Compiler Design (CS6109) PROJECT DOCUMENTATION Team number:-3

DATE:-12/10/2020

Predicting and Classifying Interests of Users based on Direct Interactions

Vivek Ramkumar, 2018103082
Kariketi Tharun Reddy, 2018103034
G. R. Srikanth, 2018103603

Introduction:

The basis of our project is to provide a textual analysis of the interactions between multiple users, and predict their interests, as well as categorize them. This is an important function of recommendation systems, that is, to offer a personalized experience for each user, by understanding their preferences. Direct interactions between users will be subject to lexical, syntax and semantic analysis, from which we will obtain their preferences to items, as well as actions and hobbies.

List of Modules(Brief):

We will employ the standard principles of compiler design, up to intermediate code generation. Lexical, syntax and semantic analysis will be used in the following order to analyze the input text:

- Identify keywords(chicken, leather bag, smart phone) and verbs in the continuous sense(eg. eating, playing) and such, using lexical analysis.(1)
- Grouping identifiers by users, check for proper suffixes to the tenses, removing unnecessary characters, and other such error checks, by using syntax analysis.(2)
- Check for meaning to thus obtained terms and interpreting them, by using semantic analysis.(3,4)

The result will be an organized collection of information obtained from the interaction, showing the preferences and their respective categories.

<u>**How it works**:</u>

Vivek: *I like using the mongoose cricket bat to play with my friends.*
Tharun: *I enjoy drinking coffee and playing carrom board with my family.*
Srikanth: *I find table tennis a refreshing activity to play with my family as well.*

The interactions between these three users will be sent through the three analyzers in respective order(lexical, syntax and semantic) and their preferences will be classified and obtained as follows:

Please note: This is **not an accurate depiction of the final result.**

Vivek -> Item: mongoose, bat
Interest: animal, cricket
Action: using, play
Classification: Sports, Pets

Tharun-> Item: coffee, carrom, board
Interest: coffee, carrom board
Action: drinking, playing
Classification: Beverages, Sports

Srikanth-> Item: table, tennis
Interest: tables, tennis
Action: play, refreshing
Classification: Sports

Common topics: Sports

Common Recommendation: Sports kits, cricket bats, carrom board sets, Table Tennis bats and balls, etc.

