CEUR-WS.org/Vol-2500/paper_21.pdf

Smart Time: a Context-Aware Conversational Agent for Suggesting Free Time Activities (invited paper)

Alexandre Frazão
ESTG
Polytechnic of Leiria
Portugal
alexandre.rosario@ipleiria.pt

Ricardo Maltez
ESTG
Polytechnic of Leiria
Portugal
2180240@my.ipleiria.pt

Rúben Caceiro
ESTG
Polytechnic of Leiria
Portugal
ruben.caceiro@ipleiria.pt

Carlos Grilo
ESTG, CIIC
Polytechnic of Leiria
Portugal
carlos.grilo@ipleiria.pt

José Ribeiro
ESTG, CIIC
Polytechnic of Leiria
Portugal
jose.ribeiro@ipleiria.pt

Abstract

This work describes a conversational agent, integrated into the Google Assistant platform, which is able to recommend personalised leisure activities for a user's free time. *Smart Time* is a context-aware Action on Google capable of extracting free slots from a user's calendar and recommend customised activities – which include book, restaurant or outdoor recommendations – in accordance to his/her specific context computed with basis on data extracted from the smart device's various sensors. Additionally, the conversational agent integrates a constantly evolving recommendation engine, which is able to adapt in accordance to the user's choices.

1 Introduction

Voice assisted and conversational agents are a trending technology and research topic; it is predicted that they will soon be predominant in search actions [Sha18, Reb18] and that, by 2020, 30% of all web navigation will performed without a screen. This mostly means better and faster web browsing and user experience, and the websites' ability to provide this service will influence user engagement.

Simple tasks, such as turning off a lamp or scheduling a reminder, are also evolving from simple manual gestures to voice commands that agents can comprehend and act accordingly. Home automation and medical automatic recording were the first fields for voice command targets, as indoor conditions are more favorable to these systems due to lower noise level and consistent internet connection. In most recent times, with the appearance of smartphones and with the constant increased performance of mobile devices and connection quality, more complex systems are appearing. Nowadays people's devices are deeply and constantly interconnected, with

Copyright 2019 for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

In: S. Hölldobler, A. Malikov (eds.): Proceedings of the YSIP-3 Workshop, Stavropol and Arkhyz, Russian Federation, 17-09-2019–20-09-2019, published at http://ceur-ws.org

several other personal and public devices. These interconnections, coupled with powerful information systems, allow developers to extract user context with exceptional accuracy.

User context is any information that a system or a person can gather in order to characterize the situation, place or the environment which the person is into. Context-awareness dives into the gathered contexts and correlates them to better predict relevant tasks, warnings and recommendations to the person or device [ADB+99].

Conversational agents [MLB04] are becoming part of our society since they take advantage of natural language technologies to engage users in text-based information-seeking and task-oriented dialogues for a broad range of applications (e.g. chatbots) [IL17]. By using Natural Language Processing technologies, available through services like Dialogflow[1] or LUIS[2], it is possible to determine the literal meaning of the utterance (e.g., "I'm hungry", "How is the weather today?"). Besides extracting user intentions, Conversational Agents keep context from previous user interactions, making them ideal for complex user requests. These agents are already beginning to play an important role in enterprise software, e.g. by allowing the automation of customer service.

This paper describes *Smart Time*, an Action on Google that employs user context to recommend ways of spending his/her time. It extends Google Assistant functionalities by giving suggestions about what the user can do in his/her free time, recommending books and places. To achieve this, it uses information from Google Calendar to retrieve user events and compute free time for recommending better activities, thus making the day of the user more interesting. It also collects context information from the weather and the places close to the user location, with the intention of presenting more appealing suggestions. Dialogflow is used in order to understand the user request.

Dialogflow consists in a natural language processing service, owned by Google, commonly used in voice apps and conversation agents, which uses machine learning to understand user utterance and extract the user intention. Google Assistant[3] was introduced in 2016

and is currently available on various Android and iOS devices such as smart tv's, smart displays, headphones, and also on lower-end budget devices [TO19]. It supports the extension of app functionalities of already existing Android apps via "App Actions"[4], or native Assistant functionalities via "Action on Google"[5]. Dialogflow's capabilities can be used in order to extract user intents and act accordingly. Since its launch, Google Assistant has rapidly evolved with new functionalities, such as the recently announced facial capability and faster performance for speech recognition [Vin19]. Being one of the most advanced conversational agents available to the general public and the fact that it is available free of charge, makes it one of the best targets for developers.

