

Emoji converter

Limbaje Formale si Translatoare



VOICU LAURA-LUISA

30236

CTI-UTCN

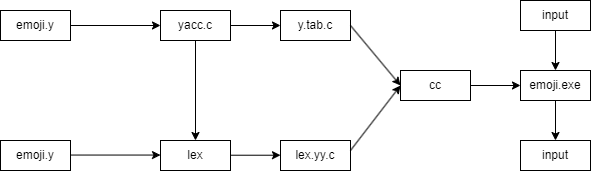
1. Descrierea Proiectului
   1. Cerinta proiectului

Proiectul “Emoji converter” este un program ce permite utilizatorilor sa introduca texte in limba engleza si sa reprezinte cuvintele cheie sub forma de emoji. Scopul proiectului este de a oferi o metoda interactiva si universala de comunicare.

Cerintele proiectului include:

* 1. Traducerea cuvintelor in emoji: Programul trebuie sa traduca cuvinte in emoji-ul cel mai sugestiv. De exemplu, cuvantul “dog” trebuie sa fie tradus in emoji-ul "🐶".
  2. Interpretarea expresiilor si frazelor: In plus fata de traducerea cuvintelor individuale, programul poate sa interpreteze si sa traduca expresii si fraze. De exemplu, fraza “Birds sing happily” va fi tradusa “🐦 🎤 😊”. Cu toate astea, expresia “Happy birthday” e considerata o expresie si poate fi interpretata ca un singur emoji: “🥳”
  3. Gestionarea cardinalitatii: Programul poate sa reprezinte un numar de obiecte prin repetarea emoji-ului corespunzator obiectului : “ 3 birds” e convertit in “🐦🐦🐦”
  4. Gestionarea cuvintelor sinonime: Programul are alocata o lista de sinonime ce e folosita atunci cand un utilizator introduce un cuvant ce nu face parte din cuvintele de baza. Cuvantul “kitten” este sinonim cu “cat” si, prin urmare, ambele vor fi traduse prin acelasi emoji: “🐱”
  5. Gestionarea expresiilor neintelese: Acand un cuvant necunoscut este emis de utilizator, se va incerca potrivirea acestuia cu unul dintre campurile lexicale pe care programul le are integrate si reprezentarea lui prin intermediul a catorva emoji astfel incat sa sugereze cuvantul descris. Exemple:
     + “thing” - “(🌳 💻 🚗)”
     + “meal” – “(🍔 🍟 🍕)”
     + “profession” – “(💼 🔨 💻)”
     + “human” – “(👨 👩 🧒) “

1. Detalii Tehnice
   1. Tokenii folositi in program sunt definiti in sectiunea de declaratii a fisierului Yacc, avand asociat un tip si o valoare optionala
      * **<subject\_flag>**: reprezinta Tokeni pentru substantive comune si pronume, ele reprezentand subiectul unei propozitii
      * **<adjective\_flag>**: reprezinta Tokenii ce sunt rezervati pentru adjective
      * **<verb\_flag> , <adverb\_flag>**: reprezinta Tokenii pentru verbe si adverbe
   2. Gramaticile definesc structura sintactica a limbajului, in cazul de fata ilustrand compozitia poeziilor. Regulile gramaticale definite:
      * **Poema**: reprezinta intreaga poezie, formata din **linii**
      * **Linii:** O linie este formata din una sau mai multe propozitii urmate de new-line.
      * **Propozitii:** O propozitie poate contine un subiect (substantiv sau pronume personal), adjective, verbe si adverbe in diverse cominatii
      * **Subiectele:** pot fi pronume personale, substantive simple sau combinatii intre substantive si adjective numerale
      * **Verbe:** Verbele descriu actiuni comune (eat, drink, sing, etc) si pot fi detectate verbe la timpul trecut
      * **Adverbe:** folosite pentru insusiri asociate verbelor (sadly, happily, etc)
   3. Structura programului
      * Analizatorul Lexical: Scopul lui e de a tokeniza textul dat de utilizator. Acesta imparte textul in caractere pentru a detecta eventualele cuvinte cheie si a produce un token corespunzator care va fi trimis Analizatorului Sintactic.
      * Analizatorul Sintactic: Are drept scop analizarea structurii sintactice a tokenilor primiti din LEX ce creeaza un arbore de analiza, bazandu-se pe regulile gramaticale definite de programator. Atunci cand se detecteaza o secventa valida, Yacc va efectua rezulile specificate in regulile gramaticale corespunzatoare.



