Voyageur: An Experiential Travel Search Engine

Sara Evensen
Megagon Labs
sara@megagon.ai

Aaron Feng
Megagon Labs
aaron@megagon.ai

Alon Halevy
Megagon Labs
alon@megagon.ai

Jinfeng Li
Megagon Labs
jinfeng@megagon.ai

Vivian Li
Megagon Labs
vivian@megagon.ai

Yuliang Li
Megagon Labs
yuliang@megagon.ai

Huining Liu
Megagon Labs
yuliang@megagon.ai

George Mihaila
Megagon Labs
george@megagon.ai

John Morales
Megagon Labs
john@megagon.ai

Natalie Nuno
Megagon Labs
natalie.nuno@megagon.ai

Ekaterina Pavlovic
Megagon Labs
kate@megagon.ai

Wang-Chiew Tan
Megagon Labs
wangchiew@megagon.ai

Xiaolan Wang
Megagon Labs
xiaolan@megagon.ai

ABSTRACT

We describe Voyageur, which is an application of experiential search to the domain of travel. Unlike traditional search engines for online services, experiential search focuses on the experiential aspects of the service under consideration. In particular, Voyageur needs to handle queries for subjective aspects of the service (e.g., quiet hotel, friendly staff) and combine these with objective attributes, such as price and location. Voyageur also highlights interesting facts and tips about the services the user is considering to provide them with further insights into their choices.

CCS CONCEPTS

• **Information systems** → **Search interfaces**; **Structured text search**; *Information retrieval*; • **Computing methodologies** → **Information extraction**; *Natural language processing*.

KEYWORDS

experiential search; subjective database; linguistic domain; information extraction;

1 INTRODUCTION

The rise of e-commerce enables us to plan many of our future experiences with online search engines. For example, sites for searching hotels, flights, restaurants, attractions or jobs bring a wealth of information to our fingertips, thereby giving us the ability to plan our vacations, restaurant gatherings, or future career moves. Unfortunately, current search engines completely ignore any experiential aspect of the plans they are helping you create. Instead, they are primarily database-backed interfaces that focus on searching based on objective attributes of services, such as price range, location, or cuisine.

The need for experiential search. An experiential search engine is based on the observation that at a fundamental level, users seek to satisfy an experiential need. For example, a restaurant search is meant to fulfill a social purpose, be it romantic, work-related or a reunion with rowdy friends from college. A vacation trip plan is meant to accomplish a family or couple need such as a relaxing location with easy access to fun activities and highlights of local cuisine. To satisfy these needs effectively, the user should be able to search on experiential attributes, such as whether a hotel is romantic or has clean rooms, or whether a restaurant has a good view of the sunset or has a quiet ambience.

Table 1 shows a snippet from a preliminary study that investigates which attributes users care about. We asked human workers on Amazon Mechanical Turk [1] to provide the most important criteria in making their decisions in 7 common verticals. We then conservatively judge whether each criterion is an experiential one or not. As the table shows, the majority of attributes of interest are experiential.

To the best of our knowledge, online services for these domains today lack support for directly searching over experiential attributes. The gap between a users' experiential needs and search

Table 1: Experiential attributes in different domains.

Domain	%Exp. Attr	Some examples
Hotel	69.0%	cleanliness, food, comfortable
Restaurant	64.3%	food, ambiance, variety, service
Vacation	82.6%	weather, safety, culture, nightlife
College	77.4%	dorm quality, faculty, diversity
Home	68.8%	space, good schools, quiet, safe
Career	65.8%	work-life balance, colleagues, culture
Car	56.0%	comfortable, safety, reliability

capabilities raises an important challenge: *can we build search systems that place the experience the user is planning at the center of the search process?*

Challenges. Supporting search on experiential aspects of services is challenging for several reasons. First, the universe of experiential attributes is vast, their precise meaning is often vague, and they are expressed in text using many linguistic variations. The experience of "quiet hotel rooms" can be described simply as "quiet room" or "we enjoyed the peaceful nights" in hotel reviews. Second, by definition, experiential attributes of a service are *subjective* and personal, and database systems do not gracefully handle such data. Third, the experiential aspect of a service may depend on how they relate to other services. For example, a significant component of a hotel experience is whether it is close to the main destinations the user plans to visit. Finally, unlike objective attributes that can be faithfully provided by the service owner, users expect that the data for experiential attributes come from other customers. Currently, such data is expressed in text in online reviews and in social media. Booking sites have made significant efforts to aggregate and surface comments from reviews. Still, while these comments are visible when a user inspects a particular hotel or restaurant, users still cannot search by these attributes.

