Vote Goat: Conversational Movie Recommendation

Jeffrey Dalton
University of Glasgow
jeff.dalton@glasgow.ac.uk

Victor Ajayi
University of Glasgow
dsvictor.ajayi@gmail.com

Richard Main
University of Glasgow
rmain.gla@gmail.com

ABSTRACT

Conversational search and recommendation systems that use natural language interfaces are an increasingly important area raising a number of research and interface design questions. Despite the increasing popularity of digital personal assistants, the number of conversational recommendation systems is limited and their functionality basic. In this demonstration we introduce Vote Goat, a conversational recommendation agent built using Google's DialogFlow framework. The demonstration provides an interactive movie recommendation system using a speech-based natural language interface. The main intents span search and recommendation tasks including: rating movies, receiving recommendations, retrieval over movie metadata, and viewing crowdsourced statistics. Vote Goat uses gamification to incentivize movie voting interactions with the 'Greatest Of All Time' (GOAT) movies derived from user ratings. The demo includes important functionality for research applications with logging of interactions for building test collections as well as A/B testing to allow researchers to experiment with system parameters.

CCS CONCEPTS

• **Information systems** → **Users and interactive retrieval**; **Recommender systems**; • **Computing methodologies** → *Discourse, dialogue and pragmatics*;

KEYWORDS

Conversational search, Multimedia, Recommender systems, Collaborative filtering

ACM Reference Format:
Jeffrey Dalton, Victor Ajayi, and Richard Main. 2018. Vote Goat: Conversational Movie Recommendation. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.1145/3209978.3210168

1 INTRODUCTION

Recent advancements in speech recognition technology and natural language understanding are leading to renewed attention on digital personal assistants, such as Apple's Siri, Microsoft Cortana, and the Google Assistant. Each of these assistant platforms provides its own framework to create agents. One common personal assistant task is the recommendation of relevant content based on user preferences.

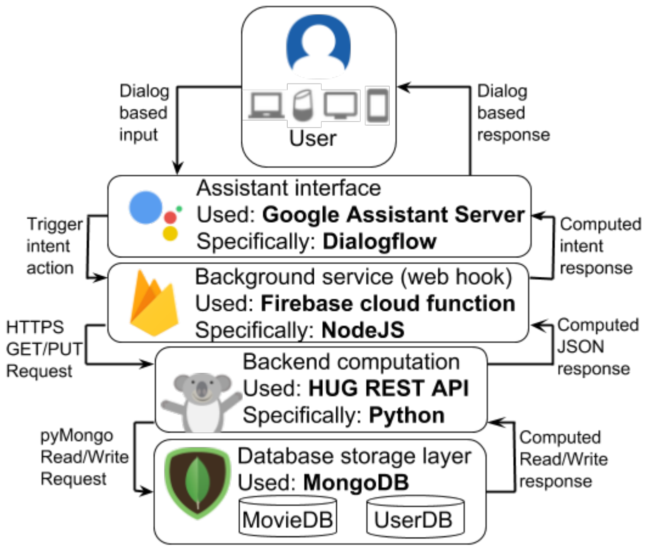


Figure 1: Architectural overview

For this demo, we focus on the creation of a movie recommendation agent, Vote Goat, which is an open-source agent built using Google's DialogFlow platform and released on the Google Assistant.

Vote Goat is a prototype agent system that demonstrates a number of functions important for research. It supports the logging of user interactions through Dialogflow and Google analytics, including recommendation session history and A/B test data. These features are implemented for studying the impact of recommender system algorithm quality as well as gamification.

The Vote Goat demonstration system encompasses multiple dimensions and challenges of creating a robust and effective recommendation agent system:

• Designing robust grammars for intents using Dialogflow's web interface, accounting for alternative language formulations.
• Sourcing reliable and large-scale movie metadata.
• System design of a low-latency open source software architecture capable of meeting Google's response constraints.
• Collecting a crowdsourced movie rating dataset through the movie ranking intent.
• Gamification of repeated movie ranking through a leaderboard and input on Great Of All Time (GOAT) lists.
• Flexible APIs for experimenting with collaborative filtering algorithms for movie recommendation.