This document is organized as follows: the next section explores and discusses related work; Section 3 presents the proposed solution's vision, requirements, architecture and functionalities; Section 4 focuses on implementation, that is, how *Smart Time* internally works, its technologies, deployment environment and proposed voice queries; Section 5 shows and explains the possible outputs when the user interacts with the system; Section 6 concludes the paper and sets ground for future work.

2 Related Work

There are already several applications that take advantage of user context to provide recommendations or extract additional information about the user's intention. Some use time or emotion as a key variable, but the most predominant attribute tends to be the user geo-location. This section presents some related works on recommendation systems that take advantage of user context.

"Emotion Analyst" [SAD+19] uses emotion detection technologies to extract the user emotion as the main context and, by mapping all emotions to movie genres. It also recommends the user nearby cinemas to watch the movies it thinks are the most interesting. As opposed to *Smart Time*, Emotion Analyst solely focus in one task: to recommend the best movies for the user's current emotion. For this task, it uses IBM Tone Analyzer[6] to extract emotions from text, Twitter integration for tweets analysis and Microsoft Face API for emotion extracting from pictures associated to tweets. It matches the emotion with the movie genre using a rule-based approach, created with the information collected from the AGoodMovieToWatch[7] platform. After identifying the movie genre, Emotion Analyst uses the TheMovieDB API to get movies from that genre. Finally, it collects the user

[1] https://dialogflow.com/
[2] https://www.luis.ai/
[3] https://assistant.google.com/
[4] https://developers.google.com/actions/appactions/overview
[5] https://developers.google.com/actions/
[6] https://cloud.ibm.com/apidocs/tone-analyzer
[7] https://agoodmovietowatch.com/mood/

location to provide the closest cinema locations and the session hours in which the movie is being presented. This application is able to detect up to eight types of emotions and act according to each one of them. It also uses text analysis to perform decisions and context extraction. However, the decision process is somewhat limited since it maps each emotion to movie genres without providing any means for changes in the future. This means that if a user does not agree with the movie genre mapping, it has no way of changing it.

"Travel Recommendation Using Geo-Tagged Photos in Social Media for Tourist" [MCM+15] proposes an architecture for a system capable of identifying user context for location-based services and use this for travel recommendations. It tries to split the responsibility of context extraction from actuation, by categorizing each functionality as tasks, and further, into computational needs. For location processing, the authors suggest the use of a user-location matrix and a user-user similarity matrix. In this case, the geo-tags presented in photos taken from each user and its associated labels are used.

The proposed architecture is very similar to that of the proposed solution, since it starts with the user interaction with the application, moves to data processing using user settings and fetches external data to take into account when should recommendations be processed, so that it presents more interesting results to the user.

Similar to [MCM+15], "Smart space based recommendation service for historical tourism"[VKI+15] focuses in the tourism attraction and experience. In this case, Latgale historical sites are the main target. It focuses on smart recommendations in a smart space approach, paired with event-oriented multi-agents for historical tourism. The project's main goal was to promote regional and cultural tourism in order to impact historical interest, research, tourism and local economy in environments where historical data is abundant. User context extraction was done using Points of Interest (POI) spread across the historical sites, when a user takes an interest in one POI by searching about it. His/her action is mapped into a semantic network that analyzes all related information and recommends similar POI's that the user might like.

The proposed smart space approach takes advantage of ubiquitous computing by defining the following properties: mobile services; mobile information sources, such as personnel equipped with mobile devices, embedded devices and web services; inter-operable information exchange, heterogeneous agents and information exchangers. To implement these properties the authors used the Smart-M3 platform, which aims to provide semantic web information between devices. The presented concept introduces an interesting point of view, by not seeing problems as something with a fixed solution but as a network of possible solutions. Those solutions can then be linked by looking at user history with similar contents, and with those, present new interesting results for future actions. The proposed solution, tries to simulate the same type of user experience, as well as to optimize the algorithm to best match user expectations on every recommendation.

