With One Voice: Composing a Travel Voice Assistant from Repurposed Models

Shachaf Poran  
shachaf.poran@booking.com  
Booking.com, Tel Aviv

Amit Beka  
amit.beka@booking.com  
Booking.com, Tel Aviv

Gil Amsalem  
gil.amsalem@booking.com  
Booking.com, Tel Aviv

Dmitri Goldenberg  
dima.goldenberg@booking.com  
Booking.com, Tel Aviv

ABSTRACT
Voice assistants provide users a new way of interacting with digital products, allowing them to retrieve information and complete tasks with an increased sense of control and flexibility. Such products are comprised of several machine learning models, like Speech-to-Text transcription, Named Entity Recognition and Resolution, and Text Classification. Building a voice assistant from scratch takes the prolonged efforts of several teams constructing numerous models and orchestrating between components. Alternatives such as using third-party vendors or re-purposing existing models may be considered to shorten time-to-market and development costs. However, each option has its benefits and drawbacks. We present key insights from building a voice search assistant for Booking.com. Our paper compares the achieved performance and development efforts in dedicated tailor-made solutions against existing re-purposed models. We share and discuss our data-driven decisions about implementation trade-offs and their estimated outcomes in hindsight, showing that a fully functional Machine-Learning product can be built from existing models.

CCS CONCEPTS
• Software and its engineering → Software design tradeoffs; • Information systems → Search interfaces; Speech / audio search.

KEYWORDS
Voice, Search, Recommendation, Machine Learning Architecture

1 INTRODUCTION
Voice assistants have become a prevailing mode of communication between customers and companies [12, 20]. Today, you can pick up your smart device and utter a request or a command to do things we wouldn’t have dreamt of in the past. The most appealing aspect of this feature is the transfer of touch and typing interfaces into spoken commands, conveying your request in free language and making the action easy to perform and almost instantaneous. For example, you can simply ask a question rather than navigating a verbose FAQ page, or you can use the voice interface when you have limited hand dexterity[4]. Using voice assistants in search and recommendation tasks serves various customer expectations and needs. Introducing a free-form speech input allows customers to generate unstructured queries, resulting in a complex input to the search and recommendation systems [6, 15]. The unstructured nature of natural language also allows users to explore different options in their apps that otherwise would be hidden for the sake of simplicity. The user would have to reach these options using buttons and menus that involve more attention and more steps to progress through a digital experience [7].

A voice assistant relies on a function $v : U \to A$ that maps an utterance $u \in U$ provided by the user to an action $a \in A$ which can be performed by the app, aiming to fulfill the user’s intent which was presented in the utterance. 1 Example for such a mapping in the groceries domain might be:

$$v(\text{we are out of milk}) = \text{place order for milk}$$

and in the travel domain (relevant in our case) it might be:

$$v(\text{I need to book a hotel in Paris}) = \text{present a list of hotels in Paris}$$

The actions taken by the app may include searching for accommodation, finding inspiration for upcoming travels, asking for help, amending existing bookings, etc. The function $v$ may be seen as a chain of auxiliary functions starting with transforming the raw voice input to text. Only then is the natural language processed to extract the intent of the user and the entities mentioned in the text. Eventually, a decision is made about which action to perform. In practice, the former two steps are realized using Machine Learning models. In creating these Machine Learning elements, there’s a point of decision around how the research and development teams implement them[24]. Options include but are not limited to:

1. In-house development
2. Open-source frameworks

1Conversational assistants may have additional context which is out our paper scope.
(3) Fine-tuning pre-trained models
(4) Pre-trained models used as-is
(5) Third-party vendors

Each of these options entails implicit costs, whether monetary, development time, or how well the results fit the business needs. The above list starts with the more lengthy and costly development times that on the other hand should also result in a more specialized model[24]. These costs however are difficult to know in advance and might vary wildly depending on the kind of problem to be solved, existing expertise in the workforce, and demand for high accuracy metrics for the models. Moreover, recent work in the online travel domain has shown that an improvement in an offline metric does not necessarily reflect business impact [8, 21], and requires online validation via a randomized controlled experiment [17]. At the same time, orchestrating a cascade of machine learning models requires a supporting software system designed to allow a combination of business logic with ML-driven decisions [25]. Productionizing an existing model from a prototype stage to a ready-to-use product might require additional effort in its serving, and result in misaligned business metrics.

