YARB: a Movie Recommendation Bot using User Modeling Component

Yangwook Kang Jerroyd Moore Justin Chen Harrison Vuong
University of California, Santa Cruz

Abstract

We present YARB, yet another movie recommendation bot, that provides a personalized movie recommendation based on user preferences on actors, actresses, directors, genres, and movies. YARB computes a score for each movie by calculating a weighted sum of movie ratings, popularities, and user preferences, and generate a list of recommendations. In addition, in order to improve the accuracy and the quality of the results, we use WordNet to match verbs and adjectives to synonyms to determine classifiers and various degrees of ratings in NLU. The dialogue manager and user modeling component are designed to support multiple output types such as paging and one best result, and manage users' and adjust the weights reacting to users' feedback.

1. Introduction

There are several movie recommendation engines available online. Popular movie sites, such asNetflix and Movielens, ask users to rate movies they watched before recommending movies [3, 5]. IMDB provides a section for similar movies based on the user search criteria [1]. Nanocrowd, a more advanced search engine, provides recommendations based on users' favorite movie, actor, or director [4]. While many of these systems leverage crowd-sourcing, a technique that matches groups of users with similar taste, to recommend movies, this is not feasable for systems with a small user base. In addition, these systems do not support natural language inputs.

In this paper, we present YARB, Yet Another Movie Recommendation Bot, that provides a personalized movie recommendation based on user preferences. YARB performs much of the other movie recommendation project's functionality, but with an added benefit of adjusting the recommendation rule, by reacting to the user's end-of-session survey to provide superior results. The way it shows the recommendation can also be chosen among the four types the our system supports: Ordered top results, best one, shuffled top results, and a random top result.

We focused on improving the Natural Language Understanding (NLU) and the user modeling component in order to improve accuracy and the quality of the results. Because the English language allows a user to ask for a movie recommendation and express his or her preference in an uncountable and infinite number of ways, WordNet was choosen to increase YARB's understanding of user utterances, specifically for verb and adjectives. The user modeling component stores and manages user preferences, computes the weights, and generates a movie recommendation. When compared to the baseline NLU, our improved NLU increased accuracy by 40% in our experiment. In addition, the system also presents the recommendations better than the old system, improve the User Experience and User Satisfaction, although this is not evaluated in this paper due to the time contraints.

The rest of the paper is organized as follows. Section 2 discusses the design issues of each subsystem. Section 3 explains the implementation of our system. In Section 4 shows our preliminary results. Finally, we draw conclusions in Section 5.

2. Design

The dialogue systems consist of three subsystems: Natural Language Understanding (NLU), Dialogue Manager (DM), and Natural Language Generation (NLG). NLU disassembles and parses an user's utterance to derive meaing and user intent. This is delivered to the DM, which determines and executes what YARB's action will be and how to respond to the user. This information is then sent to the NLG, which constructs a user response based on pertinant information the DM has passed to it. This is the basic control flow of the YARB agent. In this paper, we improved NLU by using WordNet, and extends DM to maintain a user model, which contains explicit assumptions on all aspects of the user that is relavant to the system [7].

