Ahmed Shahab

CS311 Artificial Intelligence

July 25, 2024

Assignment 2 (2.0)

/\*This seems to be the best code I can come up with.

The program initially seems to be unable to predict the political party.

But when it receives and stores multiple scores from multiple users then it is able to predict user’s political party after every answer it receives\*/

import java.io.\*;

import java.util.\*;

class Question {

String text;

List<String> options;

public Question(String text, List<String> options) {

this.text = text;

this.options = options;

}

}

enum PoliticalParty {

DEMOCRAT,

REPUBLICAN,

GREEN,

LIBERTARIAN

}

class DataStorage {

private static final String FILE\_PREFIX = "responses\_";

public static void saveResponse(Map<Integer, Integer> answers, PoliticalParty party) {

String fileName = FILE\_PREFIX + party.name().toLowerCase() + ".txt";

try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName, true))) {

answers.forEach((questionId, answerId) -> {

try {

writer.write(questionId + ":" + answerId + ",");

} catch (IOException e) {

e.printStackTrace();

}

});

writer.newLine();

} catch (IOException e) {

e.printStackTrace();

}

}

public static Map<PoliticalParty, List<Map.Entry<Map<Integer, Integer>, PoliticalParty>>> loadAllResponses() {

Map<PoliticalParty, List<Map.Entry<Map<Integer, Integer>, PoliticalParty>>> allResponses = new HashMap<>();

for (PoliticalParty party : PoliticalParty.values()) {

allResponses.put(party, loadResponses(party));

}

return allResponses;

}

private static List<Map.Entry<Map<Integer, Integer>, PoliticalParty>> loadResponses(PoliticalParty party) {

List<Map.Entry<Map<Integer, Integer>, PoliticalParty>> responses = new ArrayList<>();

String fileName = FILE\_PREFIX + party.name().toLowerCase() + ".txt";

try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {

String line;

while ((line = reader.readLine()) != null) {

Map<Integer, Integer> answerMap = new HashMap<>();

String[] parts = line.split(",");

Arrays.stream(parts)

.map(pair -> pair.split(":"))

.forEach(keyValue -> answerMap.put(Integer.parseInt(keyValue[0]), Integer.parseInt(keyValue[1])));

responses.add(new AbstractMap.SimpleEntry<>(answerMap, party));

}

} catch (IOException e) {

}

return responses;

}

}

class ScoringSystem {

private List<Map.Entry<Map<Integer, Integer>, PoliticalParty>> data;

public ScoringSystem(List<Map.Entry<Map<Integer, Integer>, PoliticalParty>> data) {

this.data = data;

}

public PoliticalParty predict(Map<Integer, Integer> answers) {

Map<PoliticalParty, Integer> partyScores = new HashMap<>();

for (Map.Entry<Map<Integer, Integer>, PoliticalParty> entry : data) {

PoliticalParty party = entry.getValue();

int score = calculateScore(entry.getKey(), answers);

partyScores.put(party, partyScores.getOrDefault(party, 0) + score);

}

if (partyScores.isEmpty()) {

System.out.println("No scores calculated. Unable to predict party.");

return null;

}

return Collections.max(partyScores.entrySet(), Map.Entry.comparingByValue()).getKey();

}

private int calculateScore(Map<Integer, Integer> a, Map<Integer, Integer> b) {

int score = 0;

for (Map.Entry<Integer, Integer> entry : a.entrySet()) {

if (b.containsKey(entry.getKey()) && b.get(entry.getKey()).equals(entry.getValue())) {

score++;

}

}

return score;

}

}

class Survey {

private List<Question> questions = new ArrayList<>();

public void addQuestion(Question question) {

questions.add(question);

}

public List<Question> getQuestions() {

return questions;

}

}

public class Main {

public static void main(String[] args) throws IOException {

/\* Creating the survey \*/

Survey survey = new Survey();

survey.addQuestion(new Question("What should the government do to help the poor?",

Arrays.asList("Make it easier to apply for assistance.",

"Allow parents to use education funds for charter schools.",

"Create welfare to work programs.",

"Nothing.")));

survey.addQuestion(new Question("Which policy do you support the most?",

Arrays.asList("Universal healthcare",

"Lower taxes",

"Stronger military",

"Environmental regulations")));

survey.addQuestion(new Question("What is the best way to improve the education system?",

Arrays.asList("Increase teacher salaries.",

"Reduce class sizes.",

"Expand school choice.",

"Focus on STEM education.")));

survey.addQuestion(new Question("How should the government address climate change?",