Another concern when choosing one of these options over another revolves around domain-specific data and knowledge [7]. The first three options in the list above require having data available for training and evaluation, while the last two do not. Having the same distribution of data when training a model and when using it for inference is considered good practice, and a significant mismatch between the two might lead to accuracy metrics being irrelevant. Knowledge of these distributions in advance might in some cases lead to using different modeling techniques and better performance.

Constructing a voice assistant usually require a complex architecture, and a generous investment in research and development [9, 14]. At the same time, repurposing existing state-of-the-art ML models towards new applications [22] becomes a popular solution for various product needs. We suggest to adopt a well-known software reuse paradigm [5], that allows to achieve high quality and reduce development time [19] by repurposing existing machine learning components, or considering using external third-party off-the-shelf services [3, 23]. In this paper, we share insights regarding these challenges and how decisions were made in the process of developing a mobile voice assistant (see Figure 1 for overview of the product flow).

Our key contributions are evidence-based comparisons of dedicated tailor-made solutions against repurposing existing models for different Machine Learning tasks. The paper demonstrates how to overcome the lack of in-domain data and to compose a Machine Learning product without training new models, all the while not compromising on potential impact. We share and discuss our data-driven decisions about implementation trade-offs, and their estimated outcomes in hindsight, by examining the main four components of the system: Voice-to-Text, Translation, Named Entity Resolution, and Text Classification. The ML pipeline we developed is summarized in a flowchart in Figure 2. To understand user queries voice systems are composed of a voice-to-text element followed by a language understanding element [26]. The voice-to-text system is discussed in section 2. For our use case we chose to construct the language understanding element with three steps, as will be discussed in section 3, section 4, and section 5. section 6 concludes our findings and discusses opportunities for future research.

2app is available at https://www.booking.com/apps.en-gb.html
2 VOICE-TO-TEXT (VTT)

The first ML-driven element of the pipeline has an utterance as a waveform as input and outputs transcribed text for the uttered speech. It is worthwhile discerning how the distribution of inputs may vary between domains, as it may determine the performance of pre-trained models versus models that were trained on in-domain data. For example:

1. **Sampling rate** - values may be either 8KHz or 16KHz
2. **Background noises** - mechanical hum, vehicle noise, colleagues chatter for call-centers, music etc.
3. **Accents** - e.g. differences in American vs British English
4. **Word distribution** - usage of wordings in different contexts

Item 4 in the list is especially relevant as VTT systems use statistics of words and n-grams to decide their likelihood. Different domains may exhibit differences in word frequencies that affect accuracy. Even when disregarding domains, different dialects may have a similar effect. We evaluated two options for the VTT element, One is an open-source framework (OSF) which comes out-of-the-box with ready-made models and tools to tweak them, and the other is a third-party vendor (TPV). Prior comparisons between an open-source framework and an off-the-shelf third-party vendor tool [16] have shown a trade-off between domain-specific accuracy and generalization and productization capabilities.

Developing any model without data generated from the end product produces a classical "chicken or the egg" problem since we cannot infer data distribution. A common practice in this scenario is to use data from an adjacent domain or product to train models. We obtained recordings from customer-service conversations for bootstrapping. Using an annotation tool built in-house, we collected ground-truth transcriptions for these conversations and used them to compare the different models. The metric we used to compare different VTT models and tools was Word Error Rate (WER)[27]. After launching the voice assistant product, we gathered data that allowed another iteration of annotation and comparison.

Both TPV and OSF allow for the tweaking of results: the former receives a set of hint phrases that may appear in the utterance; and the latter allows fine-tuning modular steps in its VTT pipeline including an acoustic model, a lexicon, and a language model using in-domain data. We tweaked both of them to achieve the lowest WER we could with either. OSF out-of-the-box model achieved 45.01% WER, compared to 25.25% WER by the TPV. An effort to tweak OSF model eventually resulted in 28.99% WER resembling similar comparisons with open access datasets [16]. Tweaking TPV resulted in negligible boost in performance. At this point, a product decision was made to use the TPV and defer the OSF development indefinitely since other parts of the product development were waiting to be taken on by the team. After releasing the product real-world data was gathered and the two models were reevaluated based on it. Table 1 reports both evaluations, showing that the performance is better for the OSF for utterances taken directly from the product. The same table presents error rates for specific words in the text, explaining some of the difference in performance between the two datasets by the higher abundance of domain-specific words in the latter. Table 2 shows common errors by TPV that were transcribed much more accurately by the OSF.


