Khandakar Wahiduzzaman & Kenneth Guillont
Professor Lei Xie
CSCI 49376 01
Project 2

Introduction:

Hetionet is a heterogeneous network of biomedical knowledge constructed from a collection of 29 publicly available databases of genes, compounds, diseases, and more. In this project, we will use a subset of this dataset containing the three entities mentioned. These entities are nodes on a graph and the relationships between them serve as the edges that connect them.

This project implements Pyspark to identify key relationships between genes, diseases, and compounds. Pyspark is the Python API for Apache Spark, its implementation enables real-time, large-scale data processing in a distributed environment using Python.

Given this information, the project uses PySpark to:

1. Classify relationships by filtering between drugs, genes, and disease.

2. Compute gene and disease associations for drugs

3. Sort drugs by the number of gene associations (highest to lowest)

Technologies used:

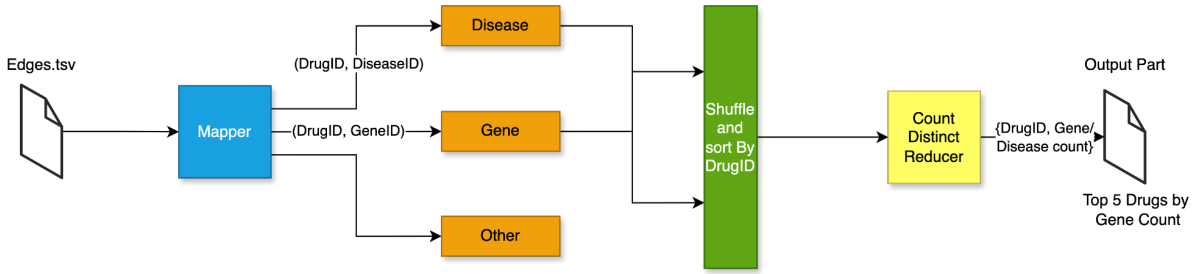
- Python: The programming language used for scripting.

- PySpark: For distributed data processing and analysis.

- TSV: File containing tab-separated values. The data format for nodes and edges.

implemented map-reduce algorithms:

Q1: For each drug, compute the number of genes and diseases associated with the drug. Output results with the top 5 number of genes in descending order.



This section of code shows our preliminary task to prepare our information, as well as our approach to solving this first task. The first steps include: loading both tsv files and implementing each as dataframes, classifying our target column into Gene, Disease, or other, and filtering drug-to-disease and drug-to-gene interactions.

To solve this task, first count the number of genes and diseases per drug. Once complete, the program reduces gene and disease counts by merging variables GeneCount and DiseaseCount. Lastly, the program outputs the top five Drugs based on gene count in the form of a dataframe named combined_counts.

```
Load `nodes.tsv` as nodes_df with schema (id, name, kind)
Load `edges.tsv` as edges_df with schema (source, metaedge, target)

For each row in edges_df:
    If target starts with 'G', set type as 'Gene'
    Else if target starts with 'Dis', set type as 'Disease'
    Otherwise, set type as 'other'

Filter edges_df:
    Where source starts with 'Compound::' and type is 'Disease' →
drug_to_disease_df
    Where source starts with 'Compound::' and type is 'Gene' →
drug_to_gene_df

Group drug_to_gene_df by source:
    GeneCount = Count distinct target values

Group drug_to_disease_df by source:
    DiseaseCount = Count distinct target values

Join gene_counts and disease_counts on source using an outer join:
    Fill missing GeneCount or DiseaseCount with 0
Convert GeneCount and DiseaseCount to integers

Sort combined_counts by GeneCount in descending order:
    Select top 5 records
```

Mapping:

- **Inputs:** Data from edges.tsv containing relationships between the metaedge, target, and source entities.

- This data is filtered by classification, drug information is acquired by filtering source entities, considering entities that only start with “Compound::”. While the target entities are classified as either a Gene or Disease by filtering for their category.

Key-Value Pairs:

- Our key-value pairs are represented as {DrugID:TargetID}
    - Key: DrugID from source col.
    - Value: TargetID from target col.

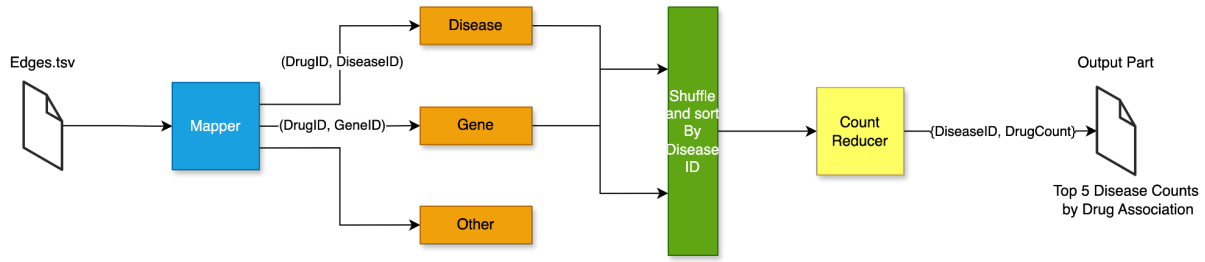
Reducing stage:

- Values are aggregated by key to calculate all distinct target IDs for all given drugs. Then, they are separated into the GeneCount and DiseaseCount variables.
- **Intermediate Key Value Pair:** produces a key pair in {DrugID: GeneCount and DiseaseCount tuple}
    - Key: DrugID
    - Value: a tuple with GeneCount and DiseaseCount

Post-Processing:

- **Output:** Our computed information consists of the top 5 drugs with the highest amount of associated genes, this is sorted by the GeneCount variable in descending order, represented in table format.