* + - Configurarea mediului de dezvoltare:
      1. Instalare Flex si Bison
      2. sudo apt-get update
      3. sudo apt-get install flex bison
         1. Fisierele Lex si Yacc

In proiect exista 2 fisiere “.l” si “.y”. Acestea contin programul responsabil de preluarea cuvintelor de la utilizator, detectarea tokenilor, prelucrarea lor conform gramaticilor, executarea operatiilor necesare si transmiretea textului convertit utilizatorului .

* + - 1. Fisierul input

In proiect exista un fisier “.txt” in care sunt scrise diverse texte: poezii, cuvinte sau propozitii ce acopera anumite cazuri speciale pentru a evidentia functionalitatile programului. Utilizatorul poate adauga si alte fisiere cu propriul text, cu mentiunea ca la rulare va fi specificat acel fisier

* + - 1. Rularea programului

Compilarea programului:

* + - * 1. lex emoji.l
        2. yacc emoji.y
        3. gcc y.tab.c -ll -ly

ii. Rularea programului:

1. ./a.out < input.txt
2. Detalii de implementare
   * + Implementarea componentelor principale in Lex:

Pe langa partea reguli (ce reprezinta cuvintele cheie din texturi) , ca si subrutine, programul contine urmatoarele functii:

1. Identificarea sinonimelor: Functia ‘get\_main\_token’ e folosita pentru a gasi sinonime si a inlocui cuvintele necnoscute (‘UNKNOWN\_WORD’) cu tokenul principal corespunzator
2. Transformarea in litere mici: Functia “to\_lower” transforma toate cuviintele in litere mici pentru a uniformiza intrarea, asigurand ca procesarea ulterioara nu e afectata de variatiile de caz.
   * + Subrutinele principale din Yacc:

Functia “print\_emoji”

* **Scop**: Aceasta functie primeste un emoji si un contor, adaugand fiecare repetare a emoji-ului in buffer-ul principal folosind functia save\_buffer.

Functia ‘process\_expression’

* **Scop**: Afiseaza o expresie formata dintr-o secventa de cuvinte.
* **Funcționalitate**:
* Primesc un array de cuvinte și numărul de cuvinte.
* Iterează prin cuvintele date și le afișează pe ecran, pentru a arăta expresia completă.

void process\_expression(char \*\*words, int num\_words) {

    printf("Expression: ");

    for (int i = 0; i < num\_words; i++) {

        printf("%s ", words[i]);

    }

    printf("\n");

}

Functia ‘save\_buffer’

* **Scop**: Afiseaza o expresie formată dintr-o secvență de cuvinte.
* **Functionalitati:**
* Primesc un array de cuvinte și numărul de cuvinte.
* Iterează prin cuvintele date și le afișează pe ecran, pentru a arăta expresia completă.

void process\_expression(char \*\*words, int num\_words) {

    printf("Expression: ");

    for (int i = 0; i < num\_words; i++) {

        printf("%s ", words[i]);

    }

    printf("\n");

}

Functia ‘save\_buffer”

* **Scop**: Salveaza cuvintele sau emoji-urile în buffer-ul principal.
* **Forme:**
* Primește un cuvant sau un emoji ca argument.
* Concateneaza cuvantul sau emoji-ul în buffer-ul principal, adaugand un spatiu dupa fiecare cuvant pentru a separa elementele.