This paper describes Voyageur, the first search engine that explicitly models experiential attributes of services and supports searching them. We chose to build Voyageur in the domain of travel because it is complex and highly experiential, but its ideas also apply to other verticals.

The first idea underlying Voyageur is that the experiential aspects of the service under consideration need to be part of the database model and visible to the user. For example, when we model a hotel, we'll also consider the time it takes to get there from the airport and nearby activities that can be done before check-in in case of an early arrival. Furthermore, Voyageur will fuse information about multiple services. So the proximity of the hotel from the attractions of interest to the user is part of how the system models a hotel. Of course, while many of the common experiential aspects can be anticipated in advance, it is impractical that we can cover them all. Hence, the second main idea in Voyageur is that its schema should be easily extensible, it should be able to handle imprecise queries, and be able to fall back on unstructured data when its database model is insufficient.

Even with the above two principles, Voyageur still faces the challenge of selecting which information to show to the user about a particular entity (e.g., hotel or attraction). Ideally, Voyageur should display to the user aspects of the entity that are most relevant to the decision she is trying to make. Voyageur includes algorithms

for discovering items from reviews that best summarize an entity, highlight the most unique things about them, and useful actionable tips.

2 OVERVIEW OF VOYAGEUR

We illustrate the main ideas of Voyageur. Specifically, we show how Voyageur supports experiential queries and how these queries assist a user in selecting a hotel.

User scenario. Elle Rios is a marketing executive living in Tokyo. She is planning a vacation to *San Francisco* in early October. Her goal is to have a relaxing experience during the vacation. Her entire travel experience will be influenced by a variety of services, including flights, hotels, local attractions, and restaurants. Elle visits the Voyageur website and first enters the destination with the travel period. Voyageur then displays a series of screens with recommendations for each of these services.

In searching for hotels, Elle's experiential goal to have a relaxing stay is achieved by a balance of her *objective* constraints (her budget for hotels is *$250-350 per night*) and *subjective* criteria; she is an introvert and she knows that *quiet* hotels with *friendly staff* will help her relax[1]. In addition, Elle also cares about whether the hotel is conveniently located for reaching the famous and historic attractions she wants to visit and the high-quality vegetarian restaurants she is interested in.

Figure 1 shows a screenshot of Voyageur, where Elle can plan a trip that satisfies her requirements. (The screens for attraction and restaurant search are similar). The screenshot shows how Voyageur emphasizes the experiential aspects of trip planning. First, Voyageur allows users to express their subjective criteria (in addition to their objective criteria) and generates recommendations accordingly through the *experiential search* function (Box A). Second, Voyageur tailors the display of *interesting facts and tips* and *summary of reviews* based on the search criteria entered (Boxes B and C). Third, Voyageur supports a series of additional features, such as map view and travel wallet, to further improve the user's experience in the hotel search (Box D). The map view provides a holistic view of the trip by putting together all recommended entities of different types. The travel wallet, as we explain later, takes into consideration the user's travel history and preferences if the user chooses to share them with Voyageur.

Experiential search. Elle expresses her objective and experiential/subjective criteria as query predicates to Voyageur's interface. While the objective requirements like "$250 to 300 per night" can be directly modeled and queried in a typical hotel database, answering predicates like *"quiet"* and *"friendly staff"* is challenging as these are subjective terms and cannot be immediately modeled in a traditional database system. Voyageur addresses this challenge with a subjective database engine that explicitly models the subjective attributes and answers subjective query predicates. Voyageur extracts subjective attributes such as *room quietness* and *staff quality* from hotel customer reviews, builds a summary of the variations of these terms, and then matches those attributes with the input query predicates.