The goal is to accelerate research and development of new conversational recommendation agents by reusing components. Vote Goat is one of the few open and freely available exemplars of more complex agents.

2 SYSTEM OVERVIEW

2.1 Architecture overview

The end user interacts with Vote Goat through the Google Assistant platform using a compatible device/application and then invoking Vote Goat explicitly by requesting to 'Talk to Vote Goat' or implicitly by requesting a Vote Goat sub-function such as "What are the best movies of all time?" Once in the Vote Goat agent, the user's natural language input is interpreted by Google's Dialogflow API and matched to an appropriate intent which transparently triggers the intent's Firebase cloud function action (webhook) with any user input in order to perform complex fulfillment (computations/operations) and returns data formatted for consumption via the Google Assistant interface.

We now provide an overview of the agent intents and backend system functionality. A HUG REST[1] miroservice API bridges between the NodeJS cloud functions running on Firebase and backend systems (in Python) running in Google Cloud virtual machines. For data storage, MongoDB is used to store movie and user data due to its flexibility and use of JSON as a native format. To fulfill an intent the backend system retrieves movie data from MongoDB to build a metadata field response for Dialogflow to render to the end user. Beyond this, additional service proxy and process management are performed using NGINX and Gunicorn.

2.2 Recommendation System

The fundamental component of a conversational recommendation engine is the recommendation algorithm. There are a wide variety of algorithms for movie recommendation. Instead of a single model or framework, Vote Goat supports multiple recommendation algorithms and libraries: Tensorflow-based neural collaborative filtering models served via Google CloudML [5], PyTorch models via the Python backend interfaces, as well as deterministic models. The demonstration system also supports basic random or popular movie baselines. For demonstration purposes, the models are trained on publicly available MovieLens (20M) data [4] as well as user generated Vote Goat movie ratings from interactions with the system. We note that this architecture could be extended to further support additional recommendation and retrieval algorithms using the flexible micro-architecture design.

2.3 Open Source Stack

The technology stack for Vote Goat is intended to be simple to use and extend (using NodeJS and Python), highly scalable (using Gunicorn and HUG), modular, and easily replicated. All source code components within Vote Goat are fully open-source and MIT licensed on GitHub [2].

2.4 UI Device functionality

Vote Goat is designed to be multi-platform with support for both visual and speech interfaces. It seamlessly handles differences in device capability throughout every intent. For devices with a touch screen it displays rich responses containing cards, suggestion chips, outbound links, and result carousels. For hands-free (voice only) devices there is descriptive (less verbose) speech rendering. The optimal way to address differences across platforms remains one of the usability challenges, particularly for hands-free devices with limited information bandwidth.

Vote Goat is compatible with all Google Home devices (Home, Mini, Max) and the Google Assistant mobile application for iOS and Android. Vote Goat is published on the Google Assistant directory and is available to all users without any additional installation required.

2.5 Logging support

Each layer of the Vote Goat architecture has logging support. The Actions on Google console provides high-level analytics when the agent is published in the Assistant directory. The Dialogflow framework provides high-level intent analytics. Firebase provides cloud function debug logging. Chatbase provides low-level conversation analytics throughout the cloud functions and Google Analytics is used to instrument the HUG REST API usage. Using these logging mechanisms allows developers and researchers to both optimize the system and develop user models of conversational interaction.

For demonstration purposes, we use the Google provided temporary identifier that is not linked to any permanent user profile. The anonymous user ID is used to store all user-specific data including: the user movie rankings, the movies recommended, and interactions with the movie recommendations for explicit and implicit feedback. The anonymous IDs are transient and expire after thirty days of inactivity.

2.6 Gamification

In order to maximize user movie ranking participation the demonstration system implements the following forms of gamification:

- Personal leaderboard ranking statistics that enable users to compare their participation compared to the entire user population.
- 'Greatest Of All Time' (GOAT) lists of the most popular movies based on the user ratings (overall or within user defined genres).
- Leaderboard progress notifications as users rank movies the system notifies the user of their leaderboard status and progression.