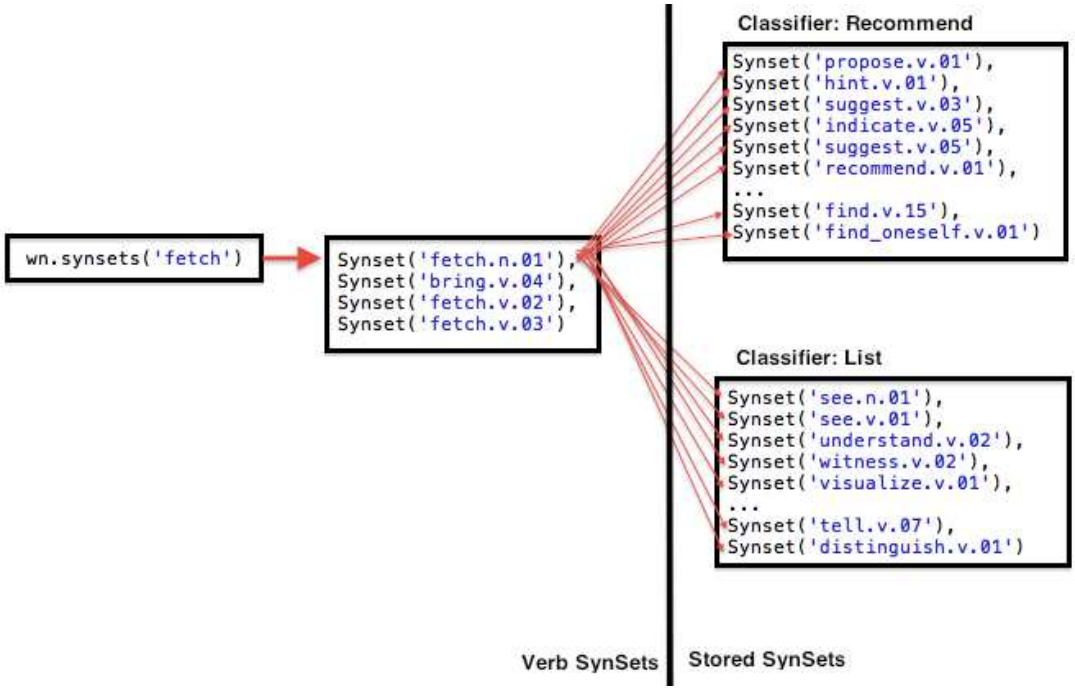

Figure 1. Verb Synonyms and their classification: for command verbs, their synset is computed. Each element in this list of verb synsets will be compared against all synsets stored, and each comparison will generate a Confidence Level. The pair with the highest confidence level will win, and the correct classifier will be assigned.

2.1. Natural Language Understanding

The goal of the NLU is derive meaning from user utterances. This is achieved through parcing and acutely identifying entities useful to the DM. Because the movie recommendation domain is narrow, The NLU utilizes trained sentences to assist in chunking, to build an ontology of genres and emphasize certain keywords, such as the words ”director,” ”cast,” ”movie,” etc. Since the number of synomyms for ”recommend” is vast, and the number of ways expressing user preferences is incalculable, the NLU leverages WordNet’s synonyms to match large sets of verbs and adjectives to meaningful classifiers and variable ratings. This improves the NLU’s performance and increases code readability, because we avoid writing large quantities of rules or if-then statements for the various types of verbs and adjectives. WordNet allows us to easily classify verbs and adjectives.

First, the NLU receives and chunks the user utterance to construct the parse tree. The parse tree’s first branch is examined to determine which parsing function to send the tree to. Each parsing function is designed and optimized to handle a particular configuration of the parsing tree. The trained sentences mentioned earlier assists in determining what type of output the user expects (i.e. does the user want a list of movies or a list of cast members?), and the trained sentences assist in identifying entities.

There are three node types of the parse tree that the NLU examines: questions, commands (also covers existential utterances), and requests or REQ. Questions are directed to the trivial parsing module, whose functionality remains mostly unmodified from the base code. Commands and Existential statements are directed to the command parser function, where the classifier to dynamically determined. REQ nodes are treated as requests. Typically, they may be a child of the questions or command nodes (depth: 2). In this case, semantically, the user is requesting the agent to answer a question, or is requesting the user to perform a command. When the REQ node at depth one, the utterance is determined to be of ’user preference.’.

When the NLU needs to dynamically determine a classifier from a verb, the verbs synsets are computed and compared against our stored synset mapping, classifierSS_list, as shown in Figure 1. We examine the cross product of the verbs synsets and our synset mapping. For each round, the hyponym distance is the primary metric to calculate similarity, also known as Confidence Level. Another factor is if the pair of synsets (one from the verbs collection of synsets and one from the stored collection of synset to classifier

mapping) shares the same tag of speech, the Confidence Level for this round will receive a bonus weight of 1.5. Finally, the synset with the highest confidence level will be chosen, and the classifier that mapped to that synset will be chosen.

The two classifiers that use this technique are ”recommend” and ”list.” This may be expanded to look for other classifiers by adding entries to the verbToClassifierMapping. During NLU initialization, the verbToClassifierMapping is expanded into classifier to a synset mapping, classifierSS_list. The verbToClassifierMapping dictionary includes verbs we predict the user will use to ask for a recommendation or a list.

To determine the rating entity of the user_preference classifier, this same process using a different set of stored system synsets is used. An adjective or user-preference verb will map to a rating from 1 (lowest) to 5 (highest).