---

[1]Elle can also directly search for hotels with relaxing atmosphere in Voyageur.
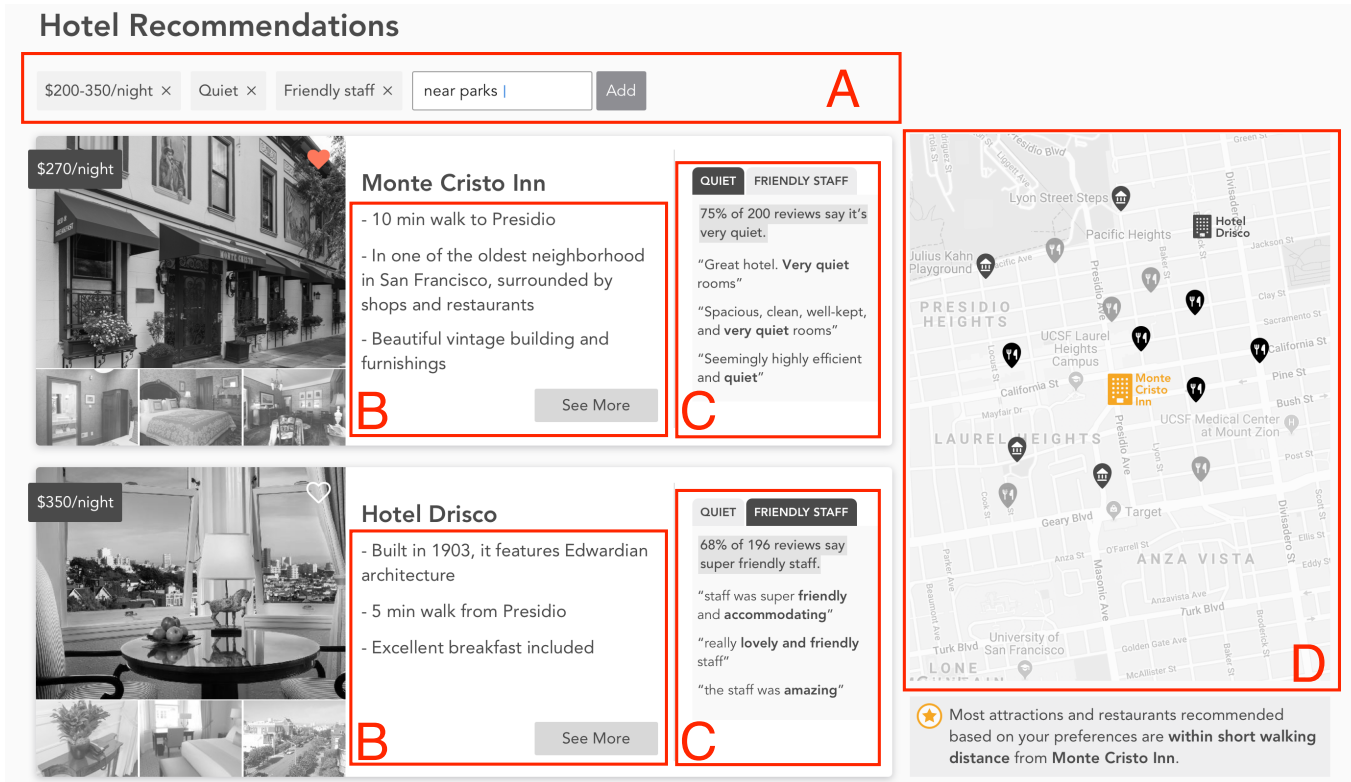
Figure 1: A screenshot of the hotel recommendation screen of VOYAGEUR. The user can interactively customize the search (Box A), to view interesting facts/tips (Box B) and a *review summary* (Box C) of each recommended hotel and to check out a *map view* (Box D) of the recommendations.

In Figure 1, VOYAGEUR generates a ranked list of hotels by matching the query predicates specified by Elle with the subjective attributes extracted from the underlying hotel reviews. The review summaries (Box C) show that the selected hotels are clearly good matches. Specifically, VOYAGEUR recommends Monte Cristo since 75% of 200 reviewers agree that it is very quiet and it has friendly staff (not shown). Hotel Drisco, next on the list, is recommended because 68% of 196 reviewers agree that it has friendly staff and the it is also very quiet (not shown).

Interesting facts and tips. Along with the search results, VOYAGEUR shows snippets of travel tips and/or interesting facts of each result (Box B) it thinks is relevant for Elle. An interesting fact typically highlights an unusual or unique experience about the service. For example, being very close to Presidio Park (one of the largest parks in San Francisco) is unique to Monte Cristo Inn and Hotel Drisco and is thus an interesting fact to show for each hotel. The fact that Monte Cristo Inn is a "beautiful vintage building and furnishings" is unique only to Monte Cristo Inn. Such interesting facts can be important for decision making. It also enables Elle to better anticipate the type of experiences that will be encountered at the hotel [8]. On the other hand, tips are snippets of information that propose a potential action the user may take to either avoid a negative experience or create a positive one [2]. For example, a useful tip for a hotel may be that there is free parking two blocks away.