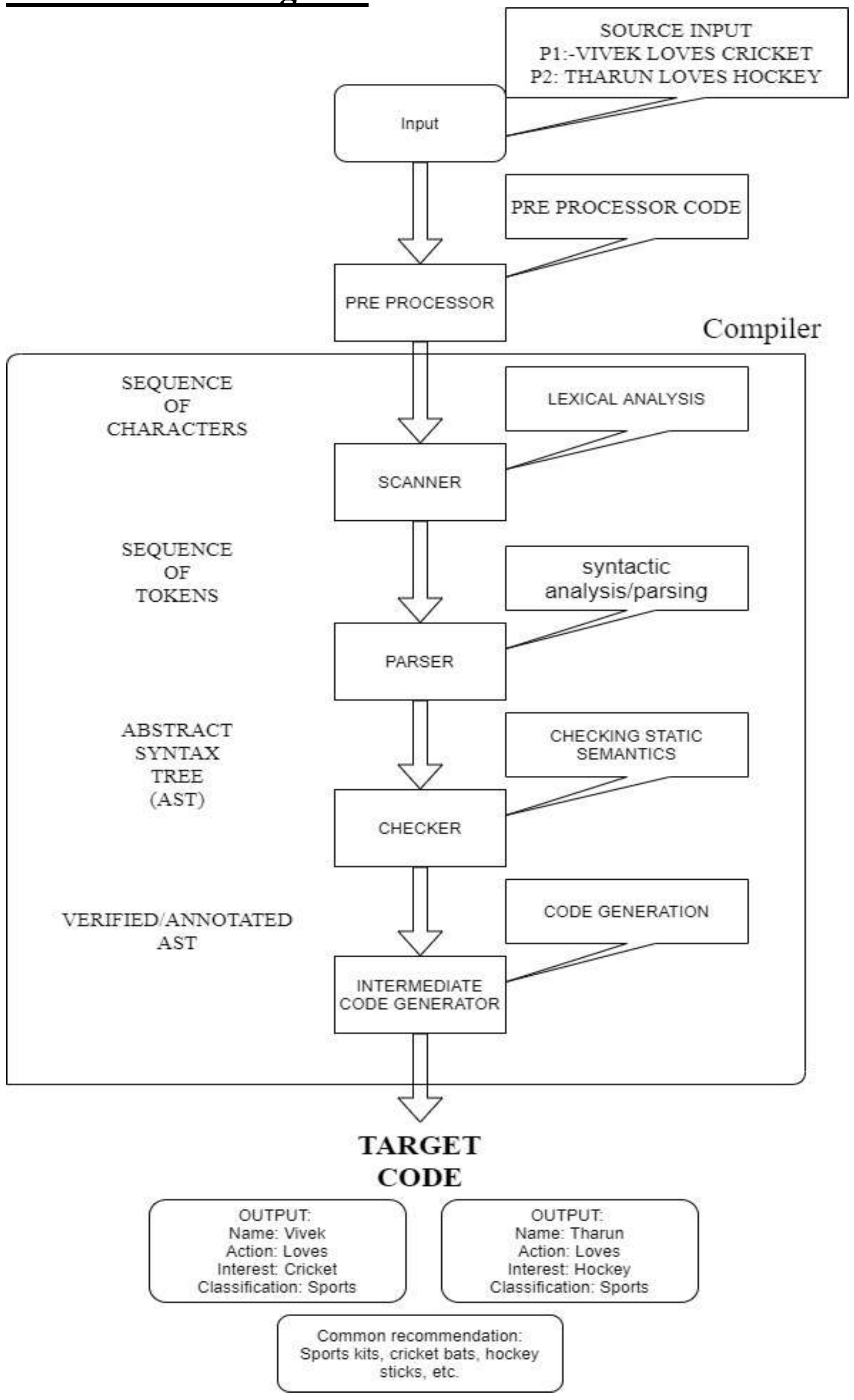
If we evaluate Vivek alone on his statement, it could recommend animals, but they are eliminated on the basis of a common topic, as neither Srikanth nor Tharun talk about animals, but they all talk about sports.

Literature Survey/Name Related Work:

Recommender systems are defined as a decision making strategy for users under complex information environments . Also, recommender systems were defined from the perspective of E-commerce as a tool that helps users search through records of knowledge which is related to users' interest and preference . Recommender system was defined as a means of assisting and augmenting the social process of using recommendations of others to make choices when there is no sufficient personal knowledge or experience of the alternatives. Recommender systems handle the problem of information overload that users normally encounter by providing them with personalized, exclusive content and service recommendations.
Recently, various approaches for building recommendation systems have been developed, which can utilize either collaborative filtering, content-based filtering or hybrid filtering . Collaborative filtering technique is the most mature and the most commonly implemented. Collaborative filtering recommends items by identifying other users with similar taste; it uses their opinion to recommend items to the active user. Collaborative recommender systems have been implemented in different application areas. GroupLens is a news-based architecture which employed collaborative methods in assisting users to locate articles from massive news database .Ringo is an online social information filtering system that uses collaborative filtering to build users profile based on their ratings on music albums . Amazon uses topic diversification algorithms to improve its recommendation .
The system uses collaborative filtering method to overcome scalability issue by generating a table of similar items offline through the use of item-to-item matrix. The system then recommends other products which are similar online according to the users' purchase history. On the other hand, content-based techniques match content resources to user characteristics. Content-based filtering techniques normally base their predictions on user's information, and they ignore contributions from other users as with the case of collaborative techniques . Fab relies heavily on the ratings of different users in order to create a training set and it is an example of content-based recommender system. Some other systems that use content-based filtering to help users find information on the Internet include Letizia .
The system makes use of a user interface that assists users in browsing the Internet; it is able to track the browsing pattern of a user to predict the pages that they may be interested in. Pazzani et al. designed an intelligent agent that attempts to predict which web pages will interest a user by using naive Bayesian classifier. The agent allows a user to provide training instances by rating different pages as either hot or cold. Jennings and Higuchi describe a

neural network that models the interests of a user in a Usenet news environment.

