With One Voice: Composing a Travel Voice Assistant from Re-purposed Models

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

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

Amit Beka
amit.beka@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 search and recommendation system. 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.

KEYWORDS

Voice, Search, Recommendation, Machine Learning Architecture

1 INTRODUCTION

Voice assistants have become a prevailing mode of communication between customers and companies [16, 23]. Today you can pick up your smart device and utter a request or a command and the device complies, a thing 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 [6]. Using voice assistants in search and recommendation tasks serves various customer expectations and needs [9]. Introducing a free-form speech input allows customers to generate unstructured queries, resulting in a complex input to the search and recommendation systems [17]. The unstructured form of natural language also allows users to explore different options in their apps that otherwise would be hidden for the sake of simplicity of the graphical user interface. The user would have to reach these options using buttons and menus that involve more attention and more steps to progress through [10].

A voice assistant relies on a function $v : U \rightarrow A$ 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] An example for such a mapping in the groceries domain might be:

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

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

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

The actions taken by the Booking.com app may include searching for accommodation, seeking travel inspiration, asking for help, amending existing bookings, recommending a property, 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 [7, 32]. 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 about how the research and development teams implement them [29]. Options include but are not limited to those shown in Figure 1.

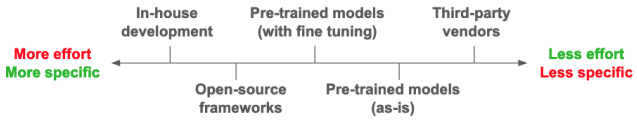

Figure 1: Different options to implementing machine learning models and the trade-off between effort of implementation and specificity of the resulting solution.

Each of these options entails implicit costs, whether monetary, development time, or how well the results fit the business needs. The items on the left are the ones that are more lengthy and costly in development time, but on the other hand they should also result in more specialized models [29]. These costs are difficult to estimate in advance and might vary widely depending on the kind of problem to be solved, existing expertise in the workforce, and demand for

[1] Conversational assistants may have additional context which is out scope for this paper.

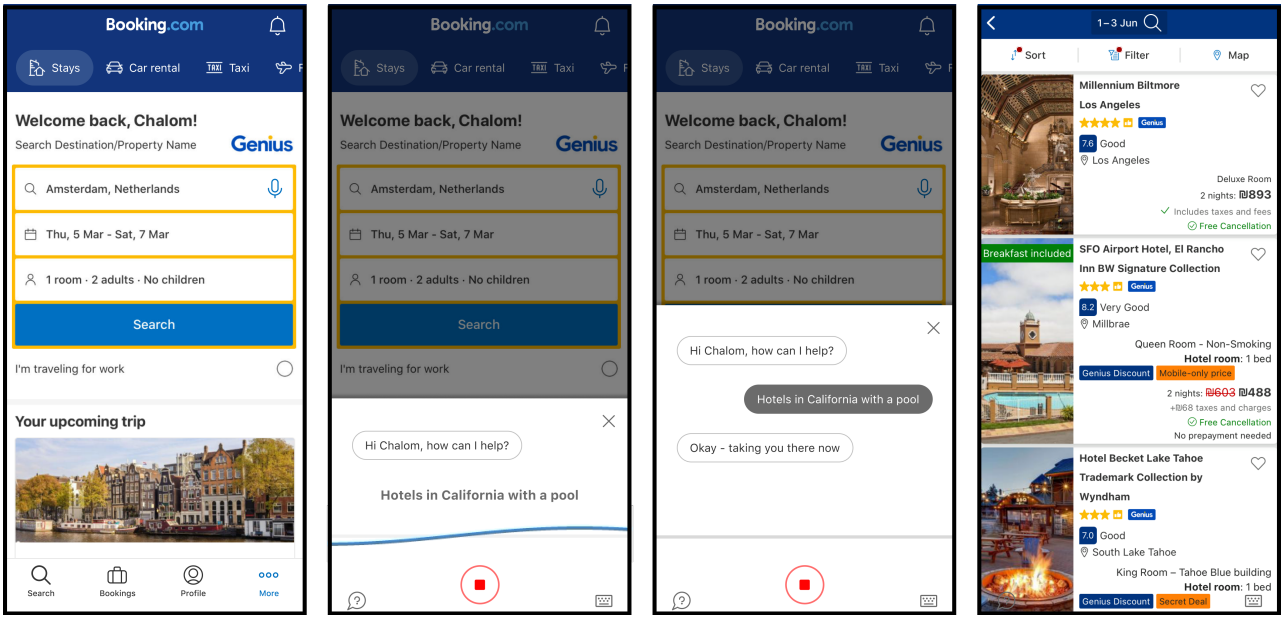

Figure 2: Voice assistant flow, screens from left to right: Index entry point, Search prompt, Input Query, and Search Results.