In the article "Book Recommendation System Using Opinion Mining Technique" [SSA13], the authors take the data mining approach to provide users with book recommendations on the discipline of computer science, more specifically, the top 10 ranked books of the discipline. To implement the system, the authors used costumer reviews as inputs and, by extracting various terms in those reviews, they classify each term as neutral, negative or positive. This is then mapped to a -5 (negative), 0 (neutral) and 5 (positive) scale. The proposed system handles a quite rich environment, making distinctions about non-technical and technical books. For example, a book writer for master's degree engineering students might be useless for first year students, since they lack the technical knowledge needed to understand the contents. In order to handle this sub-problem. The same method was applied to the "price" feature. The final list of features contained seven different attributes (occurrence, helpfulness, material, availability, others, irrelevancy, price).

The results were promising, showing that the proposed method can identify the top leading books in each topic. In contrast, the proposed solution focus in multi-activity recommendation, relying in Play Books recommendation for the eBook's functionality and place location of places is based on weights given by the user submitted rating.

3 *Smart Time* – The Platform

The Action presented in this work focuses on the interaction between the user and Google Calendar to take advantage of his/her time by checking if it has events at that time; with basis on this information, *Smart Time* suggests an activity to be performed. The intention is to promote local exploration, social interaction and to help users to take the maximum advantage of their time. This section describes the Action's functionalities, system layout and how each problem was approached and suppressed.

Smart Time's source code is available on:
`https://gitlab.com/MEI-CM-ARR/context-assistant-fulfilment.git`

3.1 Functionalities

Google Assistant already supports some queries, such as calendar events, reminders and user location. It lacks, however, support for more specific tasks, such as free time calculation, recommendations of books and activity recommendations.

Smart Time proposes extending Google Assistant's default functionalities with the following:

- Give recommendations about what to do with the user's free time: suggestions according to the Google Calendar free time by computing time between events or dates;
- Give book recommendations: use Google Books user account for new book recommendations;
- Give place recommendations by type: recommend restaurants, monuments and cities according to the amount of free time;
- Detect current user context to weight about each decision (time, weather, user history): by using external API's for context extraction, it is possible to obtain better accuracy and more interesting results;
- Continuous learning based on user preferences: each time the user interacts with the Action, the internal engine will improve with the feedback provided and keep track of user interests;
- Full integration with google services: by using Google native API's it is possible to get all services without the constant need for authentication.

These functionalities aim to make the solution as engaging as possible for the user, making it an invaluable companion for travels and day-to-day use.

3.2 Architecture

Smart Time is a service-based Action: it integrates its functionality with several external systems in order to perceive the user context, as showed in Figure 1.

The user interacts with the system by talking or sending text to Google Assistant, which processes the input and sends it to Dialogflow which, in turn, checks if the user request matches a registered intent of an Action on Google. When the *Smart Time*'s intents match the user's request, Dialogflow triggers the registered webhooks with the intents as parameter.

Most of the services were chosen due to their native Google integration, such as using the same authentication token across all the services. This allows for a native user experience and the user only has to grant permissions and authorization once during all usage.

Smart Time collects user context and acquires information using the following APIs: *OpenWeather API*[8], to collect the current weather status; *Google Calendar API*[9], to compute the free time available; *Google Places API*[10], to collect interesting places near the user; *Google Distance Matrix API*[11], to obtain the distance between the user and a specific place; *Google Book API*[12], to collect books of interest to the user.

In addition to the functional services, *Smart Time* also depends on the AWS S3 service to store the decision tree model and choice history of each user for later retrieval and analysis. All the web services consumed by the Action use HTTP requests following the REST architecture.

4 Implementation

The main system logic is implemented via cloud functions programmed in TypeScript and hosted in the Firebase Cloud Functions[13]. This allows for high availability, reliability and resilience, removing the preoccupations inherent of infrastructures from developers. The Firebase Cloud Function was configured to run Node.js (version 10).