The demonstration system provides a platform to experiment with these incentives as well as future mechanisms.

2.7 A/B Testing

The system includes support for A/B testing to vary the recommendation algorithm as well as other system parameters. The testing is implemented using a test id session field that is stored in MongoDB. This is flexible depending on the scope of experiment required. It allows switching between recommendation models depending on the experiment field. The model selected is logged alongside the movie recommendation session history enabling later analysis of test data.

The implemented A/B testing participation solution is quite aggressive since all users are subjected to A/B testing in production. In future work, the demo may be extended for more targeted and selective test participation. This is important to balance user experience with the need to collect experimental data. Also, one current limitation is that experiments require manually extracting the A/B test data from MongoDB for analysis.

[1]http://www.hug.rest/
[2]https://github.com/know-ail

3 MOVIE AND RATING DATA

Vote Goat uses movie description and metadata from 'The Open Movie Database' (OMDB). The data is publicly available, supported by Patreon, and has non-restrictive licensing (CC BY-NC 4.0) for non-commercial research purposes. Vote Goat has wide coverage with approximately 160,000 movies crawled from OMDB in December 2017. From this data, approximately 30,000 movies are removed because they lack IMDB ratings. Also, adult or pornographic movies are removed in accordance with Google's content restriction policies. The final movie count available is approximately 114,000. The data used is available on the Vote Goat online appendix, as well the code to crawl new data and update the database.

For the recommendation algorithm, we use the MovieLens 20M ratings research dataset [4]. We translate between the MovieLens movie identifiers and the OMDB item identifiers. Additional (anonymous) rating data collected by Vote Goat is also used for training and we plan to release it on the website in the future.

4 DIALOGFLOW IMPLEMENTATION

We use Dialogflow to implement Vote Goat's voice-based interface for the Google Assistant. This includes the creation of 11 intents that trigger cloud functions depending on the user's response to the prompt. To infer each intent DialogFlow uses machine learning to learn intent grammars from a small number of examples. Also, every intent that prompts the user for information has a unique fallback that provides the error handling.

A fully detailed list of all the intents and how they are triggered is beyond the scope of this paper, but is documented with the source code online. An example of the primary intent is shown in Figure 2 and other intents in Figure 3.

4.1 Dialog context management

Managing ongoing dialogue state context is an important aspect of conversational agent design. Whenever a user interacts with an intent, Vote Goat stores temporary intent flags in the Dialogflow context data storage, enabling error-handling fallback intents to provide relevant error text and the passing of movie metadata in JSON between intents. These are used for intent handling as well as constructing rich responses to display to the user.

The lifetime of contexts controls when they expire between intent interactions, if set too high it can cause unexpected issues such as showing the wrong fallback text if a context lives longer than it should (zombie context) or an unexpected error warning if the context expires prematurely despite being required for the next intent. The latest DialogFlow API also allocates a small quantity of local storage to store data across sessions.

5 LESSONS AND CHALLENGES

In this section we detail some the challenges and experiences developing the system. We highlight areas that require future development both in the DialogFlow framework as well as Vote Goat.

Discoverability of the agent and agent's capabilities is an important feature for virtual assistants. It is also a new adversarial domain for the agent vs. platform. The platforms are federated agent search systems that route conversations to agents both on explicit as well as implicit intents. The explicit invocation requires

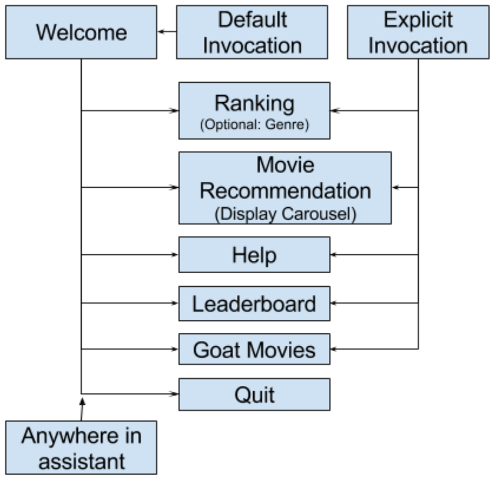


Figure 2: Dialogflow primary intent overview.