![Figure 2: Overview of the voice assistant architecture.](image)

Table 1: Comparison of WER for the TPV vs. the OSF model on the adjacent domain and in-domain data-sets.

<table>
<thead>
<tr>
<th>Error word</th>
<th>Data source</th>
<th>TPV</th>
<th>OSF</th>
</tr>
</thead>
<tbody>
<tr>
<td>All words</td>
<td>Conversations</td>
<td><strong>25.25%</strong></td>
<td>28.99%</td>
</tr>
<tr>
<td></td>
<td>App commands</td>
<td><strong>45.24%</strong></td>
<td>38.68%</td>
</tr>
<tr>
<td>booking</td>
<td>App commands</td>
<td><strong>198/415 (47.7%)</strong></td>
<td><strong>31/415 (7.5%)</strong></td>
</tr>
<tr>
<td>cancellation</td>
<td>App commands</td>
<td><strong>46/108 (42.6%)</strong></td>
<td><strong>23/108 (21.3%)</strong></td>
</tr>
</tbody>
</table>

Table 2: Examples of domain-specific errors from the TPV which the OSF model got correct.

<table>
<thead>
<tr>
<th>TPV</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact hotel for registration details</td>
<td>reservation</td>
</tr>
<tr>
<td>Can I have the information Consolation</td>
<td>confirmation cancellation</td>
</tr>
</tbody>
</table>

3 TRANSLATION

The work described in section 2 focused on English. When expanding to new markets, the voice assistant is expected to support local languages. Every new language once again faces the same problems already discussed in the previous section, and the time and effort to create the relevant models does not scale well as practically all stages should be repeated, including data collection and annotations. Using the TPV allowed us to transcribe numerous languages easily, but downstream models were all trained using English inputs. Lack of multilingual training data presented a serious hold-back, which led us to translate the transcriptions before passing them forward [10]. An in-house team has been developing in-domain translation models described in [18]. These models showed consistent results independent of sentence length, which hints that using it for our use case is acceptable. We easily interfaced with their service and served multiple languages with nearly zero effort.

The incisive time to enable new languages has proven essential for testing new markets. Aside from model performance, which may differ for each language, user habits for using voice assistants vary with country and culture. Presenting the product to users was key to understanding product-market fit [1, 2].

4 NAMED ENTITY RECOGNITION

Named entity recognition (NER) is an NLP task that asks the question about a word or a sequence of words whether they are “a person, a place or a thing”. Named Entity Resolution is a task that asks “what person, place, or thing is it”. In our context, resolution matches a recognized entity to a closed set of destinations such as countries, cities, villages, and hotels. Any human hearing the utterance “I’m looking for a hotel in Amsterdam” will assume the
When we fulfill searches for flights, both for origin and destination, this has accelerated our time-to-market, allowing us to present the finished product to our users quickly.

Entity Resolution is a highly specialized task involving annotation of a substantial amount of in-domain data for both recognition and resolution sub-tasks[11]. This task is essential for a voice assistant in the travel domain. However, anything other than using a ready-made solution would be infeasible and delay deployment of the product for a long time. An in-house team has been developing an Entity Resolution API for use in a chat-bot product. By the time we came to use it for the voice assistant, it was at near-SOTA performance with more than 90% F1 score for the recognition task. We performed a qualitative inspection and interfaced with the API. This has accelerated our time-to-market, allowing us to present the finished product to our users quickly.

5 TEXT CLASSIFICATION

In this step of the pipeline, the text is fed into a multi-class classification model and converted into enumerated cases to be used by the client to initiate the response appropriate for the user’s utterance. As a free-form input method, we expected utterances that address both searching for an accommodation to book (“pre-book” intents) and for treating existing bookings (“post-book” intents). User surveys confirmed that, with a distribution of 50% pre-book intents, 30% post-book, and the rest are other intents such as greetings and nonsensical queries. This revealed that we have two main sub-domains to address when building the text classification. Once again, training any model before collecting any data is not feasible. To allow product development and eventually lead to data collection we used two different internal models that serve these features:

- **Travel Assistant**: a text-input interface used to guide users through the FAQ on the site and app. Their NLP model maps text to post-book intents [15]
- **Chat bot**: the tool described in section 4. As support to their NER model, they had models to decide whether a user wants to book a flight or a hotel (or neither).