The NLU also includes a catch-all entity-pair parsing function. This is called on every utterance, and will find search for entity pairs, such as movieTitles, directors, actors, and genres. It’s important to note that the entity ”type” is not found with this method, instead that is determined by the parsing function that set the classifier. Not all classifiers require the entity ”type.”

2.2. Dialogue Manager

Dialogue Manager in YARB is a simple state machine. It internally maintains three states: normal, paging, and feedback. Each states expects different kinds of utterances. For example, ’next’ and ’prev’ entities can be interpreted in paging state, but not in other states.

In the normal state, the system expects to get entities related to user preferences, recommandation requests, trivia, or a simple search requests. The DM will send a request for data to corresponding modules such as user modeling component or database module, according to the classifer and the entities. If there are more than 10 results, the normal state changes to the paging state.

The paging state shows a paged list. If the NLU passes the ”next” or ”prev” classifier, the DM will remain in the paging state. If it receives a different classifer, the DM will regress back to the Normal State and process the classifier as such. The ”next” and ”prev” classifiers allows users to navigate through the list, 10 results as a time.

The DM will enter the third state, the Feedback State, towards the end of the user session. The agent will prompt the user to find out if the user was satisfied with the recommendation results, or they liked the output type.

Four different types of outputs are supported in the Dialogue Manager: Ordered top results, best one result, shuffled top results, and a random top result. Ordered top results mode shows the movies that have the highest score first, and best one result returns a movie that has the highest score. In shuffled top results mode, the system first selects top 200 movies based on their score, shuffles the list and return it. Lastly, a random top result picks one movie randomly among the top 2500 movies.

2.3. User Modeling Component

A user modeling component incrementally constructs a user model to supply other components of the system with assumptions about the user [7]. Our user modeling component is designed to stores user preferences on movies, actors, genres, directors, output type, and recommendation score, all of which are relevant to and affect the movie recommendation given to the user.

Movie Recommendation. A naive approach to recommending a movie using user preferences to to find the intersection between highly rated movies and movies starring the user’s favorite actors and directors. This approach will yield a list of movies the user is already familiar with. In addition, taking the intersection of these is not enough to provide a recommendation. Taking the intersection is more akin to trivia: finding movies that the actors stars in. YARB uses a heuristic algorithm to recommend movies. YARB provides a personalized and reasonable movie recommendation based on three dimensions: the user modeling component, IMDB movie rating, and IMDB popularity. [1].

The first dimension discussed is the user modeling component. Since NLU identifies the degree rating of adjectives, DM is able to generate a point scores ranging from -0.2 to 0.2 according to this rates. Favorite things of the user will be assigned a weight of 0.2. Things the user likes will receive 0.1. A score of -0.1 is assigned to things the user dislikes. Finally -0.2 is reserved for things the user hates. YARB computes sum of points for each user preference, which is multiplied by the user preference weight in the final score. This is the user modeling component dimension.

The second dimension is movie rating. By examining movie ratings data from IMDB, We’ve discovered that many obscure movies have IMDB’s highest rating, but are not recognizable. Most of these movies are short films that would dominate our top 10 results. Since our target audience does not desire to watch short films, our recommender algorithm needs to exclude these movies. However, because the running time of

films are stored in plain text along with other metadata information, such as nationality, explicitly excluding these short films will affect database performance. As an alternative, we introduce a weight for movie ratings, and set it to 0.3–0.5 by default. By doing this, we calculate short films with a lower score so they will appear lower on the recommendation list. This is the movie rating dimension.

The third and final dimension is popularity. Popularity is calculated by converting the number of votes for a movies to a score based on its voting distribution. We chose this method, because unlike the ratings, we found that the movies that have a higher number of votes tend to be more popular than the movies that have a smaller number of votes. Due to this observation, our recommendation algorithm includes this popularity dimension.

YARB's final score for each movie is generated by computing the weighted sum, defined as follows:

$$WEIGHTED\_SUM = \sum_{i=1}^{n} WiPi$$

where Wi is the weight of each type of information and Pi is the point of each type of information we described above (rating, popularity, user preferences). This calculation is performed on the database server for performance. Attempting to pull thousands of movie records to perform this algorithm on the client-side is proved to be disastrous.