Arrays.asList("Invest in renewable energy.",

"Implement carbon taxes.",

"Promote energy efficiency.",

"Deregulate the energy industry.")));

survey.addQuestion(new Question("What should be done about the national debt?",

Arrays.asList("Cut government spending.",

"Increase taxes on the wealthy.",

"Implement balanced budget amendments.",

"Borrow more to stimulate the economy.")));

survey.addQuestion(new Question("How can healthcare be made more affordable?",

Arrays.asList("Expand Medicare/Medicaid.",

"Increase competition among insurers.",

"Reduce regulations on healthcare providers.",

"Implement price controls on drugs and procedures.")));

survey.addQuestion(new Question("What is the most important aspect of immigration reform?",

Arrays.asList("Strengthening border security.",

"Providing a path to citizenship.",

"Increasing legal immigration quotas.",

"Deporting undocumented immigrants.")));

survey.addQuestion(new Question("What should be the government's role in the economy?",

Arrays.asList("Regulate to ensure fair competition.",

"Intervene to reduce income inequality.",

"Promote free markets.",

"Provide safety nets for the unemployed.")));

survey.addQuestion(new Question("How should the government handle cybersecurity threats?",

Arrays.asList("Increase funding for cybersecurity defense.",

"Implement stricter data privacy laws.",

"Encourage private sector innovation.",

"Enhance international cooperation on cyber issues.")));

survey.addQuestion(new Question("What is the best approach to criminal justice reform?",

Arrays.asList("Reduce mandatory minimum sentences.",

"Increase funding for police departments.",

"Focus on rehabilitation and education.",

"Implement harsher penalties for repeat offenders.")));

survey.addQuestion(new Question("How should the government address income inequality?",

Arrays.asList("Raise the minimum wage.",

"Implement universal basic income.",

"Provide tax incentives for businesses to hire locally.",

"Reduce government intervention in the economy.")));

survey.addQuestion(new Question("What should be the focus of foreign policy?",

Arrays.asList("Promoting democracy abroad.",

"Strengthening military alliances.",

"Fostering economic partnerships.",

"Prioritizing national security.")));

survey.addQuestion(new Question("How can the voting process be improved?",

Arrays.asList("Expand mail-in voting.",

"Implement stricter voter ID laws.",

"Increase the number of polling places.",

"Introduce online voting.")));

survey.addQuestion(new Question("What is the best way to support small businesses?",

Arrays.asList("Provide tax breaks and incentives.",

"Increase access to affordable loans.",

"Reduce regulatory burdens.",

"Invest in workforce training programs.")));

survey.addQuestion(new Question("How should the government address homelessness?",

Arrays.asList("Increase funding for affordable housing.",

"Provide more mental health and addiction services.",

"Create job training programs for the homeless.",

"Reduce zoning restrictions on housing development.")));

/\* Load stored user responses \*/

Map<PoliticalParty, List<Map.Entry<Map<Integer, Integer>, PoliticalParty>>> allResponses = DataStorage.loadAllResponses();

List<Map.Entry<Map<Integer, Integer>, PoliticalParty>> responses = new ArrayList<>();

allResponses.values().forEach(responses::addAll);

/\* Scoring System \*/

ScoringSystem scoringSystem = new ScoringSystem(responses);

/\* The actual survey \*/

Scanner scanner = new Scanner(System.in);

Map<Integer, Integer> userAnswers = new HashMap<>();

for (int i = 0; i < survey.getQuestions().size(); i++) {

Question question = survey.getQuestions().get(i);

System.out.println(question.text);

for (int j = 0; j < question.options.size(); j++) {

System.out.println((j + 1) + ". " + question.options.get(j));

}

System.out.print("Your answer: ");

while (true) {

try {

int answer = scanner.nextInt() - 1;

if (answer >= 0 && answer < question.options.size()) {

userAnswers.put(i, answer);

break;

} else {

System.out.print("Invalid answer. Please enter a number between 1 and " + question.options.size() + ": ");

}

} catch (InputMismatchException e) {

System.out.print("Invalid input. Please enter a number: ");

scanner.next();

}

}

/\* Predict party affiliation after each answer \*/

if (i < survey.getQuestions().size() - 1) {

PoliticalParty predictedParty = scoringSystem.predict(userAnswers);

if (predictedParty != null) {

System.out.println("Predicted Political Party: " + predictedParty);

}

}

}

/\* Asking user's political party \*/

System.out.print("Which political party do you affiliate with? ");

PoliticalParty party = null;

while (party == null) {

try {

String partyStr = scanner.next().toUpperCase();

party = PoliticalParty.valueOf(partyStr);

} catch (IllegalArgumentException e) {

System.out.print("Invalid party. Please enter one of the following: DEMOCRAT, REPUBLICAN, GREEN, LIBERTARIAN: ");

}

}

/\* Store user's response \*/

DataStorage.saveResponse(userAnswers, party);

System.out.println("Thank you for completing the survey!");

}

}