Hash Map and Markov Chain Report

Jiakun Fan(jfan30@u.rochester.edu)

Module Introduction

There are 3 modules in `lib.rs`

- hashmap_tests: test cases of hash map.
- hashmap_entry_tests: test cases of hash map entry
- markov_chain_tests: test cases of markov chain

Design of Markov Chain

We implement the graph of markov chain by `HashMap<T, HashMap<T, u32>>`

```
pub struct Chain<T> where T: Eq+Hash+Clone {
    graph : HashMap<T, HashMap<T, u32>>,
}
```

We implement several methods for `Chain`

```
pub fn new() -> Chain<T>
// Create a new Chain struct.

pub fn add(&mut self, from: T, to: T)
// Add the transition in graph.

pub fn train(&mut self, sequence: &[T]) -> &mut Chain<T>
// The associated function for training

pub fn most_likely_after(&self, from: &T) -> Option<&T>
//the associated function that returns the most likely successor after
// token, the edge with the highest weight. If such edge is not
// unique, choose one of them

pub fn get(&self, from: &T) -> Option<&HashMap<T, u32>>
// Get possible next states with weights of the state `from`

pub fn generate_from_seed(&self, seed: &T, length: usize) -> Vec<T>
// Generate a  markov chain starting from specified seed with length specified.

pub fn generate(&self, length: usize) -> Vec<T>
// Generate a  markov chain starting from a random seed with length specified.
```

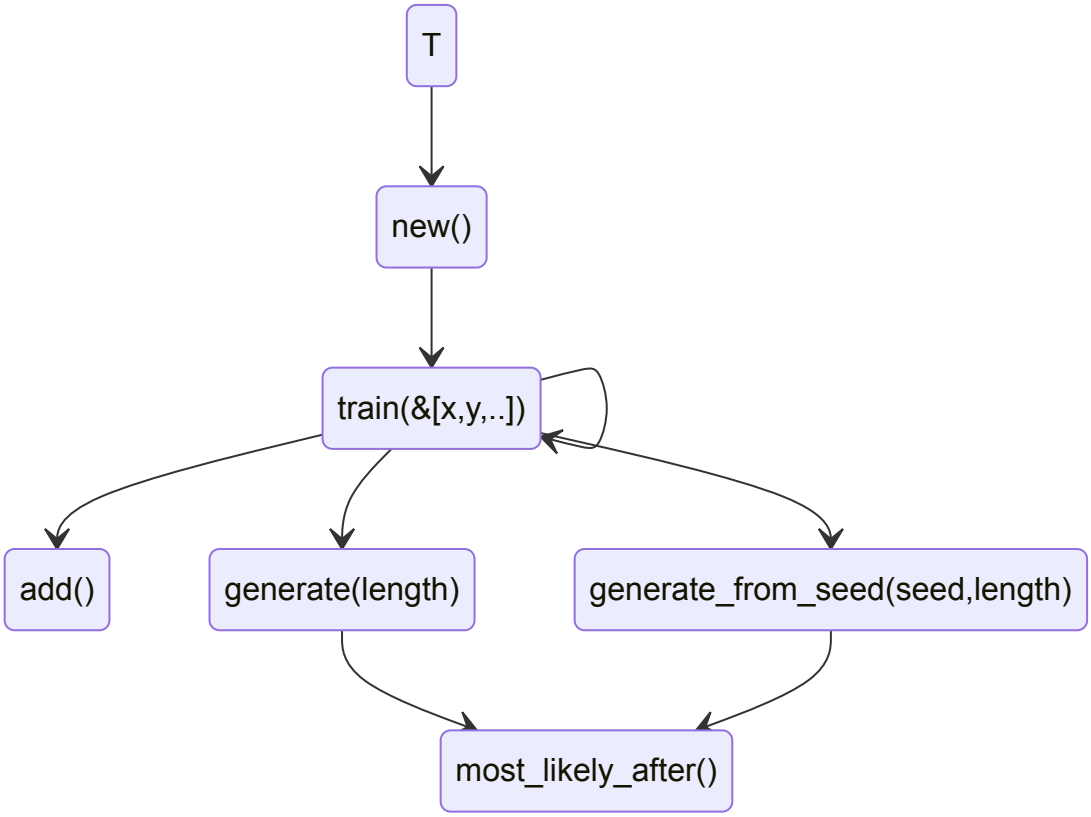
We also implement some additional methods for `Chain<String>`

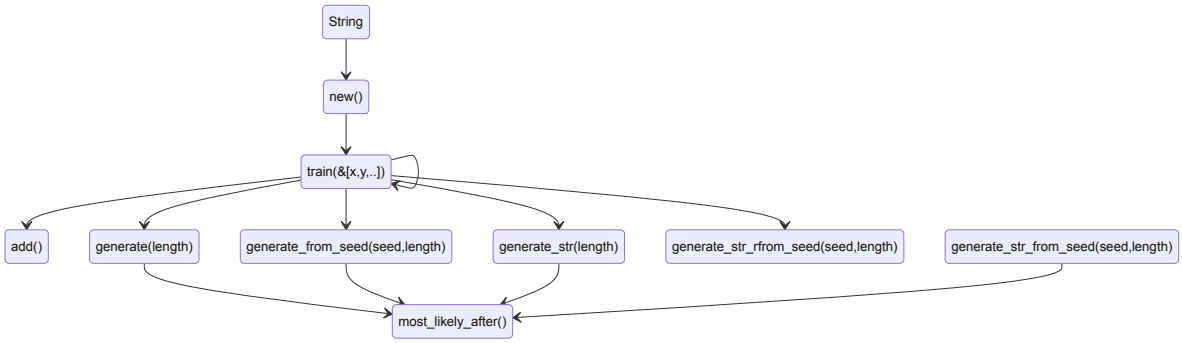
```
pub fn train_str(&mut self, string: &str)
// Training a String (token splited by any numbers of space)

pub fn generate_str(&mut self, length: usize) -> String
// Generate a String with length specified starting from a random seed

pub fn generate_str_from_seed(&mut self, seed: &str,length: usize) -> String
// Generate a String with length specified starting from a specified seed
```

Procedure of generating a Markov Chain.





Reference

Refer the github repo listed in the assignment. According to its function `feed_str()`, `generate_str()`, we learned that we should add additional functions to `Chain<String>` to have a better user experience. So we add functions `generate_str()`, `generate_str_from_seed`, `train_seed()` in our project.