Finally, the weights are adjusted based on the user's feedback. If the user dislikes the movie recommendation for the current session, YARB reduces the weight that has the highest priority, and increase the weight that has the lowest priority for the next session. The amount of reduction and increment is chosen randomly in a given range.

2.4. Natural Language Generator

The Natural Language Generator is a module in YARB that will generate understandable statements for a user. The dialog manager and the nlg contains its own set of classifiers that will be used to pass outputs from the DM through to the NLG. Furthermore, the mechanism that enables YARB to generate readable outputs is by first creating templates with variables which will later be replaced by data returned from the DM. As mentioned earlier, this data is passed from the DM to the NLG with another set of classifiers. While the NLG will handle most cases and scenarios, in instances where the variables are not handled by the NLG, the NLG will ask the user for further clarification.

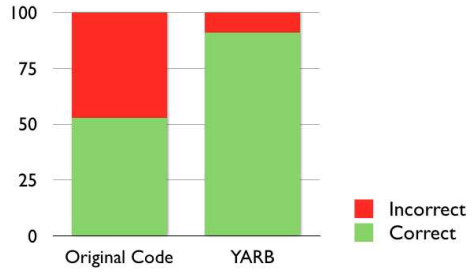

Figure 2. NLU classifier and Entity Recognition evaluation. The left-hand side is the base code's performance. The right-hand side is the improved NLU's performance.

3. Implementation

We have implemented YARB using Natural Language Toolkit [6] in python 2.6. Based on the skeleton code, we first start with refactoring the code to improve a readability, but eventually we ended up rewriting whole parts except for parts of the NLU.

IMDBpy module [2] is used to retrieve some information from the IMDB web site, such as the top 250 and the bottom 100 movies, This module does not connect to the IMDB database on the School of Engineering's MySQL server because of the slow performance. For other types of database requests such as trivia requests and recommendations, we directly connect to the MySQL server, owned by the School of Engineering, to execute the curated SQL commands.

We utilize two tables to calculate a score: yarb_score and yarb_genres. These tables were specifically created for YARB and they improve the query performance. The table yarb_genres stores the names of the genres in IMDB, and their ids. This table allows YARB to find movies with the same genre quickly. This table specifically addresses significant performance issues the database has when filtering and selecting over movie genres. The table yarb_score is used to store the score for each movie. It acts as an intermediary step for the recommendation algorithm. This table's attributes contain the IMDB rating, its popularity score, and user preference score for each movie. These SQL schemas expect to only one YARB instance to be running at any given time.

4. Experiments

We have measured the accuracy of NLU to see the effects of synonym detection and our improved grammer. Figure 2 shows the difference in accuracy between the skeleton code and our improved NLU. When compared to the baseline NLU, the current version improves the accuracy by 40%.

5. Future Work and Conclusion

We have presented YARB that provides a personalized movie recommendation based on 3 dimensions: IMDB rating, popularity and user preferences. YARB's NLU uses WordNet to catch verb synonyms to determine classifiers, and the degree of ratings based on adjectives. It also adjusts the recommendation rules and the output type based on users' feedback. We have added new tables to optimize the query performance, compute a score for each movie and generate an ordered list of movies.

Currently, the algorithm that adjusts weights are rather straightforward. We think various learning algorithms can be used to more accurately adjusts the weights. Future work for the NLU includes pronoun resolution and support for conjunctive clauses. Pronoun resolution should be straightforward. Simply keep track of the last movie and person the agent or the user mentions and substitute it for pronoun words. Support for conjunctive clauses should be relatively straightforward in theory. If utterances are broken by punctuations, such as periods or commas, or words with the tag of speech: CC, this could be accomplished. Conjunctive noun phrases would complicate this process. We've decided not to impliment this due to this complicate, and the additional testing that would be required.

References

[1] IMDB. Imdb. http://www.imdb.com/.

[2] IMDBpy. Imdbpy. imdbpy.sourceforge.net/.

[3] Movielens. Movielens. http://movielens.umn.edu/login.

[4] Nanocrowd. Nanocrowd. http://www.nanocrowd.com/.

[5] Netflix. Netflix. http://www.netflix.com/.

[6] nltk. nltk. http://www.nltk.org/.

[7] W. Wahlster and A. Kobsa. User models in dialog systems. In *User Models in Dialog Systems*, pages 4–34. Springer-Verlag, 1989.