Functia ‘replace\_expressions\_in\_buffer ‘

* **Scop**: Inlocuieste anumite combinatii de emoji-uri și cuvinte din buffer cu alte expresii predefinite.
* **Functionalitati:**
* Creeaza un nou buffer pentru a stoca rezultatele procesarii.
* Imparte buffer-ul initial în cuvinte individuale.
* Verifica fiecare cuvânt și combinatiile de cuvinte pentru a identifica si înlocui expresiile predefinite (de exemplu, 😃 urmat de 🎂 devine 🥳).
* Dupa procesarea tuturor cuvintelor, copiaza continutul noului buffer în buffer-ul initial.

void replace\_expressions\_in\_buffer() {

    char \*new\_buffer = malloc(strlen(buffer) + 1);

    if (new\_buffer == NULL) {

        fprintf(stderr, "Memory allocation failed\n");

        return;

    }

    new\_buffer[0] = '\0';

    char \*expression = strtok(buffer, " ");

    while (expression != NULL) {

        if (strcmp(expression, "😃") == 0) {

            char \*next\_token = strtok(NULL, " ");

            if (next\_token != NULL && strcmp(next\_token, "🎂") == 0) {

                strcat(new\_buffer, "🥳 ");

                expression = strtok(NULL, " ");

                continue;

            }

        } else if (strcmp(expression, "merry") == 0) {

            char \*next\_token = strtok(NULL, " ");

            if (next\_token != NULL && strcmp(next\_token, "🎅🏻") == 0) {

                strcat(new\_buffer, "🎄 ");

                expression = strtok(NULL, " ");

                continue;

            }

        } else if (strcmp(expression, "welcome") == 0) {

            char \*next\_token = strtok(NULL, " ");

            if (next\_token != NULL && strcmp(next\_token, "home") == 0) {

                strcat(new\_buffer, "🏡 ");

                expression = strtok(NULL, " ");

                continue;

            }

        } else if (strcmp(expression, "👍") == 0) {

            char \*next\_token = strtok(NULL, " ");

            if (next\_token != NULL && strcmp(next\_token, "🍀") == 0) {

                strcat(new\_buffer, "🤞 ");

                expression = strtok(NULL, " ");

                continue;

            }

        }

        strcat(new\_buffer, expression);

        strcat(new\_buffer, " ");

        expression = strtok(NULL, " ");

    }

    strcpy(buffer, new\_buffer);

    free(new\_buffer);

}

Functia ‘replace\_unknown\_word\_with\_emojis ‘

* **Scop**: Inlocuiește cuvintele necunoscute cu o secventa de emoji-uri.
* **Functionalitati:**
* Primeste un cuvant necunoscut.
* Cauta in array-ul de câmpuri lexicale pentru a gasi cuvantul necunoscut.
* Daca cuvantul este gasit, returneaza un sir de emoji-uri asociat acelui cuvant.
* Daca cuvantul nu este gasit, returneaza cuvantul original.

char \*replace\_unknown\_word\_with\_emojis(char \*word) {

    char \*emojis = malloc(4 \* 3);

    emojis[0] = '\0';

    printf("unknow word: %s\n", word);

    for (int i = 0; i < sizeof(lexical\_fields) / sizeof(lexical\_fields[0]); i++) {

        for (int j = 0; j <lexical\_fields[i].num\_keywords; j++) {

           if (strcmp(word, lexical\_fields[i].keywords[j]) == 0) {

                strcat(emojis, "(");

                strcat(emojis, lexical\_fields[i].emojis[0]);

                strcat(emojis, " ");

                strcat(emojis, lexical\_fields[i].emojis[1]);

                strcat(emojis, " ");

                strcat(emojis, lexical\_fields[i].emojis[2]);

                strcat(emojis, ") ");

                return emojis;

            }

        }

    }

    return word;

}

1. Testare si validare

4.1 Cazuri de test si Rezultate

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | the bird flies quickly  a big dog runs happily  she dances gracefully  we study carefully    the cat jumps proud  an old tree sleeps gracefully  3 birds sing loudly  they listen quietly    I draw patiently  the moon watch curiously at me  I paint rarely  human brave like you |  | the 🐦 ✈️ 🏃‍  a 💪 🐶 🏃 😊  👧 💃 🩰  👫 📚 🔍    the 🐱 🤸 🏆  an (👨 👩 🧒) 🌳 😴 🩰  🐦 🐦 🐦 🎤 🔊  👬 👂 🤫    👤 ✏️ ⏳  the 🌜 👀 🔍 at 👤  👤 🎨 🔀  (👨 👩 🧒) 🦸‍ like 👥 |
|  |  |  |

