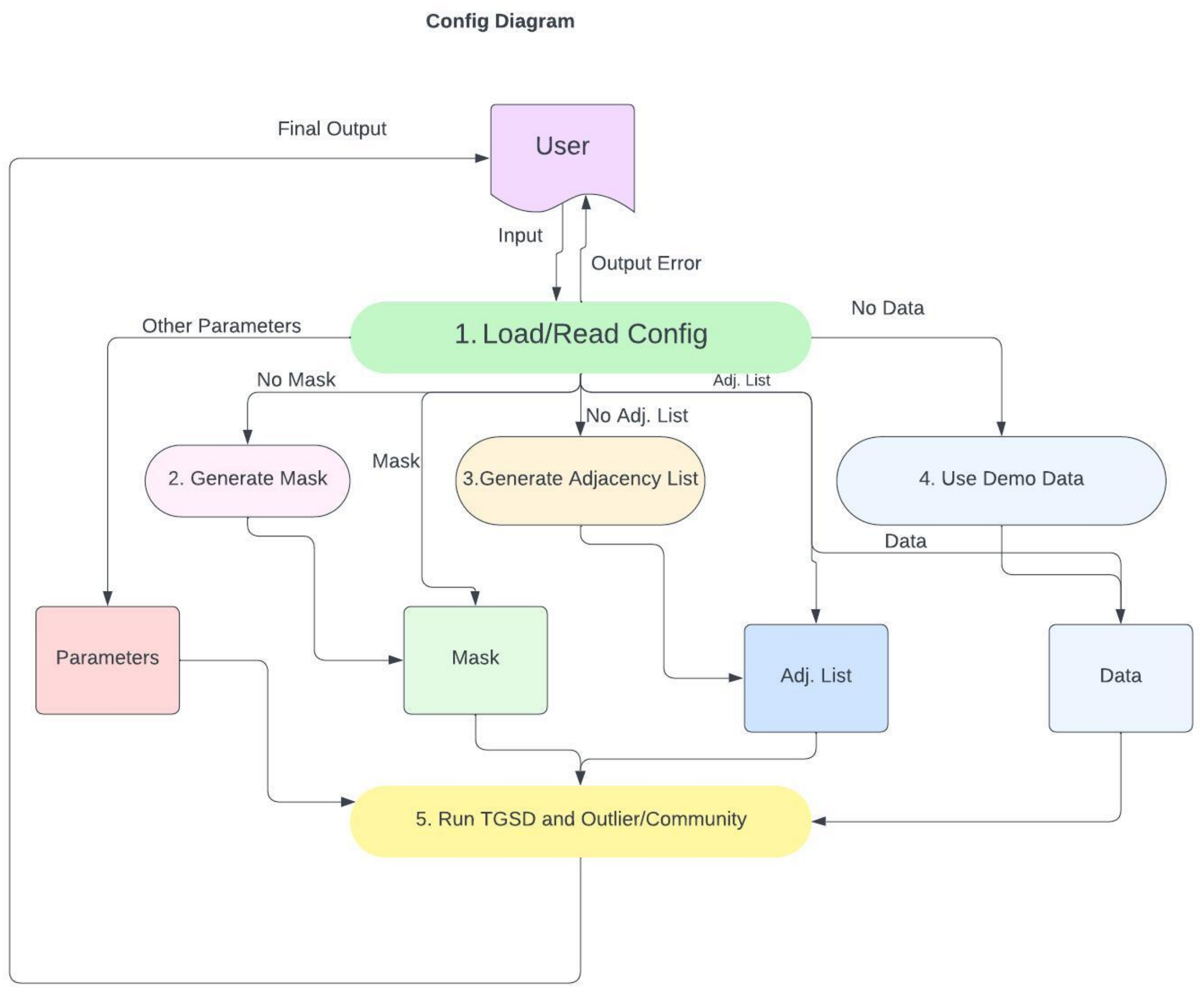

Figure 2: Data Flow Sub-Diagram

- The user inputs data (could be demo data).
- The user can use manual config or auto config.
  - Process 1: Loads data to memory and reads the config.
  - Process 2, 3, and 4 generate required inputs if they are not provided.
  - The 4 databases correspond to required inputs for TGSD.
  - Process 5 is to run TGSD/Outliers.