Users can interact with the Assistant by calling "Hey Google" or "Ok Google", which triggers the basic Assistant app (if Google is activated in the system). To be able to access Smart Time, users need to implicitly

[8] https://openweathermap.org/
[9] https://developers.google.com/calendar/
[10] https://cloud.google.com/maps-platform/places/
[11] https://developers.google.com/maps/documentation/distance-matrix/start
[12] https://developers.google.com/books/
[13] https://firebase.google.com/docs/functions

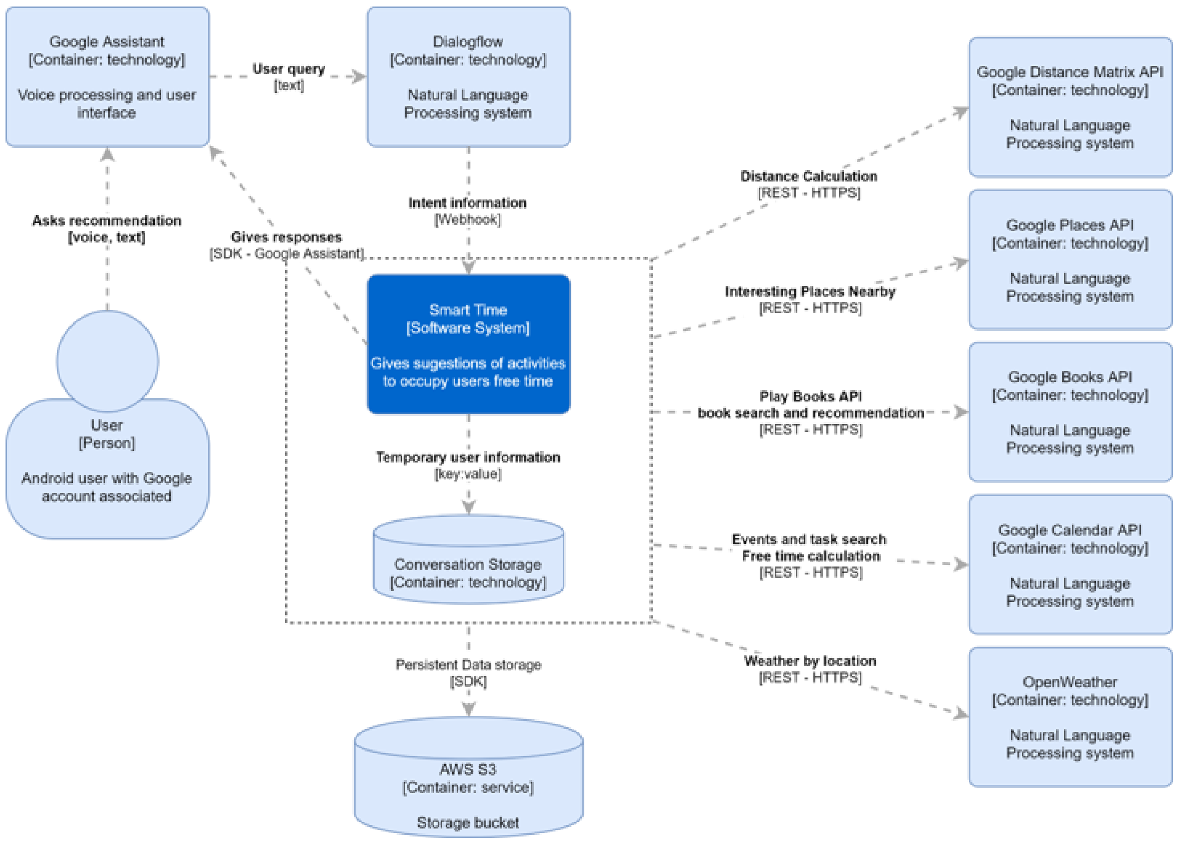


Figure 1: System's architecture

call it by saying "Talk to Smart Time"; this action enables the Action's default "welcome" intent. After this step, the Assistant says "Hello" or other greeting from a pool of responses, and the user is prompted to accept the required permissions (access to calendar, books and location). After these steps, the user is able to see the app icon in the top action bar and some suggestions of possible uses.

4.1 Queries and Responses

Smart Time relies on Dialogflow user intent extraction; Dialogflow extracts context and intentions from voice commands and makes the connection to the Action, which answers back to the user. These commands come in form of "strings" and are treated when some event (voice/text) is triggered via the webhook configured in the Dialogflow panel.

Several types of queries were taken into account when creating the intent training set (cf. Table 1). Simple queries with the same words as the intents, funny quotes that can have meaning (e.g., "fill my belly") and emotion based expressions (e.g., "I'm bored") get translated into "recommend me something". This kind of interactions aim help engaging the user into using the Action.

Table 1 and Table 2 show all registered options that the Action can listen to (supported queries) and respond to (responses). These intents allow for the system to intersect and react to each action individually, making it easier to parse and decide on how to handle each query.