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, and requires online validation via a randomized controlled experiment [4, 11]. 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 [30].

Another concern when choosing one of these options over another revolves around domain-specific data and knowledge [10]. The three leftmost options in Figure 1 require having data available for training and evaluation, while the other 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 [12, 18]. At the same time, re-purposing existing ML models towards new applications [26] becomes a popular solution for various product needs. We suggest to adopt a well-known software reuse paradigm [8], that allows to achieve high quality and reduce development time [22] by re-purposing existing machine learning components or considering using external third-party off-the-shelf services [5, 27]. 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 2 for an overview of the product flow). [2]

[2]app is available at https://www.booking.com/apps.en-gb.html

Our key contributions are evidence-based comparisons of dedicated tailor-made solutions against re-purposing 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 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 assistant systems are composed of a voice-to-text element followed by a language understanding element [31]. The ML pipeline we developed is summarized in the flowchart shown in Figure 3. The voice-to-text (VTT) system is discussed in Section 2. For our use case we chose to construct the language understanding element with three steps: Machine Translation (MT), Named Entity Resolution (NER), and Text Classification (TC). These steps are discussed in Section 3, Section 4, and Section 5 respectively. The output of the last element is fed into a downstream recommender systems [24]. Section 6 concludes our findings and discusses opportunities for future research.

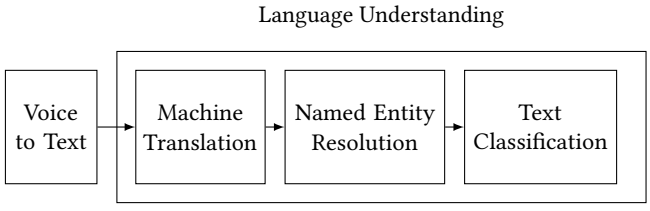

Figure 3: Overview of the voice assistant architecture.

Data source	Error word	TPV	Kaldi
Conversations	All words	**25.25%**	28.99%
App commands	All words	45.24%	**38.68%**
App commands	"booking"	198/415 (47.7%)	**31/415 (7.5%)**
App commands	"cancellation"	46/108 (42.6%)	**23/108 (21.3%)**

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

2 VOICE-TO-TEXT

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** such as mechanical hum and vehicle noise for car assistants, colleagues chatter for call-centers, etc.
(3) **Accents** such as differences in American vs British English.
(4) **Word distribution** differs 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 the open-source framework Kaldi [28] 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 Kaldi and an off-the-shelf third-party vendor tool [20] have shown higher accuracy for Kaldi when testing on in-domain data and when the signal-to-noise ratio is high.

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 was Word Error Rate (WER) [25, 33], defined as the edit distance between ground truth and predicted sentences normalized by the length of ground truth. This is a common metric used in voice-to-text systems.

TPV	Correct
Contact hotel for *registration* details	reservation
Can I have the *information*	confirmation
Consolation	cancellation

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

Both TPV and Kaldi allow for the tweaking of their models: 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. We tweaked both of the alternatives by using our adjacent-domain data to achieve the lowest WER we could with either. Kaldi's out-of-the-box model achieved 45.01% WER, compared to 25.25% WER by the TPV. The effort to tweak Kaldi model resulted in 28.99% WER, resembling similar comparisons with open-access datasets [20]. Tweaking TPV resulted in a negligible boost in performance. At this point, a decision based on currently-available data was made to use the TPV and defer any additional Kaldi development indefinitely.

After releasing the voice assistant feature real-world data was gathered and annotated, and the two models were reevaluated based on it. Table 1 reports both evaluations, showing that the performance is better for the Kaldi model 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 these domain-specific words in the latter. Table 2 shows common errors by TPV that were transcribed accurately by the Kaldi.

3 MACHINE 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 [13].

An in-house team has been developing in-domain translation models described in [21]. 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 [2, 3].

4 NAMED ENTITY RESOLUTION

Named Entity Recognition (NER) is a Natural Language Processing (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

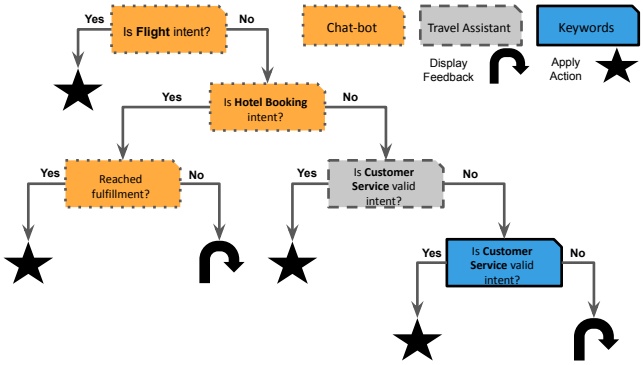

Figure 4: The business logic used to align different classifiers for the text classification task.

