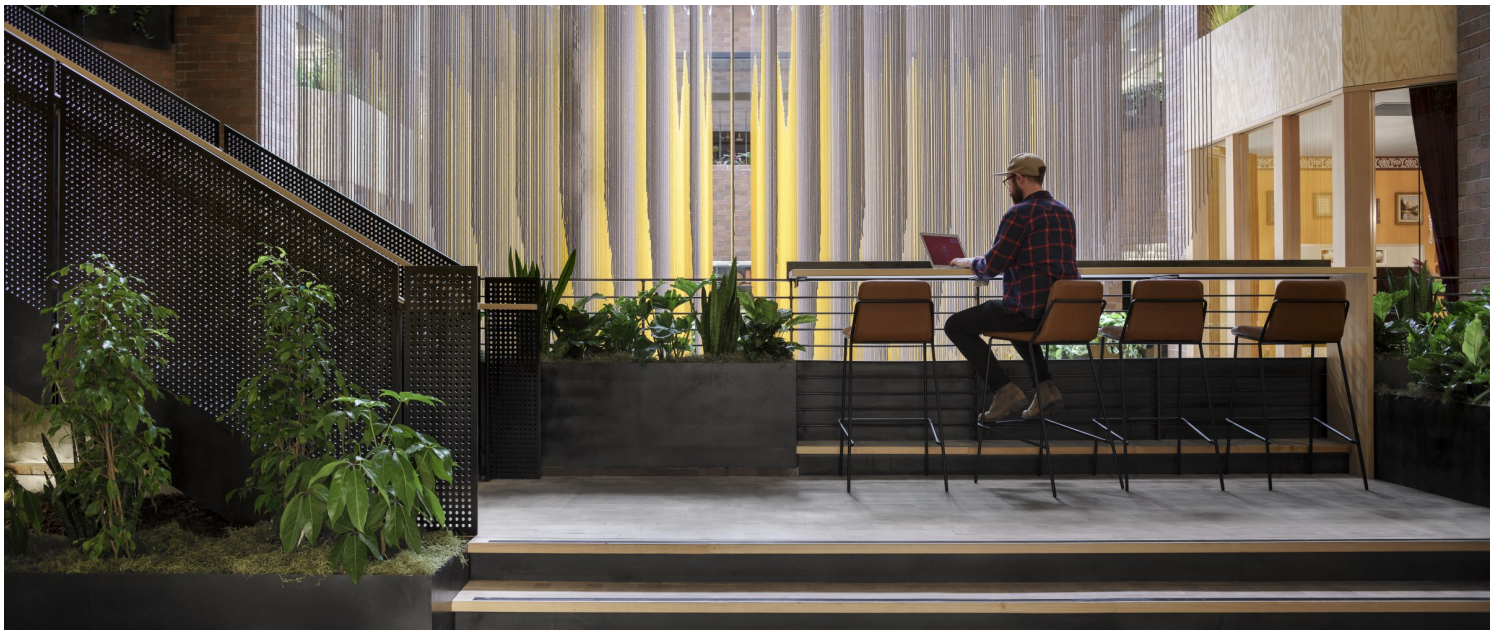
Discovering and Classifying In-app Message Intent at Airbnb

Conversational AI is inspiring us to rethink the customer experience on our platform.

Michelle (Guqian) Du Follow
Jan 22, 2019 · 11 min read

Authors: *Michelle Du*, *Shijing Yao*



Get embraced by plenty of natural light, brick, and plant in our new office in downtown Seattle.

Airbnb's mission is to create a world where everyone can belong anywhere. Ensuring good communication between guests and hosts is one of the keys to developing a sense of belonging as well as a smooth and worry-free trip experience for guests. Millions of guests and hosts communicate on the Airbnb messaging platform about a variety of topics, including booking arrangements, payment requests, trip planning, service feedback, and even sharing experiences with new friends. Thus, a huge opportunity to

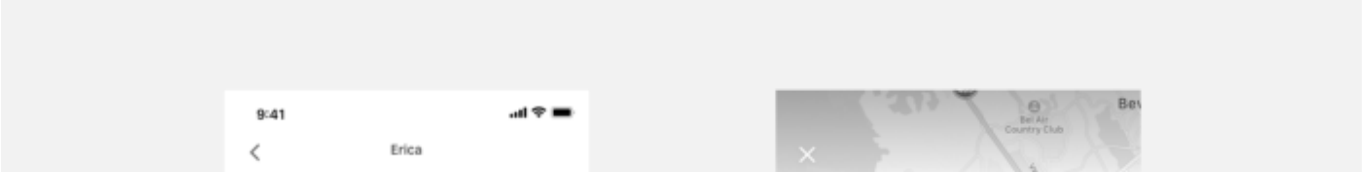
improve the experience for guests on the platform is to predict and understand the intent of their messages to hosts.

Consider the following situation. Christmas is two weeks out. You are planning a last-minute family trip to Hawaii, and you find a sweet beach house in Honolulu on Airbnb. The listing description doesn't show how many beds are available. In the Airbnb mobile app, you ask "Does your house have enough beds to accommodate six people?", and anxiously wait for the host to reply. However, the host is too busy to reply immediately. You get worried because you may miss out on other listings while waiting for the reply.

In another situation, you booked a summer trip with your best friend to Paris well in advance of when you plan to travel. Yet unexpectedly, just a few days before the trip, your friend tells you that you may need to change the schedule because she got injured. You are thinking of canceling the booking, but you are not sure if a full refund will be issued. You ask the host through an in-app message about the cancellation policy, and hope they can reply quickly. However, you have to wait for many hours because it is now midnight in Paris time.