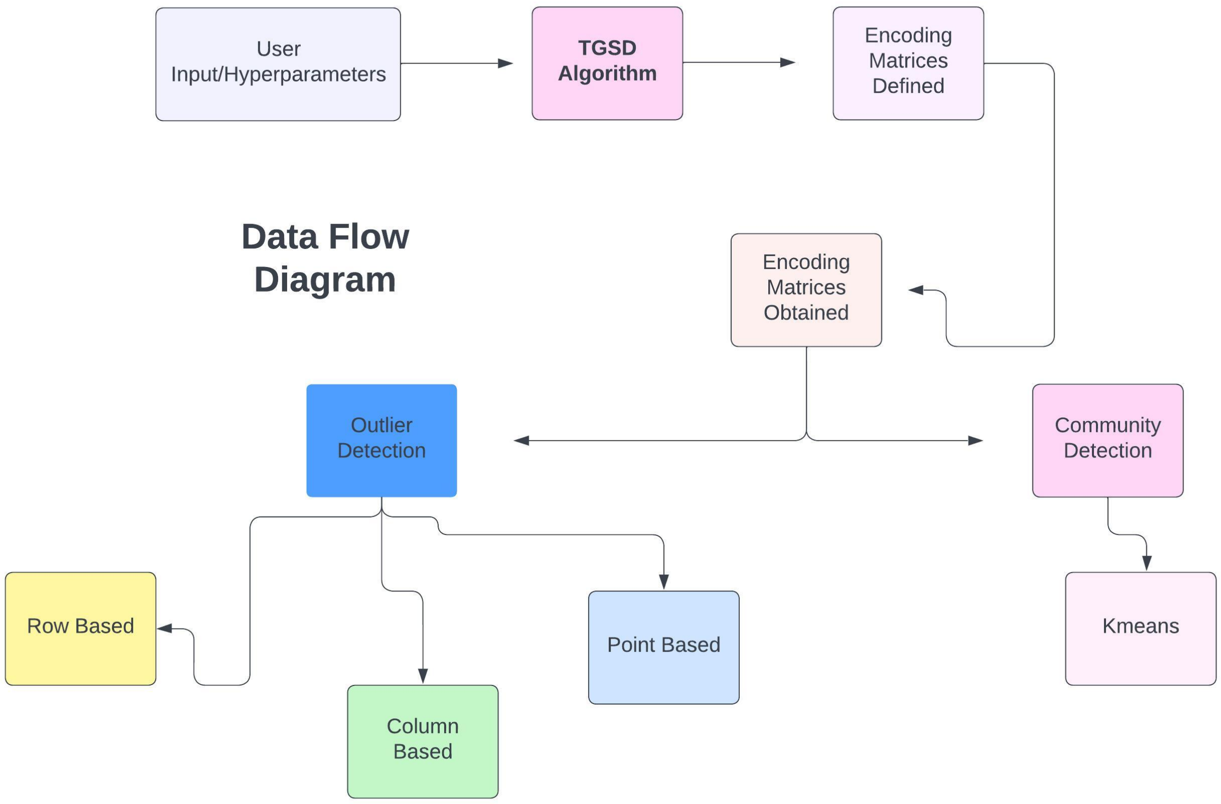

Figure 3: Config Diagram

4 Proposed GUI Sketch

4.1 GUI Approach: Command Line Based

```
>>Welcome to PySpady, developed by {...}
>>Before starting, would you like to see some demo data? Here are some pre-prepared examples:
    - ...
    - ...
    - ...
>> Provide .csv file(s) in config file or Pandas df with values loaded along with
    hyperparameters
>>Read documentation on available dictionaries, algorithms and hyperparameters here: www.test.
    com
>>Would you like to run "autocomplete"? This will search for the beset permutation of
    dictionaries, optimizers/models, and hyperparameters, but can be time intensive [Y/N]
>>if Y: run smartsearch
>>Output best parameters
>>if N: Hyperparameters
>> chosen in config
>>What downstream task would you like to perform? (Outliers/Clustering)
>>If outliers: select
    -"row"/"column"/"point"
    and count
>>If Clustering:
    -select n clusters
>>Repeat on new dataset
```

Listing 1: Python example

Upon launch, GUI will prompt user if they would like to use prepared demonstration data. Ask user for input in explanatory config.json file or a pandas DataFrame with proper values loaded, e.g. hyperparameters, dictionaries, temporal graph signal, etc. Provide user with documentation resources and ask if they want to use the "autocomplete" feature on their input. Ask user if they want to perform downstream tasks. Ask user if they want to repeat on a new dataset.

References

[1] *DBScan*. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

[2] *Django*. https://www.djangoproject.com/.

[3] *Flask*. https://flask.palletsprojects.com/en/3.0.x/.

[4] *KMeans*. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

[5] Maxwell McNeil, Lin Zhang, and Petko Bogdanov. "Temporal Graph Signal Decomposition". In: *ACM International Conference on Knowledge Discovery and Data Mining*. Online, 2021.

[6] *MySQL*. https://www.mysql.com/.

[7] *Numpy*. https://numpy.org/.

[8] *Sci-Kit Learn*. https://scikit-learn.org/stable/modules/grid_search.html.

[9] *SciPy*. https://scipy.org/.

[10] *sqlite*. https://www.sqlite.org/.