In addition to the fixed intent triggers, it is also possible to invoke the main recommendation action via "Explicit Invocation". This means that the user can invoke an intent via the main Google Assistant conversation directly into the *Smart Time* Action. For example, "Ask *Smart Time* to give me a recommendation" takes the user directly to the *Smart Time* Action and triggers the "ask_for_recommendation" intent. This way, the user does not need to access the Action directly, only after asking for a new recommendation.

Table 1: Most relevant intent calls

Intent	**Voice command**
ask_for_recommendation	“Recommend me something!” or “What do you think I should do?”
books_search	“Get me the book {bookname}” , “I’m looking for the book {bookname}”
books_recommend	“Get me a book to read”, “I want to read a new book!”
books_mybooks	“List my books”, “What are my books?”
free_time	“When am I available?”,“Tell me about my free time for tomorrow!”
get_places	“Get nearby places”, “What locations are near me?”
place_recommendation	“Recommend me a place”, “What are the best places for me?”
place_recommendation_food	“Suggest me a place to eat, “Where can I fill my belly?”
yes	“Yes”, “Yap”, “Sure”, “Go for it”
no	“ No”, “Nope”, “I don’t want to”
user_permission	Google assistant permission event (built-into the system)

Table 2: Intents registered in Dialogflow

Intent	**Voice command**
ask_for_recommendation	Reply with an activity
books_search	Search books by the given name
books_recommend	Recommend a book using Play Books account information
books_mybooks	Lists all available books in the user library
free_time	Gives information about the free time available “today, week”
get_places	Retrieves a list of interesting places nearby
place_recommendation	Recommend interesting monument, restaurant, gym or park
place_recommendation_food	Retrieves a list of places to eat out
yes	Detects affirmative response from user
no	Detects negative response from user
user_permission	Asks the user for permission to access his/her location or other sensible data

4.2 Recommendations and Suggestions

In order to be able to recommend activities and other interesting options, a recommendation algorithm was created. The implementation uses the ID3 decision tree learning algorithm [Qui86], with the assistance of the decision-tree NPM module[14].

The ID3 algorithm was chosen mainly for being lightweight – an important aspect given that it is required, by the Google Assistant service, to provide a response to the user in less than 5 seconds. They also generate understandable prediction rules that can be later analyzed and improved upon.

The recommendation engine takes into account four important context variables: user free time; user location; time of the day (evening, dinner, lunch, morning, night); current weather conditions (rain, sunny).

Depending on the user’s query, free time is calculated from the present moment up to the end of the day, week or a specific date, until an event is found. After that, the time is encoded into specific intervals: less than 30 minutes; between 30 minutes and one hour; between 90 minutes and 3 hours; between 3 hours and 24 hours; between one day and one week.

The time of day variable is also encoded to simplify the tree model: [01:00, 07:00[– dawn; [07:00, 11:30[– morning; [11:30, 14:00[– lunch; [14:00, 19:30[– afternoon; [19:30, 21:00[– dinner; [21:00, 01:00[– night. These variables are then combined in order to make the prediction of the best option for the current situation, as shown in Figure 2.

A simple model represented in Figure 4 was created using a dataset with preferences of different persons in varied situations, in order to provide a default model. This model is used in the first interactions with the Action.

[14]https://www.npmjs.com/package/decision-tree

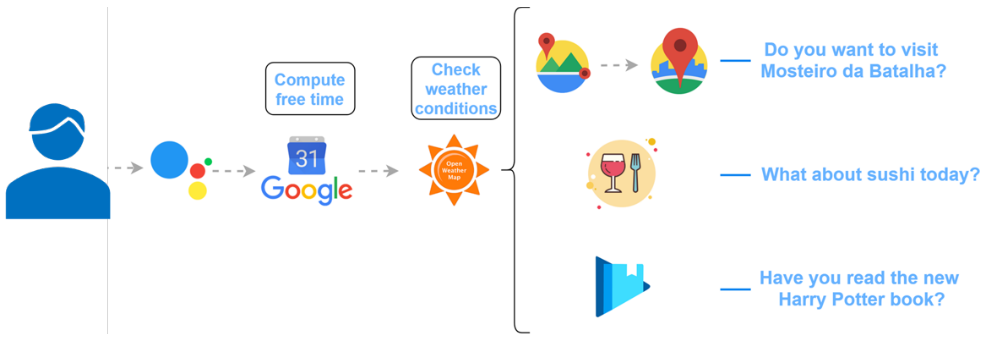
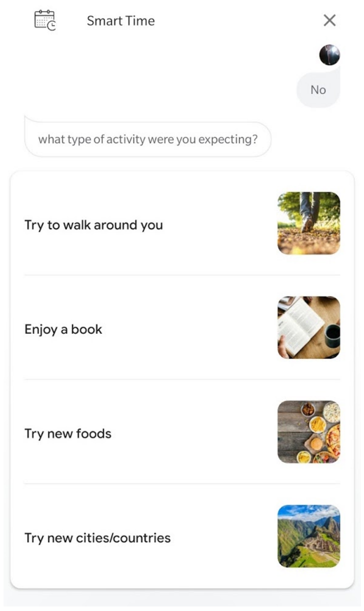


