MelvinLecoy / gitcode Private

Code Issues Pull requests Actions Projects Security Insights Settings

master

gitcode / p5-ml / main.cpp

Kwan Ting Lau gitcode History

0 contributors

223 lines (192 sloc) 6.12 KB

```cpp
// Project UID db1f506d06d84ab787baf250c265e24e
#include "csvstream.h"
#include <cmath>
#include <cstring>
#include <iostream>
#include <iterator>
#include <map>
#include <set>
#include <string>
#include <utility>

using namespace std;

class Piazza {
private:
  int post = 0;
  int unique_word = 0;
  int word_post = 0;  // number of post of with w
  int label_post = 0; // bumber of post of label C

  map<string, int> word_map;               // number of post of one label
  map<string, int> label_map;              // number of post of one label
  map<pair<string, string>, int> want_map; // C with w    time
  // map<string, string> row;

  set<string> unique_words_file; // set of unique word in the file
  set<string> label_file;        // set of label in the file
  set<string> given_post_word;   // set of given words

public:
  double get_post() { return post; }

  int get_uniq_word() { return unique_word; }

  int get_word_post() { return word_post; }

  int get_label_post() { return label_post; }

  set<string> get_unique_words_file() {
    return unique_words_file;
  } // get the set of unique word

  set<string> get_label_file() {
    return label_file;
  } // get the set of label label in file

  set<string> get_givn_post_word() { return given_post_word; }

  int size_of_unique_words_file() { return unique_words_file.size(); }

  int size_of_label_file() { return label_file.size(); }

  int size_of_given_post_word_file() { return given_post_word.size(); }

  double log_prior(string label) {
    return log(label_map[label] / static_cast<double>(post));
  }

  double log_likelihood(string label, string unique_word) {
    pair<string, string> pair_1;
    pair_1.first = label;
    pair_1.second = unique_word;
    if (want_map.find(pair_1) ==
        want_map.end()) { // doesn't appear in a post with label C
      if ((word_map.find(unique_word) ==
           word_map.end())) { // doesn't appear in training set
        return log(1 / static_cast<double>(post));
      } else {
        return log(word_map[unique_word] / static_cast<double>(post));
      }
    }
    return log(want_map[pair_1] / static_cast<double>(label_map[label]));
  }

  // log probability score, set is the set of words from a post
  double log_probability_score(string label, set<string> words) {
    double score = log_prior(label);
    for (auto word : words) {
      score += log_likelihood(label, word);
    }
    return score;
  }

  set<string> unique_words(const string &str) {
    istringstream source(str);
    set<string> words;
    string word;
    // Read word by word from the stringstream and insert into the set
    while (source >> word) {
      words.insert(word);
    }
    return words;
  }

  void train(csvstream &csvin, bool debug) {
    map<string, string> row; // store the csv
    if (debug) {
      cout << "training data:" << endl;
    }
    while (csvin >> row) {
      if (debug) {
        cout << "  label = " << row["tag"] << ", content = " << row["content"]
             << endl;
      }
      unique_words_file = unique_words(row["content"]);
      post++;
      label_map[row["tag"]]++;
      for (auto b : unique_words_file) {
        word_map[b]++;
        pair<string, string> pair_2;
        pair_2.first = row["tag"];
        pair_2.second = b;
        ++want_map[pair_2];
      }
    }
    cout << "trained on " << post << " examples" << endl;
    if (debug) {
      cout << "vocabulary size = " << word_map.size() << endl << endl;
      cout << "classes:" << endl;
      for (auto x : label_map) {
        cout << "  " << x.first << ", " << x.second
             << " examples, log-prior = " << log_prior(x.first) << endl;
      }
      cout << "classifier parameters:" << endl;
      for (auto y : want_map) {
        cout << "  " << y.first.first << ":" << y.first.second
             << ", count = " << y.second << ", log-likelihood = "
             << log_likelihood(y.first.first, y.first.second) << endl;
      }
    }

    cout << endl;
  } // #post
    // #label  num
    // #unique words num
    // label  word num
    // sum of unique words word_appear.size()

  pair<string, double> predict(set<string> words) {
    double score = -INFINITY;
    string label = "";
    pair<string, double> pair_3;
    for (auto element : label_map) {
      string label_1 = element.first;
      double score_1 = log_probability_score(label_1, words);
      if (score_1 > score) {
        score = score_1;
        label = label_1;
        pair_3.first = label_1;
        pair_3.second = score_1;
      }
    }
    return pair_3;
  }

  void test(csvstream &csvin) {
    pair<string, double> test_pair;
    map<string, string> row;
    int correct_predict = 0;
    int count = 0;

    cout << "test data:" << endl;
    while (csvin >> row) {
      string tag = row["tag"];
      string content = row["content"];
      test_pair = predict(unique_words(content));
      cout << "  correct = " << tag;
      cout << ", predicted = " << test_pair.first;
      cout << ", log-probability score = " << test_pair.second << endl;
      cout << "  content = " << content << endl << endl;
      if (test_pair.first == tag) {
        correct_predict++;
      }
      count++;
    }
    cout << "performance: " << correct_predict << " / ";
    cout << count << " posts predicted correctly" << endl;
  }
};

int main(int argc, char *argv[]) {
  cout.precision(3);

  if ((argc != 3) && (argc != 4)) {
    cout << "Usage: main.exe TRAIN_FILE TEST_FILE [--debug]" << endl;
    return -1;
  }

  if ((argc == 4) && (string(argv[3]) != "--debug")) {
    cout << "Usage: main.exe TRAIN_FILE TEST_FILE [--debug]" << endl;
    return -1;
  }

  Piazza piazza;

  try {
    string s_1 = argv[1];
    csvstream csvin(s_1);
  } catch (const exception &e) {
    cout << "Error opening file: " << argv[1] << endl;
    return -1;
  }

  try {
    string s_2 = argv[2];
    csvstream csvin1(s_2);
  } catch (const exception &e) {
    cout << "Error opening file: " << argv[2] << endl;
    return -1;
  }
  string s_1 = argv[1];
  csvstream csvin(s_1);
  string s_2 = argv[2];
  csvstream csvin1(s_2);

  bool debug = false;
  if (argc == 4 && string(argv[3]) == "--debug") {
    debug = true;
  }

  piazza.train(csvin, debug);
  piazza.test(csvin1);
}
```

Give feedback