relatively unambiguous names of agents, which is quite limiting. Additionally, many platforms support 'implicit' invocation of an agent based on the agent's declared capabilities. As currently defined in DialogFlow, each agent intent has an implicit invocation entry point that is an opportunity to attract user traffic towards the agent. However this is not appropriate for all intents as users who experience an irrelevant agent invocation for an utterance are likely to abandon the agent or inflate fallback statistics. Similar to previous work on webspam and search engine optimization, methods for effective detection and filtering of abuse and relevance of an agent for an implicit invocation need to be developed. In the current system, this can lead to inadvertent implicit invocations of the agent that may unintentionally trigger for irrelevant intents.

One current limitation of DialogFlow is support for rich third-party knowledge bases of entities with thousands or more entities. Initially, search functionality was added to the system to allow users to search for movies with specific names of actors, genres, and every other feature of the movie metadata, however, this proved problematic. Although a small file of entities can be added (via JSON files), for example movie genres, the rest of the data was not easily imported. DialogFlow imposes limits on the number of entries uploaded (both rows and attributes), which precluded the millions of actors (with tens or hundreds of attributes in the form of movie roles). This is a barrier to creating large-scale agents. As a result, this functionality must be managed by the webhook backend. This results in overly complex agent systems with parsing and NLP happening in multiple locations. Similarly, complex search that involve multiple variables (actor + genre) simultaneously was not easily supported.

Another limitation of the current platform is support for rich visual interface elements. One of the most requested features is the ability to share recommendations socially. However, it is currently impossible to insert social icons in the visual interface. Similarly, displaying rich cards was challenging and all complex elements are generated in the backend system. This again shifts processing to the agent backend unnecessarily.

Another current challenge is fallback handling. In case the user does not enter a recognized input valid for the intent, the desired behavior is a meaningful fallback dialog to prompt the user and recover fluidly. However, system testing revealed that current fallback mechanisms do not always work correctly.

6 FUTURE DEVELOPMENT

The current demonstration system has several areas for further development.

Vote Goat needs a backend data pipeline to continually update the agent's data. The movie and rating data needs to be refreshed regularly to stay up-to-date with current movies and ratings. There also need to be automatic jobs to retrain and update recommendation models from the user rating data. This is currently a manual and offline process.

Second, the current system does not implement safeguards for users spamming the agent. The gamification aspect of using leaderboards should be extended to prevent abuse by limiting the quantity of ratings and detecting spam users (possibly other agents!). This could also be addressed by requiring mandatory sign-in to Google.

Third, improved support for multiple platforms needs to be developed. The current agent is optimized for voice input, but visual output with rich card displays. The current hands-free voice output is overly verbose, leading to long interaction times and fewer overall interactions. The movie result lists needs to be more concisely summarized. The other intents also need refinement to reduce the text content and make the instructions clearer and more concise.

Finally, we plan to extend Vote Goat in multiple possible research directions. One possible area is collaborative recommendation, to allow a group of users to use Vote Goat together to decide joint movie recommendations. We also plan to further generalize the framework to support additional domains (videos, music, etc...).