Figure 2: Recommendation computation process

That is, it is the current default model that this Action uses. It has 79.00% of accuracy, measured recurring to cross-validation with 10 folds stratified sampling.

In order to better serve the user, the system can learn his/her preferences by asking for feedback after each prediction. In order to do that, a history of the choices made by the user each time he/she receives a suggestion is saved for future model tuning. The ID3 model is then recreated using this historical data, and updated every time the user disagrees with the suggestion provided (Figure 3).



Smart Time

No

what type of activity were you expecting?

Try to walk around you

Enjoy a book

Try new foods

Try new cities/countries

Figure 3: All activity options without the last suggestion

This approach intends to provide each user with a consistent experience while using the Action – making it his/her own over time.

When the user says that he/she does not agree with the suggested activity, a list of options with other activities is shown. After clicking on one of the new options, the user is redirected to another instance of the Action. In the new instance, a new recommendation of the chosen type is presented. This is possible due to so called deep linking into the Action intent. Usually this kind of functionality is used for websites or services that redirect users from the pages directly to the application on the user smartphone. In this work, the tap functionality of the browsing items was used to redirect the user back to our Action with the new request.

The places recommended by the Action have into account the user's location and the rating that the place has on Google Places. The process starts by calling the Google Places API with the type of the intended place (e.g., a restaurant, a point of interest or a museum). The returned places are sorted from the closest to the farthest one by the API.

Smart Time filters out places that do not have an image, that are closed or that do not have a rating. Finally,

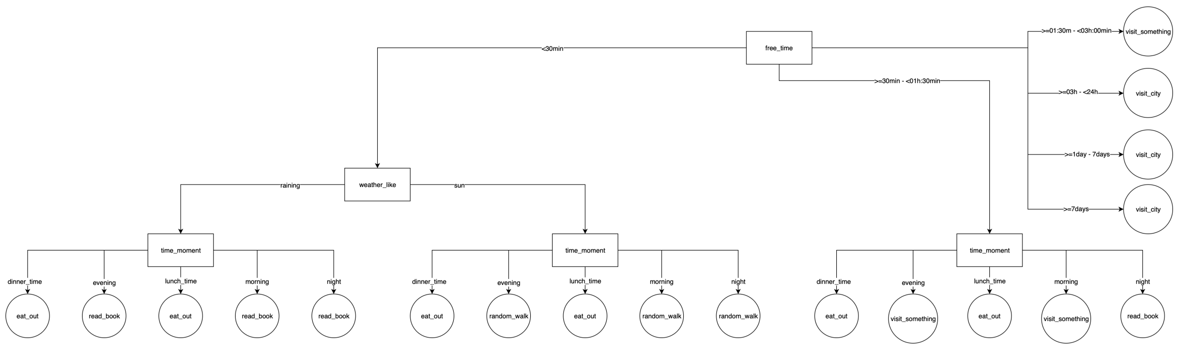


Figure 4: Default predictive model

the places are selected by a roulette wheel like stochastic algorithm where the probability that a place is chosen is proportional to its rating. As output, the algorithm returns one to five places from the complete list, depending on the user's request.

