Indian Institute of Technology Madras Zanzibar Campus

Chemistry Assignment II

Manas Pandya - ZDA23B019

May 2024

1 Assignment Project Statement:

In this assignment, we have been asked to write a Python program to analyze the structural properties of given chemical elements using data from a CSV file and the RDKit library. The program will perform the following tasks:

- Get a list of chemical elements from the user as input, restricted to a maximum of two elements from the set {"C", "H", "N", "O", "F"}.
- Generate a table-like output containing the following information for each atom in the specified elements:
  - Coordinates (x, y, z) from both the structure.csv file and RDKit.
  - Number of bonds (n_bonds) and mean bond length (mean_bond_length) calculated from the structure.csv file using a provided function.
  - Number of bonds (n_bonds) and mean bond length (mean_bond_length) calculated from the RDKit library.
  - Bond length difference between the values obtained from structure.csv and RDKit.
  - Euclidean distance between the coordinates obtained from structure.csv and RDKit.
  - Element symbol (e.g., H).

2 The code

The following is the code used for the assignment:

```
""" This is the Chemistry assignment 2 of Manas Pandya to compare rdkit and champs datasets """
import pandas as pd
import numpy as np
from rdkit import Chem
from rdkit.Chem import AllChem
import tkinter as tk
from tkinter import ttk
from tkinter import simpledialog

structures = pd.read_csv('C:/Users/User/Downloads/chem/structures.csv')

radii = {
    'H': 0.38,
    'C': 0.77,
    'N': 0.75,
    'O': 0.73,
    'F': 0.71
}

structures['radius'] = structures['atom'].map(radii)
```

```
def get_bond_info(atoms):
    """
    calculating bond information for given atoms.
    Parameters:
    atoms (pd.DataFrame): DataFrame containing atom properties.
    Returns:
    tuple: n_bonds (list), mean_bond_len (list)
    """
    max_atoms = 28
    idx = atoms['atom_index'].values
    pos = atoms[['x', 'y', 'z']].values
    mol_names = atoms['molecule_name'].values
    radii = atoms['radius'].values

    bonds = np.zeros((len(atoms), max_atoms), dtype=np.int8)
    bond_dists = np.zeros((len(atoms), max_atoms), dtype=np.float32)

    for i in range(1, max_atoms):
        for j in range(len(pos) - i):
            if mol_names[j] == mol_names[j + i]:
                dist = np.linalg.norm(pos[j] - pos[j + i])
                if 0.1 < dist < (radii[j] + radii[j + i]):
                    bonds[j, i] = 1
                    bonds[j + i, -i] = 1
                    bond_dists[j, i] = dist
                    bond_dists[j + i, -i] = dist

    n_bonds = [np.sum(bonds[row]) for row in range(len(bonds))]
    mean_bond_len = [np.mean(bond_dists[row][bonds[row] == 1]) if np.any(bonds[row] == 1) else 0 for row in r

    return n_bonds, mean_bond_len

def get_champs_data(el, structs):
    """
    getting CHAMPS data for a given element.
    Parameters:
    el (str): Element symbol.
    structs (pd.DataFrame): DataFrame containing structure data.
    Returns:
    dict: Dictionary containing molecule names, positions, number of bonds, and mean bond length.
    """
    el_data = structs[structs['atom'] == el]
    n_bonds, mean_bond_len = get_bond_info(el_data)
    return {
        "mol_names": el_data['molecule_name'].values,
        "positions": el_data[['x', 'y', 'z']].values,
        "n_bonds": n_bonds,
        "mean_bond_len": mean_bond_len
    }

def get_smiles_code(molecule_name, structures):
    """
    generating the SMILES code for a given molecule name using its atomic structure.
    Parameters:
    molecule_name (str): The name of the molecule.
    structures (pd.DataFrame): DataFrame containing structure data.
    returns:
    str: The SMILES code of the molecule.
```

```
    """
    mol_data = structures[structures['molecule_name'] == molecule_name]

    if mol_data.empty:
        return None

    mol = Chem.RWMol()

    atom_indices = {}
    for idx, row in mol_data.iterrows():
        atom = Chem.Atom(row['atom'])
        atom_idx = mol.AddAtom(atom)
        atom_indices[row['atom_index']] = atom_idx

    for i, row1 in mol_data.iterrows():
        for j, row2 in mol_data.iterrows():
            if i < j:
                dist = np.linalg.norm(row1[['x', 'y', 'z']].values - row2[['x', 'y', 'z']].values)
                if 0.1 < dist < (radii[row1['atom']] + radii[row2['atom']]):
                    mol.AddBond(atom_indices[row1['atom_index']], atom_indices[row2['atom_index']], Chem.rdch

    Chem.SanitizeMol(mol)
    smiles = Chem.MolToSmiles(mol)
    return smiles

def get_rdkit_info(smiles_code):