Interleaving both of these models using simple rules allowed us to efficiently serve both pre-book and post-book sub-domains with one client-facing interface. The logic we used for combining the two into a single cohesive product is shown in Figure 3. Simple if-else statements based on the two models result in either an action, such as a flight search being conducted or an FAQ page being opened, or in giving the user feedback within the voice UI asking for clarification or additional information. We complemented the process with an exact keyword search, such as credit being mapped to payment intent, for words we found are significantly correlated with customer-service intents. This works exceptionally well after the upstream steps filter out most of the other intents.

After the voice assistant feature was made available to the customers, we collected data and annotated it. Intent distribution is given in Table 3. We proceeded to build a model to map text directly to intent using the Natural Language Processing Framework (NLPF). The classification metrics to compare the composite business model to the NLPF model are shown in Table 4. It shows an improvement when using NLPF. These two options were tested in an A/B test setting by comparing human-handled customer service tasks, which confirmed that the NLPF model results in a reduction in human-handled tickets which was statistically significant.

6 CONCLUSION

A common perception of the Data Scientists’ work, is that their first order of business is training Machine Learning models to fit the specific tasks at hand, starting with data gathering and ending in a custom model. In the first version of the voice assistant released, we have not used any Machine Learning models custom-made for this product. None of the models we used were trained on relevant in-domain data. Instead, we composed the product from a chain of ready-made models and services. Our decisions to do so were motivated by data wherever it was applicable. Though one might argue that the VTT decision was wrong as the discarded model performance on in-domain data was better than the model we kept, this

---

**Table 3**: Distribution of intents in our annotated data. The "other" category includes also pre-booking intents.

<table>
<thead>
<tr>
<th>Intents</th>
<th>Frequency</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>None/unclear/other</td>
<td>587</td>
<td>41.0%</td>
</tr>
<tr>
<td>Cancel booking</td>
<td>198</td>
<td>13.8%</td>
</tr>
<tr>
<td>Change booking</td>
<td>169</td>
<td>11.8%</td>
</tr>
<tr>
<td>Payments</td>
<td>110</td>
<td>7.7%</td>
</tr>
<tr>
<td>Status booking</td>
<td>101</td>
<td>7.1%</td>
</tr>
<tr>
<td>Rest of intents</td>
<td>267</td>
<td>18.6%</td>
</tr>
</tbody>
</table>

**Table 4**: Precision (p) and recall (r) of topic classification models on the most common intents.

<table>
<thead>
<tr>
<th>Intent</th>
<th>Composite</th>
<th>NLPF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p</td>
<td>r</td>
</tr>
<tr>
<td>Cancel booking</td>
<td>79%</td>
<td>70%</td>
</tr>
<tr>
<td>Change booking</td>
<td>79%</td>
<td>48%</td>
</tr>
<tr>
<td>Payments</td>
<td>46%</td>
<td>50%</td>
</tr>
</tbody>
</table>
is a non-issue since the end product has proved beneficial despite the shortcoming of this element in the chain of models. Moreover, making the product available to our users - which would have been blocked without these ready-made models - is a crucial element in building more accurate models in the future. Development of the entire end-to-end process took about four months. At the same time, from the time already spent on each of the ML tasks either by the Voice Assistant team or the other teams, we estimate that the development of the same product from scratch would have taken approximately two years. Deploying the voice assistant has benefited the company business metrics two-fold, both by increasing engagement and reservations, but also by reducing the work for customer service, as users found the solutions to their problems more easily when using the voice free-form interface.

To conclude our observations, we recommend to break down complex machine learning architectures into atomic sub-tasks. Contrary to an initial urge to develop a novel tailor-made solution, we found that re-purposing existing solutions can often achieve effective results, while efficiently saving development and scaling efforts. Moreover, reusable system components drive long-term system alignments and achieve services and organizational synergy. We invite our peers to be aware of the option of building Machine Learning-centered products without ever having to train a single model, but rather to save valuable time and effort by using the work already done by peers and colleagues.

ACKNOWLEDGMENTS

We would like to thank Steven Baguley, Nam Pham, Nick Tax, Guy Nadav and David Konopnicki for their feedback during writing of this paper and to Chalom Asseraf, Noa Barbiro, Sophie Galante, Rubens Sonntag, Gora Shecherbakov, Dor Samet, Teri Hason and the rest of the contributors to the Voice Assistant product at Booking.com.

REFERENCES