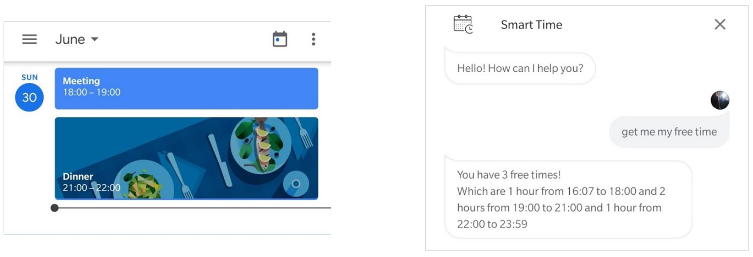
5 Features Demonstration and Discussion

This section presents examples of outputs *Smart Time* exhibits when the user interacts with Google Assistant.

5.1 Free time calculation

Figure 6 depicts an example of the calculation of user's free time, in which two events registered in Google Calendar.

Smart Time calculates the duration of all free slots of the period requested by the user. This period can be the end of the present day, week or month. For the two events, three slots are calculated.



Meeting
18:00 – 19:00

Dinner
21:00 – 22:00

Hello! How can I help you?

get me my free time

You have 3 free times!
Which are 1 hour from 16:07 to 18:00 and 2 hours from 19:00 to 21:00 and 1 hour from 22:00 to 23:59

Figure 5: Free time calculation with two events

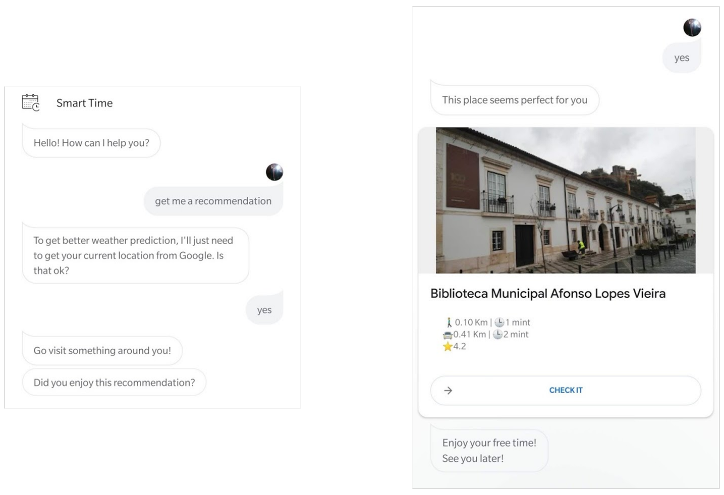
5.2 Activity Recommendation

Since the main goal of *Smart Time* is to be able to recommend activities based on context and user preferences, it is possible to get a recommendation by just saying "Get me a recommendation". *Smart Time* computes the recommendation and shows it (Figure 6).

When asking for the available free time, it is also possible to ask for other time moments, like "get my free time this week"or even "get my free time this month". If the user wants, he/she can also ask for a specific day of the week or for words like "tomorrow"or "next Sunday". The Assistant will translate this to a date so that *Smart Time* can use it.

5.3 Book Recommendation

The user can also request a book recommendation in an explicit way, instead of letting *Smart Time* suggest reading a book. This intent can be triggered by saying "Recommended me a book". The result should be similar to the one depicted in Figure 7-left.


Smart Time

Hello! How can I help you?

get me a recommendation

To get better weather prediction, I'll just need to get your current location from Google. Is that ok?

yes

Go visit something around you!

Did you enjoy this recommendation?

yes

This place seems perfect for you

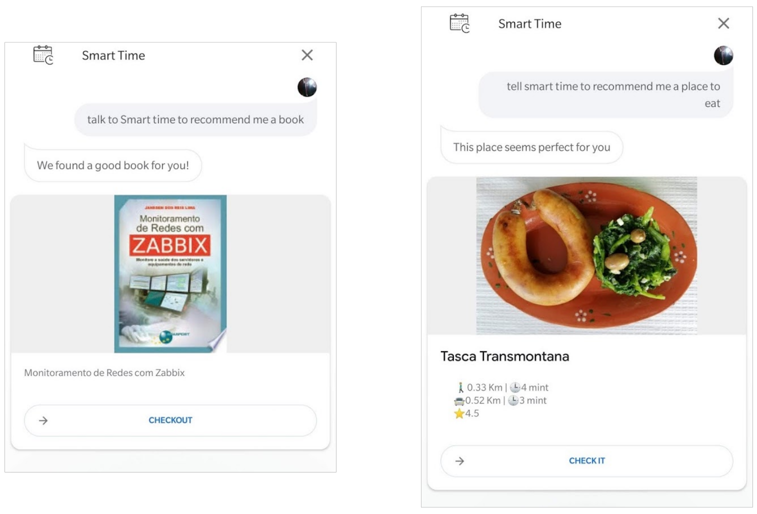
Biblioteca Municipal Afonso Lopes Vieira

0.10 Km | 1 mint
0.41 Km | 2 mint
4.2

CHECK IT

Enjoy your free time!
See you later!