1. Cazuri speciale
2. Recunoasterea unor expresii

A black background with white text

Description automatically generated A screenshot of a computer screen

Description automatically generated

1. Afisarea unui numar de substantive

A screenshot of a game

Description automatically generated

1. Inlocuirea unui cuvant necunoscut cu un sinonim

A screenshot of a computer

Description automatically generated

1. Reprezentarea unui cuvant necunoscut cu ajutorul cuvintelor din campul lexical cunoscute

A screenshot of a phone

Description automatically generated

1. Reprezentarea verberlor la forma past tense sau a substantivelor la forma plurala

A screenshot of a video game

Description automatically generated A black background with white text and birds and cats

Description automatically generated

1. Evaluare
   1. Puncte forte

* Interpretarea diverselor forme ale unui cuvant (plural, singular ; timp trecut, prezent)
* Combinatii de cuvinte ce vor fi recunoscute drept expresii sunt reprezentate printr-un singur emoji, deci nu se va converti fiecare cuvant in parte.
* Cuvintele necunoscute sunt trecute prin 2 etape ce incearca posibila potrivire a acestora cu tokeni cunoscuti de program
  + - 1. Cautarea in Lex unor sinonime care ar putea sa inlocuiasca cuvantul; daca nu se gaseste nici un cuvant potrivit, acesta este pasat in Yacc ca si ‘UNKNOWN\_WORD’
      2. Pentru cuvintele cu token-ul ‘UNKNOWN\_WORD’ se va mai face o verificare mai generala decat in cazul anterior, si anume cautarea cuvantului intr-o lista sugestiva de alte cuvinte ce apartin unor campuri lexicale (plant, animal, job, etc) . Daca nici aici nu se poate realiza conversia, cuvantul este transmis nemodificat in output.
  1. Puncte slabe
     + 1. Anumite cuvinte sunt asociate aceluiasi emoji

“beautiful” 🌸 ; “flower” 🌸

* + - 1. Cuvintele de legatura “the”, “at”, “with” nu au un emoji corespondent , prin urmare se vor transmite la output nemodificate
      2. Structura versurilor este limitata la doar cateva configuratii, motiv pentru care uneori programul nu recunoaste unele input-uri si va considera unele cuvinte cunoscute ca si “UNKNOWN\_WORDS”
      3. Semnele de punctuatie sunt ignorate
  1. Devzoltari ulterioare
     + Extinderea programului in ceea ce priveste dictionarul de cuvinte necunoscute, gasirea unei metode de reprezentare mai eficiente a cuvintelor care nu au un emoji corespondent
     + Integrarea mai multor limbi, programul poate sa traduca texte din mai multe limbi, fara a declara implicit acestea ca si tokeni, ci cuvintele sa fie traduse inainte de a fi parsate in limba engleza
     + Integrarea programului intr-o aplicatie de tip lyrics pentru cantece: cuvintele sunt convertite in emoji si afisate pe ritmul cantecului, sau repetarea emoji-ului pentru notele mai inalte din cantec.

1. Concluzie

Prin dezvoltarea proiectului Emoji Converter, a fost realizata o solutie interactiva si eficienta pentru traducerea textelor in emoji-uri, indeplinind astfel cerintele initiale si oferind utilizatorilor o modalitate universala de comunicare. Procesul de lucru a oferit oportunitatea de a invata si de a aplica concepte complexe legate de analiza lexicala si sintactica. Ca si ulterioare imbunatatiri, programul poate fi imbunatatit prin extinderea dictionarului de cuvinte, integrarea unui suport pentru mai multe limbi si a face aplicatia accesibila utilizatorulor .