Architecture Diagram:


SOURCE INPUT
P1:-VIVEK LOVES CRICKET
P2: THARUN LOVES HOCKEY
Input
PRE PROCESSOR CODE
PRE PROCESSOR
Compiler
SEQUENCE OF CHARACTERS
LEXICAL ANALYSIS
SCANNER
SEQUENCE OF TOKENS
syntactic analysis/parsing
PARSER
ABSTRACT SYNTAX TREE (AST)
CHECKING STATIC SEMANTICS
CHECKER
VERIFIED/ANNOTATED AST
CODE GENERATION
INTERMEDIATE CODE GENERATOR
TARGET CODE
OUTPUT: Name: Vivek Action: Loves Interest: Cricket Classification: Sports
OUTPUT: Name: Tharun Action: Loves Interest: Hockey Classification: Sports
Common recommendation: Sports kits, cricket bats, hockey sticks, etc.

Pseudocode: (Simplified Code)

Lexical Analysis: (Identify and classify words)

```
        array of strings: fruit_db = {“apple”,”banana”}
                                       pet_db = {“cat”,”dog”}
        string: fruits, pets, users


        [a-zA-Z] if(yytext is in fruit_db){
                            append yytext to fruits
                            append ‘$’ to fruits as a delimiter

                    else if(yytext is in pet_db){
                            append yytext to fruits
                            append ‘$’ to pets as a delimiter

                    else append rest to users and delimit with ‘-’

        if newline, increment user count and a

        ignore all other symbols

        print fruits, pets
```

Syntax Analysis: (Allocation to users)

```
temp_counter = 1

while character in fruits, pets, etc. is not null
        if ‘-’ in fruits, pets, etc.
                increment temp_counter
                if temp_counter is more or equal to user_count
                        print the user’s name(or append user’s name to array)
        else if ‘$’ in fruits, pets, etc.
                append “ “ to stdout, or print “ “(or append “ “ to array)
```

```
    else  append remaining characters to array, or print them
```

<u>Semantic Analysis:</u> (Obtain the meaning and recommend)

```
for each user:
    if a string is present in their categorical_interests:
        check the sample database for that category
        (match using strcasecmp)

        if sample database is present:
            return string of random index from sample_database

        store categorical_interest in a string
        if it matches for another user, then return a flag for that user
```

After parsing all user interests and allocating the samples,

```
for each user:

    print user's name

    while categorical_interests string array is not null:
        print each interest
        print recommended_sample

        if a flag for this user is true for an interest:
            print common interest with (user)
            print common recommended sample
```

<u>Note:</u> So far, 2 modules have been implemented. The third and fourth(part of semantic analysis) shown here is yet to be implemented.

List of modules(in detail):

Keyword Identification and classification:

We identified keywords from the input text file, by comparing them with an existing database of keywords. There are different databases for each category of words. The different databases are in the form of arrays of strings.

All tabs, spaces and special symbols are ignored. The only delimiter with an exception is the newline, by which users are identified. Since the input is in the form of a conversation, we intend to understand short text messages and grasp their meaning. This is the reason we do not use messages beyond one line.

When it matches alphabets, i.e [a-zA-Z], we then get yytext and compare it with each of the available databases, for each category. If it is found in any database, then it is promptly appended to a new string, along with a dollar sign('$'), to act as a delimiter between words. However, if it is not found in any database, it is considered an unique word, i.e a user, and regarded as thus, for the rest of the process. If it otherwise matches a newline[\n], then we increment the number of users by one.

It is important to consider that there is sufficient handling of exceptions, such as CabbAge, for which we disregard the cases, using strcasecmp(), so that it can match with the string in the database exactly as is.

Allocating to users:

After we obtain the respective intermediate identification strings, we detect the delimiters we had previously appended, i.e '$', and '-'. The dollar sign, which is used to separate the identified words in each category, was detected and we printed each character that was parsed until the dollar sign, followed by a space(" ").

When '-' was encountered, we incremented a temporary counter. This temporary counter had a fail-safe, that is, to not print out the next user unless it was within the range of the number of users.

Post-allocation, we have obtained what each user is interested in, and we will attach meaning to it using semantic analysis.