Figure 6: Example of output of the generic recommendation


Smart Time

talk to Smart time to recommend me a book

We found a good book for you!

Monitoramento de Redes com Zabbix

CHECKOUT

Smart Time

tell smart time to recommend me a place to eat

This place seems perfect for you

Tasca Transmontana

0.33 Km | 4 mint
0.52 Km | 3 mint
4.5

CHECK IT

Figure 7: Example output of book and eat recommendation

5.4 Place to Eat / Visit Recommendation

Similarly to book recommendations, it is possible to ask for recommendations of places to eat or places to visit (Figure 7-right). In the information presented to the user it is possible to see a photo representation of the place/food recommended, how long it will take the user to reach it by car or on foot, and the place's rating. If the recommended place has no photo or rating, it is excluded from the search in order to make the Action feel more interactive and appealing.

6 Conclusion and Future Work

Smart Time is a context-aware Action on Google able to suggest/recommend various activities by extracting user context from physical and virtual sensors of mobile devices. If the user is an active Google Calendar user, *Smart Time* is able to suggest different locations and activities even when travelling. Some activity suggestions allow the user to see directly how long it takes to reach the place by foot or by car; this way, the user is always informed about every detail that the Action can give.

As for future work, Smart Time could be improved by adding more activity types, more training phrases and, possibly, a more powerful back-end. This last improvement would be particularly important given that, in the current implementation, all the processing takes place on the same cloud function – which means that if any request takes longer than expected the Assistant times-out and an informative error message is presented.

As mentioned in Section 4.2, the default recommendation model is a generic model that can be improved through user usage. However, at some point, the tree model may become inefficient. This is also an opportunity for employing data-mining techniques in user activity and preferences to provide better services via more powerful learning algorithms.

References

[ADB+99] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In Hans-W. Gellersen, editor, *Handheld and Ubiquitous Computing*, pages 304–307, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[IL17] H. N. Io and C. B. Lee. Chatbots and conversational agents: A bibliometric analysis. In *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 215–219, Singapore, December 2017. IEEE.

[MCM+15] Imran Memon, Ling Chen, Abdul Majid, Mingqi Lv, Ibrar Hussain, and Gencai Chen. Travel Recommendation Using Geo-tagged Photos in Social Media for Tourist. *Wireless Personal Communications*, 80(4):1347–1362, February 2015.

[MLB04] Bradford Mott, James Lester, and Karl Branting. Conversational Agents. In Munindar Singh, editor, *The Practical Handbook of Internet Computing*, volume 20042960. Chapman and Hall/CRC, September 2004.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[Reb18] Rebecca Sentance. The future of voice search: 2020 and beyond Econsultancy, 2018.

[SAD+19] Marta De Sousa, Cruz Amado, Ricardo Silva Domingues, Rodrigo Silva Alves, Carlos Fernando, and De Almeida Grilo. Emotion analyst. page 6, 2019.

[Sha18] Gabriel Shaoolian. Why Voice Search Will Dominate SEO In 2019 – And How You Can Capitalize On It, 2018.

[SSA13] Shahab Saquib Sohail, Jamshed Siddiqui, and Rashid Ali. Book recommendation system using opinion mining technique. In *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1609–1614, Mysore, August 2013. IEEE.

[TO19] Maggie Tillman and Britta O'Boyle. What is Google Assistant and what can it do? - Pocket-lint, June 2019.

[Vin19] James Vincent. Googles next version of Assistant will be dramatically faster, hitting Pixel phones first, May 2019.

[VKI+15] Aleksey Varfolomeyev, Dmitry Korzun, Aleksandrs Ivanovs, Henrihs Soms, and Oksana Petrina. Smart Space based Recommendation Service for Historical Tourism. *Procedia Computer Science*, 77:85–91, 2015.