While the interesting facts and tips are useful, they are not always available for every service and can be incomplete. Existing work

[2, 6] proposed mining useful travel tips from customer reviews with promising results. In VOYAGEUR, we formulate the problem of finding tips and interesting facts as a query-sentence matching problem to find tips and interesting facts relevant to the users' query. Our algorithms prefer to select sentence snippets from reviews to match the user's query. The challenge we face is that users' query predicates and the tips/facts in reviews are described in different vocabularies and linguistic forms. Moreover, labeled data for the matching task is generally not available. Thus, novel techniques are needed to construct good matching functions between queries and sentence snippets.

Review summarization. The summary of reviews (Box C) provides Elle with an explanation of why the specific hotel is recommended and the summary saves her from reading the repetitive and lengthy reviews. VOYAGEUR summarizes the reviews of each recommended hotel in two different formats: (1) statistical statements and (2) sample review snippets. For example, in Figure 1, VOYAGEUR summarizes the room quietness attribute of Monte Cristo Inn with the statistical statement "75% of 200 reviews say it is very quiet" and 3 randomly selected sample review snippets that match the *quiet* requirement.

Additional features. The following features further improve Elle's ability to create a positive travel experience:

- **Map view.** In each recommendation screen, a map view (Box D of Figure 1) marks the locations of the recommended

3513

hotels. Whenever a hotel is selected, the map view is centered at a chosen hotel and shows the recommended local attractions and restaurants so the user can better plan how to travel between these places.

- **Travel wallet.** Users have the option of creating a *travel wallet*, which is similar to the Wallet feature on many smartphones. It contains information about the user that she shares only when she chooses to. In the case of a travel wallet, this information records her travel preferences. The travel wallet can be created explicitly by answering questions or can be collected automatically from previous travels. The travel wallet is used by Voyageur to further personalize the search results.

- **Trip summary.** Finally, after making several choices (flight, hotel, attractions, etc.) through all the recommendation screens, the user can view a summary of the trip underlying the key experiential components. In Elle's case, the summary includes a timeline with important dates, transportation methods to/from the airport, and tips/facts about the chosen hotel and each planned tourist attractions.

3 IMPLEMENTATION OF VOYAGEUR

We briefly touch upon the technology underlying Voyageur.

3.1 A subjective database engine

As mentioned, the main challenge of building a successful experiential search engine is the modeling and querying of the *subjective attributes*, where there is typically no ground truth to the values of such attributes. Examples of such attributes include the cleanliness of hotel rooms, quality of the food served, and cultural value of tourist attractions. They are not explicitly modeled in today's search engines and therefore not directly queryable.

Voyageur is developed on top of OpineDB [3], a *subjective database engine*. OpineDB goes beyond traditional database engines by supporting the modeling, extraction, aggregating, and effective query processing of the subjective data. Next, we illustrate the key design elements of OpineDB by showcasing its application to hotel search in Voyageur.

Data model, extraction, and aggregation. The main challenge in modeling subjective attributes is the wide range of linguistic variations with which the attributes are described in text. Consider an attribute room_quietness of hotels. The review text can be of various forms such as (1) "*the neighborhood seems very quiet at night*", (2) "*on busy street with traffic noise*", or (3) "*quiet and peaceful location*". In addition, OpineDB needs to *aggregate* these phrases into a meaningful signal for answering queries, which may themselves include new linguistic variations.

OpineDB models subjective attributes with a new data type called the *linguistic domain* and provides an aggregated view of the linguistic domain through a *marker summary*. The linguistic domain of an attribute contains all phrases for describing the attribute from the reviews. E.g., "quiet at night", "traffic noise", and "peaceful location". A subset of phrases is then chosen as the *domain markers* (or *markers* for short) for each linguistic domain. The phrases are aggregated based on the markers to constitute the marker summary.

For quietness, the markers might be [very_quiet, average, noisy, very_noisy]. To construct the marker summary of a hotel's quietness, OpineDB needs to assign each quietness phrase to its closest marker and compute the frequencies of the markers. For example, the summary [very_quiet:20, average: 70, noisy:30, very_noisy:10] for a hotel would represent that the hotel is closer to being average in quietness than to the other markers.

The linguistic domain is obtained by extracting phrases from reviews. Various techniques are available for this task in opinion mining and sentiment analysis [4, 7]. The marker summaries are currently histograms computed from the extraction relations. However, we can also leverage more complex aggregate functions.

Query processing. The query predicates from Box A is formulated as an SQL-like query for OpineDB to process.