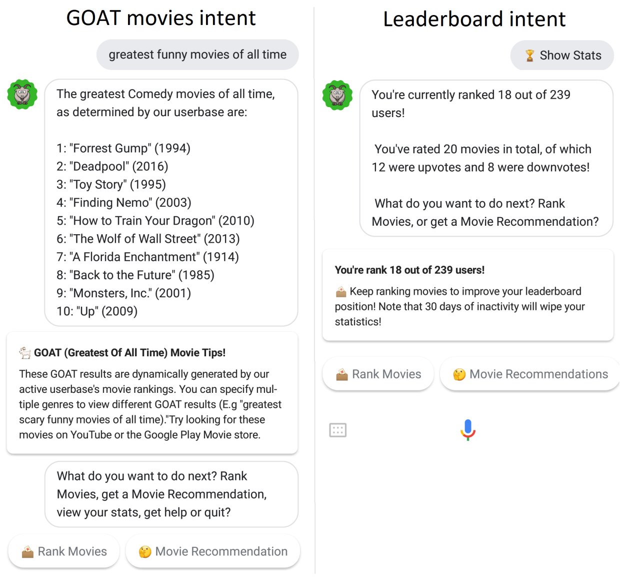
7 DEMONSTRATION

We demonstrate a full end-to-end demo of Vote Goat that is released on the Google Assistant. This will cover all aspects of the system, with a primary focus on ranking and recommending movies on YouTube and the Google Play store. It will also utilize checking the leaderboard and searching for movies based on metadata. We may create a customized SIGIR leaderboard and GOAT lists to incentivize participation at the conference.

8 RELATED WORK

The implementation of task-oriented dialog systems has been done in various fields from sales to customer service automation. These systems are often developed for specific domains and use carefully crafted keyword matching to trigger intents. In conversational recommender systems, previous work on preference elicitation [2] showed a conversational model could be effective at rapidly learning users' preferences. In this work we build on this, developing a task-oriented movie recommendation agent that elicits user's movie preferences to improve recommendation quality.

Movie recommendation systems are well-studied, with initiatives such as the Netflix prize challenge [1]. Beyond movies, recent work used neural collaborative filtering methods to recommend YouTube videos to watch [3]. An early example of research in media



The greatest Comedy movies of all time, as determined by our userbase are:

1: "Forrest Gump" (1994)
2: "Deadpool" (2016)
3: "Toy Story" (1995)
4: "Finding Nemo" (2003)
5: "How to Train Your Dragon" (2010)
6: "The Wolf of Wall Street" (2013)
7: "A Florida Enchantment" (1914)
8: "Back to the Future" (1985)
9: "Monsters, Inc." (2001)
10: "Up" (2009)

GOAT (Greatest Of All Time) Movie Tips!
These GOAT results are dynamically generated by our active userbase's movie rankings. You can specify multiple genres to view different GOAT results (E.g "greatest scary funny movies of all time)."Try looking for these movies on YouTube or the Google Play Movie store.

What do you want to do next? Rank Movies, get a Movie Recommendation, view your stats, get help or quit?

You're currently ranked 18 out of 239 users!

You've rated 20 movies in total, of which 12 were upvotes and 8 were downvotes!

What do you want to do next? Rank Movies, or get a Movie Recommendation?

You're rank 18 out of 239 users!
Keep ranking movies to improve your leaderboard position! Note that 30 days of inactivity will wipe your statistics!

Figure 3: GOAT and Leaderboard intents

recommendation for dialogues is Warnestal et al. [6], who examine human-to-human recommendation dialogue for movie recommendation in spoken dialogue systems.

Directly related to the system are other existing movie recommendation agents. The Andchill agent [3] asks the user for a movie they like and the aspect they like about it. The Google Assistant also provides rudimentary movie recommendation capability. In contrast to both systems, Vote GOAT provides a richer set of possible intents.

9 CONCLUSION

The Vote Goat demo movie agent provides a conversational approach to a task-based movie recommender system. It provides a rich set of interaction intents to learn about movies as well as receive movie recommendations. It uses gamification to incentivize participation. The demonstration system is fully-open source using data and systems that are freely available for research purposes. Vote Goat provides an example of a research agent with support for A/B testing and logging of user interactions. Follow-up work will enhance the voice UX design for hands-free devices. Additionally, work on features for new domains and collaborative recommendation will be explored.

REFERENCES

[1] James Bennett and Stan Lanning. The netflix prize. In *KDD Cup Workshop*, 2007.
[2] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *KDD*, 2016.
[3] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*, 2016.
[4] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *TiiS*, 5:19:1–19:19, 2015.
[5] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, 2017.
[6] Pontus Wärnestål. Modeling a dialogue strategy for personalized movie recommendations. In *Beyond Personalization Workshop*, pages 77–82, 2005.

[3]http://www.andchill.io/