<u>The true recommendation</u>:
The core point of recommender systems is to understand the preferences for each user and recommend them the same, or similar preferred items. However, we are monitoring a particular aspect of the user, that is, their interactions. This comes in the form of short, one-line quick messages that you often see in social messaging applications such as WhatsApp.

As we have now obtained the preferences for each user, we will try to find what lies in common with them. If any such common preference, be it any category, we will then recommend a sample of that category to them, from an existing data base of samples.

If no such category exists, then we will take a look at the individual preferences of the user, and recommend samples solely based on what each user wants. All this is performed using simple string comparison functions, along with the use of delimiters. There will, of course be existing databases in the form of arrays of strings that we will refer to.

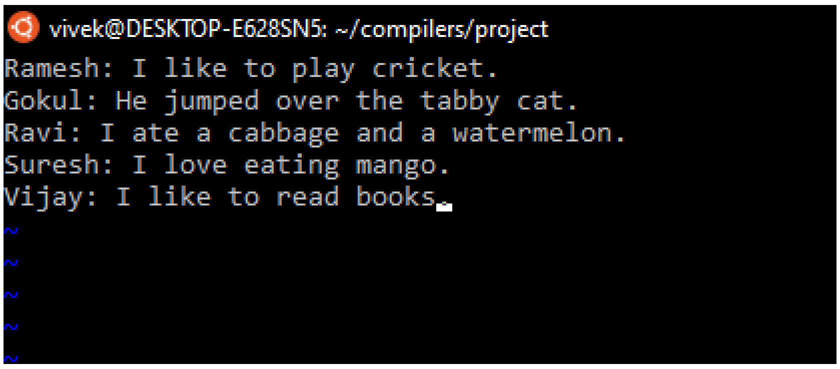
<u>Error handling</u>:
There are certain loop-holes to this system, most of which involve different forms of the words, and consideration of past, present and future tenses. Another such loop-hole is the detection of plurality and singularity in the sentences.

One example: Ramesh **ate** the **bananas**.

We cannot simply append ‘s’, as there are certain words which do not employ that principle, one being ‘formulae’ as a plural of ‘formula’. These are easily fixed by added required words into the respective categorical databases, where ‘ate’ would be added to a database of verbs, and bananas to a database of fruits.

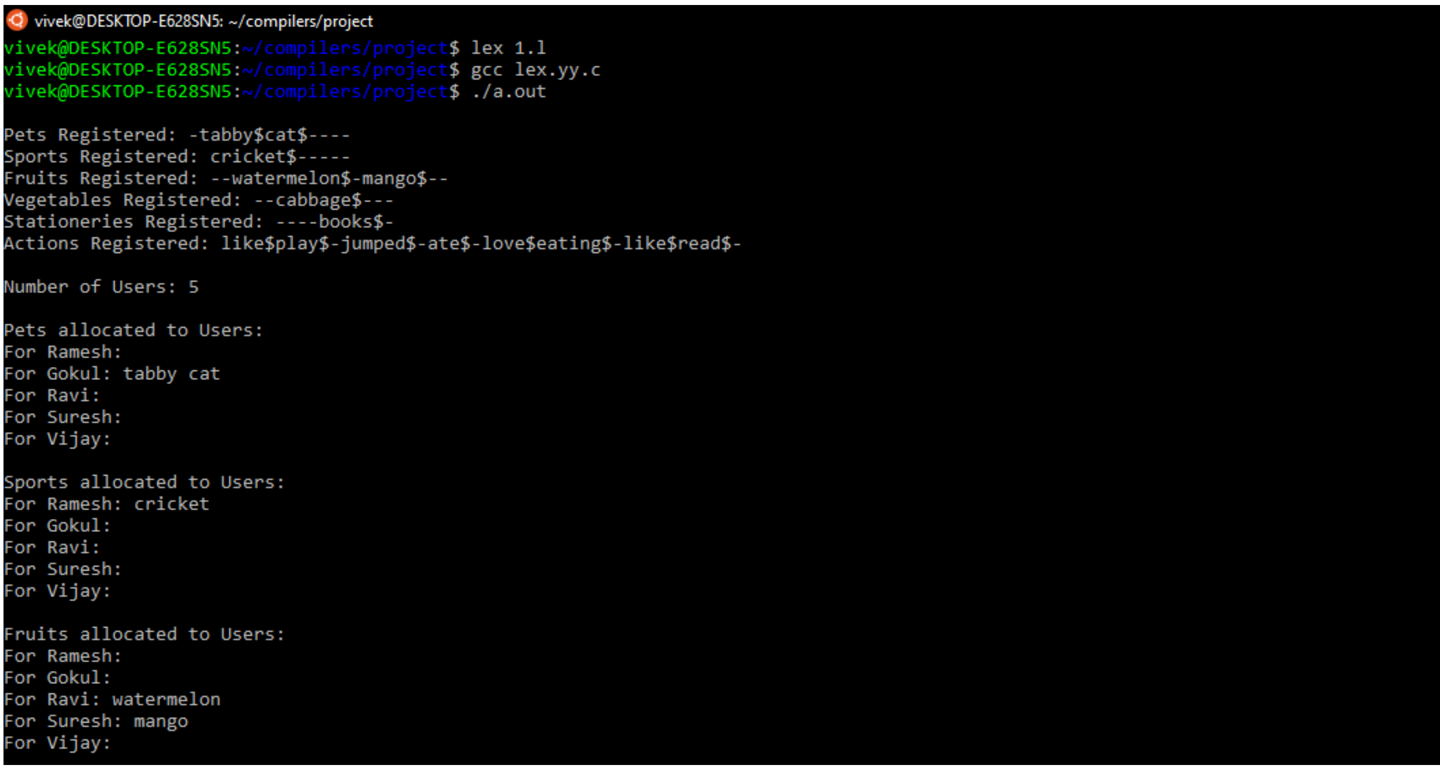
Another simple error check is the detection of the user anywhere in the sentence. This is easily done by understanding words that are not present in the existing databases as users. However, it involves adding a huge amount of words to the existing databases.