    """
    getting RDKit information for a given molecule's SMILES code.
    Parameters:
    smiles_code (str): SMILES code of the molecule.
    Returns:
    dict: Dictionary containing positions, number of bonds, and mean bond length.
    """
    if not smiles_code:
        return {
            "positions": np.array([]),
            "n_bonds": 0,
            "mean_bond_len": 0
        }

    mol = Chem.MolFromSmiles(smiles_code)
    mol = Chem.AddHs(mol)
    AllChem.EmbedMolecule(mol)
    AllChem.MMFFOptimizeMolecule(mol, maxIters=200)
    atom_pos = mol.GetConformer().GetPositions()

    bond_lengths = []
    for bond in mol.GetBonds():
        start_idx = bond.GetBeginAtomIdx()
        end_idx = bond.GetEndAtomIdx()
        bond_len = np.linalg.norm(atom_pos[start_idx] - atom_pos[end_idx])
        bond_lengths.append(bond_len)

    n_bonds = len(bond_lengths)
    mean_bond_len = np.mean(bond_lengths) if bond_lengths else 0

    return {
        "positions": atom_pos,
```

```
        "n_bonds": n_bonds,
        "mean_bond_len": mean_bond_len
    }

def compare_data(champs_data, rdkit_data, elements):
    """
    Compare CHAMPS and RDKit data.

    Parameters:
    champs_data (dict): Dictionary containing CHAMPS data.
    rdkit_data (dict): Dictionary containing RDKit data.
    elements (list): List of elements.

    Returns:
    pd.DataFrame: DataFrame containing comparison results.
    """
    comparisons = []
    for el in elements:
        if el not in champs_data or el not in rdkit_data:
            print(f"Skipping element {el} due to missing data.")
            continue

        champ = champs_data[el]
        rdkit = rdkit_data[el]

        if len(champ["positions"]) == 0 or len(rdkit["positions"]) == 0:
            print(f"No valid positions for element {el}. Skipping comparison.")
            continue

        bond_len_diff = abs(np.mean(champ["mean_bond_len"]) - rdkit["mean_bond_len"])
        min_len = min(len(champ["positions"]), len(rdkit["positions"]))
        champ_pos = champ["positions"][:min_len]
        rdkit_pos = rdkit["positions"][:min_len]
        euclid_dist = np.linalg.norm(np.array(champ_pos) - np.array(rdkit_pos))

        for j in range(min_len):
            comparisons.append({
                "mol_name": champ["mol_names"][j],
                "symbol": el,
                "n_bonds_CHAMPS": champ["n_bonds"][j],
                "n_bonds_RDKit": rdkit["n_bonds"],
                "x_CHAMPS": champ_pos[j][0],
                "x_RDKit": rdkit_pos[j][0],
                "y_CHAMPS": champ_pos[j][1],
                "y_RDKit": rdkit_pos[j][1],
                "z_CHAMPS": champ_pos[j][2],
                "z_RDKit": rdkit_pos[j][2],
                "bond_len_diff": bond_len_diff,
                "euclid_dist": euclid_dist
            })

    return pd.DataFrame(comparisons)

def main():
    """
    Main function to run the script.
    """
    root = tk.Tk()
```

```
    root.withdraw()
    elements_input = simpledialog.askstring("Input", "Enter a list of chemical elements (comma-separated):",
    elements = [el.strip() for el in elements_input.split(",")][:2]
    valid_elements = ['C', 'H', 'N', 'O', 'F']
    elements = [el for el in elements if el in valid_elements]

    champs_data = {}
    rdkit_data = {}

    for el in elements:
        champs_data[el] = get_champs_data(el, structures)
        mol_names = champs_data[el]["mol_names"]

        for mol_name in mol_names:
            try:
                smiles_code = get_smiles_code(mol_name, structures)
                if smiles_code:
                    rdkit_data[el] = get_rdkit_info(smiles_code)
                    break
                else:
                    print(f"No SMILES code found for {mol_name}")
            except Exception as e:
                print(f"Failed to fetch RDKit info for {mol_name}: {e}")
                continue

    comparison_df = compare_data(champs_data, rdkit_data, elements)
    display_dataframe(comparison_df)

def display_dataframe(df):
    """
    display the dataframe using Tkinter Treeview.
    Parameters:
    df (pd.DataFrame): DataFrame to display.
    """
    root = tk.Tk()
    root.title("Comparison Results")

    frame = ttk.Frame(root)
    frame.pack(fill='both', expand=True)

    tree = ttk.Treeview(frame, columns=list(df.columns), show='headings')
    tree.pack(fill='both', expand=True)

    for col in df.columns:
        tree.heading(col, text=col)
        tree.column(col, width=100)

    for row in df.itertuples(index=False):
        tree.insert('', 'end', values=row)

    root.mainloop()

if __name__ == "__main__":
    main()
```