Q2: Compute the number of diseases associated with 1, 2, 3, …, n drugs. Output results with the top 5 number of diseases in descending order.



This section focuses on counting the number of drugs that are associated with each disease stored in variable DrugCount. Following this, count how many diseases are associated with a specific number of drugs. Finally, output a table of expected values in descending order.

```
Group drug_to_disease_df by target:
    DrugCount = Count distinct source values

Group disease_drug_counts by DrugCount:
    DiseaseCount = Count the number of occurrences for each DrugCount
```

```
Sort results by DiseaseCount in descending order:
    Select top 5 records
Emit (DrugCount, DiseaseCount)
```

Mapping:

- **Inputs:** Data from edges.tsv contains filtered relationships where target is classified as Disease and source is a Compound, which is our drug.

Key-Value Pairs:

- Our key-value pairs are represented as {DiseaseID:DrugID}.
    - Key: DiseaseID from target col.
    - Value: DrugID from source col.

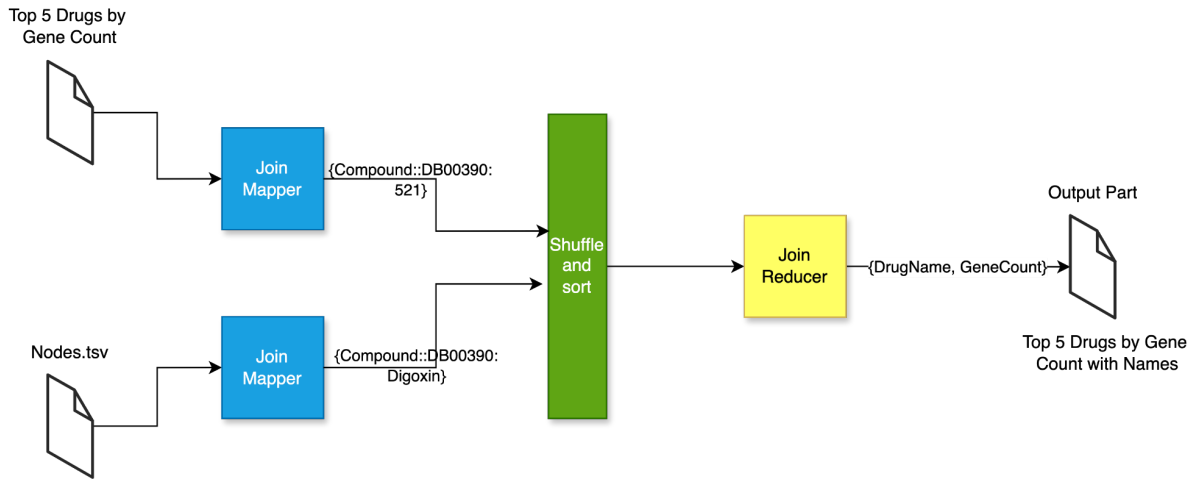
Reducing stage:

- Values are aggregated as DiseaseID to account for all unique drugs that are associated with each disease.
- **Intermediate Key Value Pair:** produces a key pair in {DrugCount: DiseaseCount}
    - Key: Number of drugs associated with a disease represented by variable DrugCount
    - Value: A count of all associated diseases.

Post-Processing:

- **Output:** Our computed information consists of the top 5 drugs with the highest recorded count of diseases, this is sorted by the DiseaseCount variable in descending order, represented in table format.

Q3: Get the name of drugs that have the top 5 number of genes. Output the results.



By using the output of Q1, this section uses a join pattern to combine are nodes_df and combined_count to create a table with a new column that maps disease IDs to their respective names. The output is similar to the output of Q1, except DrugName is the key in place of DrugID.

```
Join top_5_genes with nodes_df where source matches id(left join):
    Select columns name and GeneCount
Emit (DrugName, GeneCount)
```

Mapping:

- **Inputs:** Two inputs are used here, one of which is the entity information from nodes.tsv, this includes columns id, name, and kind. The second input is the computed GeneCount from Q1, the top 5 drugs with the highest count of genes.

Key-Value Pairs:

- Because we have two different inputs, we would expect two different key-value pairs
    - Q1 results in the pair {DrugID:GeneCount}
    - Data from nodes.tsv is mapped to the key-value pair {DrugID:DrugName}

Reducing stage:

- the information provided by DrugID and DrugName is joined together alongside of GeneCount
- **Output:** Output produces a key pair in {DrugName: GeneCount}

Post-Processing:

- **Output:** Similar to Q1, this output produces a table of the five drugs with the most associated genes listed in descending order. However, the names are listed as opposed to the drugID.

MapReduce Patterns:

Various MapReduce patterns exist to support distributed data processing tasks; within this project, summarization, filtering, and join patterns are leveraged to support our findings.

- **Summarization patterns** are implemented to help users compute aggregate values over a dataset. These clues: counts, sums, averages, and so on. Both Q1 and Q2 utilize this pattern to count occurrences.
    - Q1: Our program computes the number of genes and diseases associated with each given drug.
    - Q2: We computed the number of diseases associated with n-many drugs.
- **Filter patterns** are used when the user is interested in data that fits a specific criteria. Throughout our project, we filtered data based on our interest to narrow down data before the Reduce phase.
    - The `source` column of our edges.tsv file was filtered for sources starting with “Compound::”, discarding other possible sources (Anatomy, Gene, Disease).
    - The `target` column was filtered based on whether the target is a Gene or Disease.
- **Join Patterns** come into play when the user wants to combine different data sources based on a common key. Our approach to Q3 does this by taking the `combined_count` dataframe, produced during Q1, and joins it to `nodes.tsv`. Both files share similar id attributes and are combined to create a new dataframe `top_5_with_names` to get the names of the drugs with the highest gene count.