> **select**   *   **from** Hotels *h*
> **where**   price_pn ≤ 350 **and** price_pn ≥ 200 **and**
>   "*quiet*" **and** "*friendly staff*"

Here, price_pn is an objective attribute of the Hotels relation while "*quiet*" and "*friendly staff*" are subjective predicates. OpineDB needs to interpret these predicates using the linguistic domains in order to find the best subjective attributes of the Hotels relation that can be used to answer them. In general, this is not a trivial matching problem since the query terms may not be directly modeled in the schema. For example, the user may ask for "romantic hotels", but the attribute for romance might not be in the schema. For such cases, OpineDB leverages a combination of NLP and IR techniques to find a best-effort reformulation of the query term into a combination of schema attributes. For example, for "romantic hotels", OpineDB will match it to a combination of "exceptional service" and "luxurious bathrooms" which are modeled by the schema.

After computing the interpretation, OpineDB uses the marker summaries to compute a *membership score* for each pair of hotel and query predicate. Finally, OpineDB combines multiple predicates using a variant of fuzzy logic.

3.2 Mining interesting facts and tips

We formulate the problem of finding useful travel tips and interesting facts as a query-sentence matching problem. We adopt an approach similar to an existing work for mining travel tips from reviews [2]. The approach consists of a *filtering phase* and a *ranking phase*.

Filtering phase. This phase constructs a set of candidate tip/fact sentences by applying filters and classifiers to all the review sentences. According to [2], effective filters for tips include phrase patterns (e.g, sentences containing "make sure to") and part-of-speech tags (e.g, sentences starting with verbs). For constructing the candidate set of interesting facts, we select sentences that contain a least one *informative token*, which are words or short phrases frequently mentioned in reviews of the target entity but not frequently mentioned in reviews of similar entities. We also found that an interesting fact is more likely to appear in sentences with an extreme sentiment (very positive or negative). So we also apply sentiment analysis to select such sentences. For both tips and interesting facts, we further refine the candidate sets by removing duplicates, i.e., sentences of similar meaning or the unimportant

ones. We do so by applying TextRank [5], a classic algorithm for sentence summarization.

Ranking phase. Instead of simply selecting candidates for interesting facts/tips, sets, we implemented a novel ranking function for finding candidates that best match the user's query predicates. The ranking function considers not only the *significance* of a candidate as computed in the filtering phase but also the *relevance* of the candidate with the query. Measuring the relevance is not trivial since the tips/interesting facts can use vocabularies different from the ones used in the query. In the previous example, a fact that matches the query "near park" is "10 min walk to Presidio" which has no exact-matched word. The similarity function leverages a combination of NLP and IR techniques, analogous to query interpretation in OpineDB.

3.3 Datasets and tools

The Voyageur demo will serve hotels, attractions, and restaurants search in the San Francisco area in a browser-based web app. We collected the underlying data from two datasets: the Google Local Reviews[2] and the TripAdvisor datasets[3]. Our dataset consists of 18,500 reviews of 227 hotels, 6,256 reviews of 545 attractions and 67,382 reviews of 4,007 restaurants. We implemented the front end of Voyageur using the JavaScript library React.

4 CONCLUSION

The motivation for Voyageur is based on the discrepancy between the needs of users searching for services and the current state of search engines. The ideas of Voyageur are applicable to many other verticals beyond travel. At the core of the technical challenges that Voyageur and systems like it need to address is the ability to discover and aggregate evidence from textual reviews in response to user queries. This is a technical challenge that draws upon techniques from NLP, IR and Database technologies.

REFERENCES

[1] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. 2011. Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science* 6, 1, 3–5.
[2] Ido Guy, Avihai Mejer, Alexander Nus, and Fiana Raiber. 2017. Extracting and Ranking Travel Tips from User-Generated Reviews. In *WWW*. 987–996.
[3] Yuliang Li, Aaron Xixuan Feng, Jinfeng Li, Saran Mumick, Alon Halevy, Vivian Li, and Wang-Chiew Tan. 2019. Subjective Databases. *CoRR* abs/1902.09661.
[4] Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Morgan & Claypool.
[5] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text. In *EMNLP*. 404–411.
[6] Sapna Negi, Maarten de Rijke, and Paul Buitelaar. 2018. Open Domain Suggestion Mining: Problem Definition and Datasets. *CoRR* abs/1806.02179.
[7] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. SemEval-2016 task 5: Aspect based sentiment analysis. In *SemEval-2016*. 19–30.
[8] Jonathan Skinner and Dimitrios Theodossopoulos. 2011. *Great expectations: Imagination and anticipation in tourism*. Vol. 34. Berghahn Books.

---

[2]http://cseweb.ucsd.edu/~jmcauley/datasets.html#google_local
[3]http://nemis.isti.cnr.it/~marcheggiani/datasets/