3 Results

The following is a sample result of the above code:

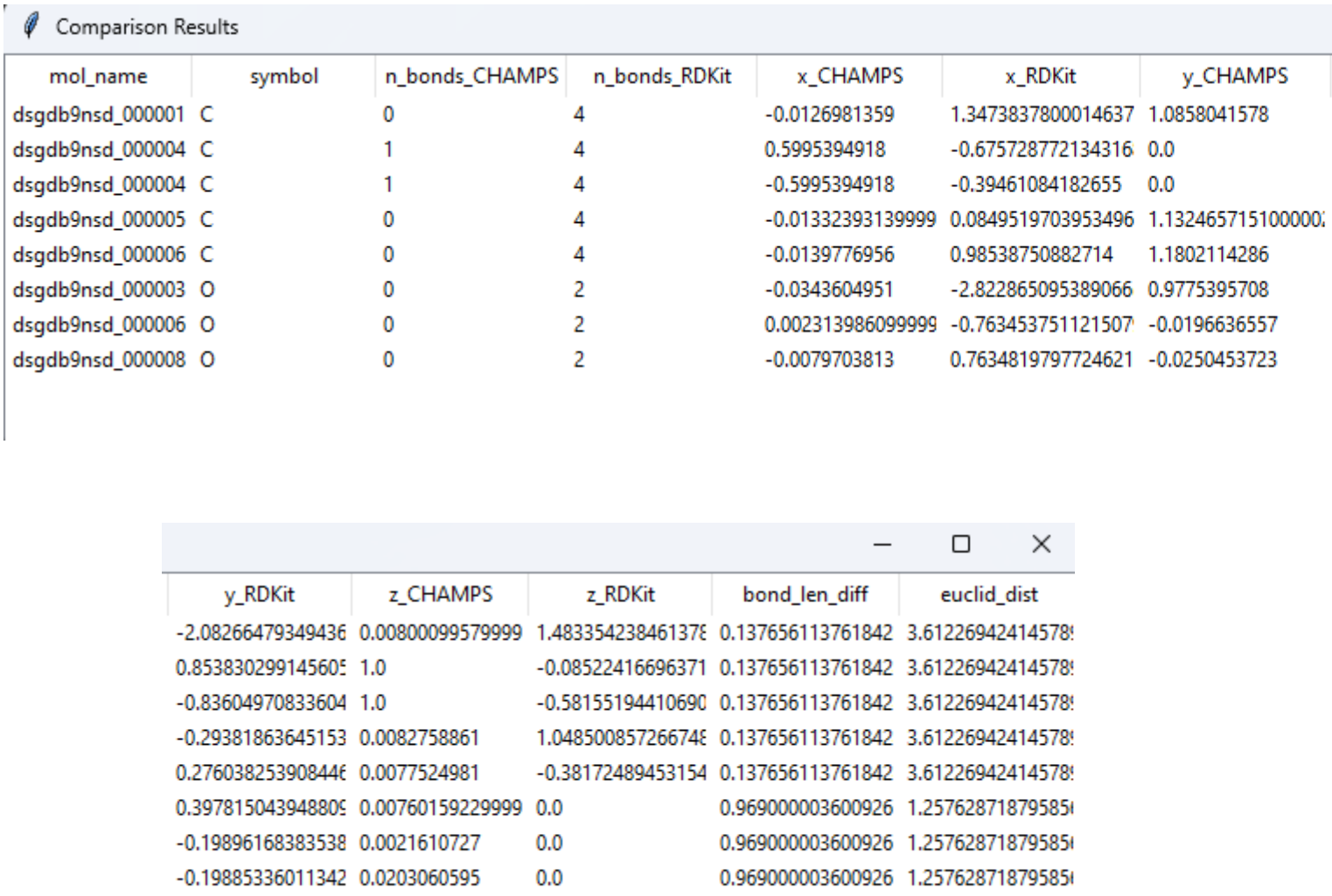

Figure 1: Sample results - Electronic configuration of Elements

Thank you