Input:



```
vivek@DESKTOP-E628SN5: ~/compilers/project
Ramesh: I like to play cricket.
Gokul: He jumped over the tabby cat.
Ravi: I ate a cabbage and a watermelon.
Suresh: I love eating mango.
Vijay: I like to read books.
~
~
~
~
~
```

Output:



```
vivek@DESKTOP-E628SN5: ~/compilers/project
vivek@DESKTOP-E628SN5:~/compilers/project$ lex 1.l
vivek@DESKTOP-E628SN5:~/compilers/project$ gcc lex.yy.c
vivek@DESKTOP-E628SN5:~/compilers/project$ ./a.out

Pets Registered: -tabby$cat$----
Sports Registered: cricket$-----
Fruits Registered: --watermelon$-mango$--
Vegetables Registered: --cabbage$---
Stationeries Registered: ----books$-
Actions Registered: like$play$-jumped$-ate$-love$eating$-like$read$-

Number of Users: 5

Pets allocated to Users:
For Ramesh:
For Gokul: tabby cat
For Ravi:
For Suresh:
For Vijay:

Sports allocated to Users:
For Ramesh: cricket
For Gokul:
For Ravi:
For Suresh:
For Vijay:

Fruits allocated to Users:
For Ramesh:
For Gokul:
For Ravi: watermelon
For Suresh: mango
For Vijay:
```

So far, two modules have been implemented, i.e identifying and classifying the interests of users, and allocating them to each user.

*This will be updated once more modules are implemented.

Conclusion:

The information obtained from direct interactions can be a valuable tool, as we can create recommendation systems on the basis of user-user interactions, such as comments on a forum, rather than user-site interactions.

Previous systems understand and provide recommendations based on similarities between users, which might lead to a bias evaluation for user preferences. This provides a solution for this issue by integrating direct interactions into recommendation systems.

However, there is still much left to improve, and a full accuracy of such systems is never guaranteed, but this approach will definitely up the bar in extending further recommendation systems.

References:

1. C.Li, F.Xiong, “Social Recommendations with Multiple Influence from Direct User Interactions”, Aug. 2017.

2. C. Pan, W. Li
   Research paper recommendation with topic analysis
   In Computer Design and Applications IEEE, 4 (2010)
   pp. V4-264

3. J.A. Konstan, J. Riedl
   Recommender systems: from algorithms to user experience
   User Model User-Adapt Interact, 22 (2012), pp. 101-123