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 speaker intends to visit the Dutch city. However, we are aware that there are different Amsterdams in the United States, South Africa, and elsewhere. Furthermore, we expect two entities to be resolved when we fulfill searches for flights, both for origin and destination.

Entity Resolution is a highly specialized task involving annotation of a substantial amount of in-domain data for both recognition and resolution sub-tasks [14]. 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 would delay deployment of the product for a long time.

An in-house team has been developing an Entity Resolution API based on the FLAIR framework [1] 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 finalized 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. Some of the cases were treated as templates and fulfilled with entities resolved from the NER model, for example searching for an accommodation was fulfilled with the destination.

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 [19]
- **Chat bot**: the tool described in Section 4. As support to the NER model, it used a different model to decide whether a user wants to book a flight or a hotel (or neither).

Interlacing 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 4. 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 element 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. *coronavirus* is yet another example for such a keyword, which forwarded users to an explanation about the COVID-19 situation in regard to their bookings. Keyword matching works exceptionally well for our use case as the upstream steps filter out most of the other intents.

After the voice assistant feature was made available to the customers, we collected data directly from their interactions and annotated it. Intent distribution, excluding 40% of unintelligible utterances, is given in Table 3. We proceeded to build a model to map text directly to intent using the NLP framework spaCy [15]. The classification metrics to compare the composite business model to the spaCy model are shown in Table 4. These two options were tested in an randomized controlled experiment with two groups, each exposed to a different text classifier. Measuring The number of customer service tasks that were handled by representatives for each of the groups confirmed that the spaCy model results in a reduction in such human-handled tasks which was statistically significant.

Intents	Prevalence
Pre-book intents	66.9%
Request human agent	8.7%
Check booking status	7.1%
Payments	3.0%
Change booking	1.9%
Other post-book intents	10.0%
Greetings	2.4%

Table 3: Distribution of intents in our annotated data.

Intent	Composite		spaCy	
	p	r	p	r
Cancel booking	**79%**	**70%**	**79%**	64%
Change booking	79%	**48%**	**87%**	31%
Payments	46%	50%	**52%**	**75%**

Table 4: Per-class precision (p) and recall (r) of topic classification models on the most common intents.

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. Conversely, in the first version of the voice assistant that was released we have not used any machine learning models custom-made for this product, and 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, and by time-to-market considerations otherwise. Though one might argue that the VTT decision was wrong as the discarded model performance on in-domain data was better than the TPV tool we used, this 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. From the time already spent on developing models for the MT, NER, and TC tasks by other teams, and the time spent on the VTT task and improving on the TC model by our team, we estimate that the development of the same product from scratch would have taken approximately two years if taken up by a single team.

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 representatives, 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, Niek Tax, Guy Nadav and David Konopnicki for their feedback during writing of this paper. We also thank Chalom Asseraf, Noa Barbiro, Rubens Sonntag, Giora Shcherbakov, Dor Samet, Teri Hason and the rest of the contributors to the Voice assistant product at Booking.com.

REFERENCES

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 54–59.

[2] Etienne Barnard, Johan Schalkwyk, Charl Van Heerden, and Pedro J Moreno. 2010. Voice search for development. (2010).

[3] Frank Bentley, Chris Luvogt, Max Silverman, Rushani Wirasinghe, Brooke White, and Danielle Lottridge. 2018. Understanding the long-term use of smart speaker assistants. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 3 (2018), 1–24.

[4] Lucas Bernardi, Themistoklis Mavridis, and Pablo Estevez. 2019. 150 Successful Machine Learning Models: 6 Lessons Learned at Booking.com. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1743–1751.

[5] Markus Borg, Panagiota Chatzipetrou, Krzysztof Wnuk, Emil Alégroth, Tony Gorschek, Efi Papatheocharous, Syed Muhammad Ali Shah, and Jakob Axelsson. 2019. Selecting component sourcing options: a survey of software engineering's broader make-or-buy decisions. *Information and Software Technology* 112 (2019), 18–34.

[6] Eric Corbett and Astrid Weber. 2016. What Can I Say? Addressing User Experience Challenges of a Mobile Voice User Interface for Accessibility *(MobileHCI '16)*. Association for Computing Machinery, New York, NY, USA, 72–82.

[7] Thierry Desot, François Portet, and Michel Vacher. 2019. Towards End-to-End spoken intent recognition in smart home. In *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*. 1–8. https://doi.org/10.1109/SPED.2019.8906584

[8] Yang Fuqing, Mei Hong, and Li Keqin. 1999. Software Reuse and Software Component Technology [J]. *Acta Electronica Sinica* 2 (1999).

[9] Dmitri Goldenberg, Kostia Kofman, Javier Albert, Sarai Mizrachi, Adam Horowitz, and Irene Teinemaa. 2021. Personalization in Practice: Methods and Applications. In *Proceedings of the 14th International Conference on Web Search and Data Mining*.

[10] Ido Guy. 2016. Searching by talking: Analysis of voice queries on mobile web search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 35–44.

[11] Malay Haldar, Mustafa Abdool, Prashant Ramanathan, Tao Xu, Shulin Yang, Huizhong Duan, Qing Zhang, Nick Barrow-Williams, Bradley C Turnbull, Brendan M Collins, et al. 2019. Applying deep learning to Airbnb search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1927–1935.

[12] Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al. 2019. Streaming end-to-end speech recognition for mobile devices. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6381–6385.

[13] Georg Heigold, Vincent Vanhoucke, Alan Senior, Patrick Nguyen, Marc'Aurelio Ranzato, Matthieu Devin, and Jeffrey Dean. 2013. Multilingual acoustic models using distributed deep neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 8619–8623.

[14] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. 782–792.

[15] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. *spaCy: Industrial-strength Natural Language Processing in Python*. https://doi.org/10.5281/zenodo.1212303

[16] Matthew B Hoy. 2018. Alexa, Siri, Cortana, and more: an introduction to voice assistants. *Medical reference services quarterly* 37, 1 (2018), 81–88.

[17] Jie Kang, Kyle Condiff, Shuo Chang, Joseph A Konstan, Loren Terveen, and F Maxwell Harper. 2017. Understanding how people use natural language to ask for recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 229–237.

[18] Veton Kepuska and Gamal Bohouta. 2018. Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home). In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 99–103.

[19] Talaat Khalil, Kornel Kiełczewski, Georgios Christos Chouliaras, Amina Keldibek, and Maarten Versteegh. 2019. Cross-lingual intent classification in a low resource industrial setting. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 6420–6425.

[20] Takashi Kimura, Takashi Nose, Shinji Hirooka, Yuya Chiba, and Akinori Ito. 2018. Comparison of speech recognition performance between Kaldi and Google cloud speech API. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. Springer, 109–115.

[21] Pavel Levin, Nishikant Dhanuka, and Maxim Khalilov. 2017. Machine translation at booking.com: Journey and lessons learned. *arXiv preprint arXiv:1707.07911* (2017).

[22] Jingyue Li, Anita Gupta, Jon Arvid, Borretzen Borretzen, and Reidar Conradi. 2007. The empirical studies on quality benefits of reusing software components. In *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, Vol. 2. IEEE, 399–402.

[23] Alex Mari. 2019. Voice Commerce: Understanding shopping-related voice assistants and their effect on brands. (2019).

[24] Themis Mavridis, Soraya Hausl, Andrew Mende, and Roberto Pagano. 2020. Beyond algorithms: Ranking at scale at Booking. com. In *ComplexRec 2020, Recommendation in Complex Scenarios and the Impact of Recommender Systems*.

[25] Sonja Nießen, Franz Josef Och, Gregor Leusch, and Hermann Ney. 2000. An Evaluation Tool for Machine Translation: Fast Evaluation for MT Research. In

Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00). European Language Resources Association (ELRA), Athens, Greece. http://www.lrec-conf.org/proceedings/lrec2000/pdf/278.pdf

[26] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.

[27] Kai Petersen, Deepika Badampudi, Syed Muhammad Ali Shah, Krzysztof Wnuk, Tony Gorschek, Efi Papatheocharous, Jakob Axelsson, Severine Sentilles, Ivica Crnkovic, and Antonio Cicchetti. 2017. Choosing component origins for software intensive systems: In-house, COTS, OSS or outsourcing?—A case survey. *IEEE Transactions on Software Engineering* 44, 3 (2017), 237–261.

[28] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The Kaldi Speech Recognition Toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding* (Hilton Waikoloa Village, Big Island, Hawaii, US). IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

[29] Johan Schalkwyk, Doug Beeferman, Françoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Kamvar, and Brian Strope. 2010. "your word is my command": Google search by voice: A case study. In *Advances in speech recognition*. Springer, 61–90.

[30] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. *Advances in neural information processing systems* 28 (2015), 2503–2511.

[31] Jessica Van Brummelen, Kevin Weng, Phoebe Lin, and Catherine Yeo. 2020. CONVO: What does conversational programming need?. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*.

[32] Ye-Yi Wang, Li Deng, and Alex Acero. 2011. *Semantic Frame Based Spoken Language Understanding*. Wiley, 35–80. https://www.microsoft.com/en-us/research/publication/semantic-frame-based-spoken-language-understanding/

[33] Ye-Yi Wang, A. Acero, and C. Chelba. 2003. Is word error rate a good indicator for spoken language understanding accuracy. In *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No.03EX721)*.