We recognize these scenarios can cause anxiety and confusion, and believe there are ways to address them in a much better way. In the aforementioned two cases, answering questions in a real-time fashion is especially desirable. When inconvenient situations like these arise, Airbnb's in-app messaging platform is a critical channel to facilitate communications. On the other hand, requiring all hosts to instantly respond to guests places lots of burden on them, not to mention that this is unrealistic.

Using recent conversational AI technologies, three Airbnb teams — the Shared Products, Applied Machine Learning, and Machine Learning Infrastructure — have developed a machine learning framework together that can mitigate the problem. The framework is capable of automatically classifying certain guest messages to help us better understand guest intent. Therefore, it can help greatly shorten the response time for guests and reduce the overall workload required for hosts. It also allows Airbnb to provide essential guidance and thus a seamless communication experience for both guests and hosts.



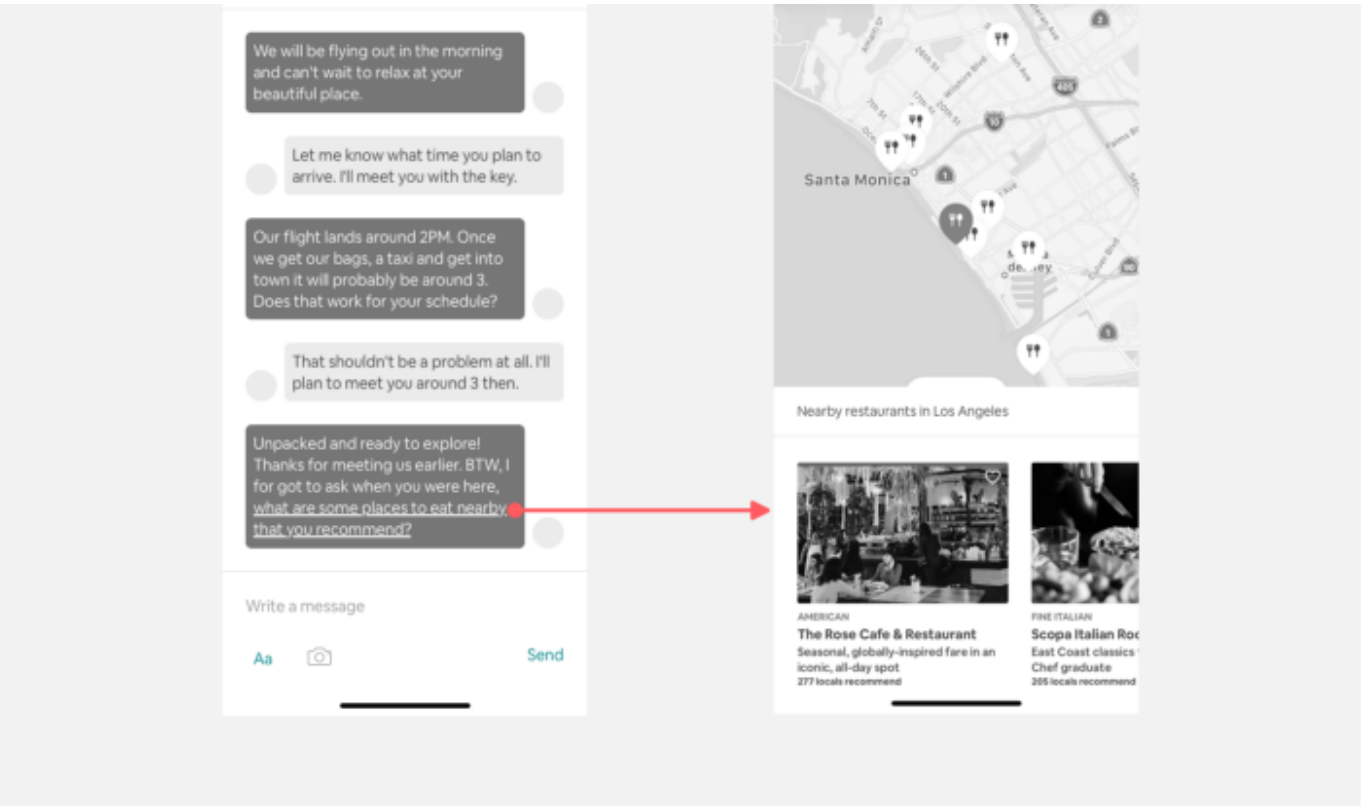


Figure 1: A concept that illustrates a guest asking a host for dining recommendations nearby.

Identifying Message Intent

Behind every message sent is an intent, be it to work out logistics, clarify details, or connect with the host. To improve upon the existing communication experience, it is vital for the AI to identify this "intent" correctly as a first step. However, this is a challenging task because it is difficult to identify the exhaustive set of intents that could exist within millions of messages.

To address this challenge, we set up our solutions in two phases: In **Phase 1**, we used a classic unsupervised approach — Latent Dirichlet Allocation (LDA) — to discover potential topics (intents) in the large message corpus. In **Phase 2**, we moved to supervised learning techniques, but used the topics derived from Phase 1 as intent labels for each message. Specifically, we built a multi-class classification model using a canonical Convolutional Neural Network (CNN) architecture. The two phases create a powerful framework for us to accurately understand the text data on our messaging platform.

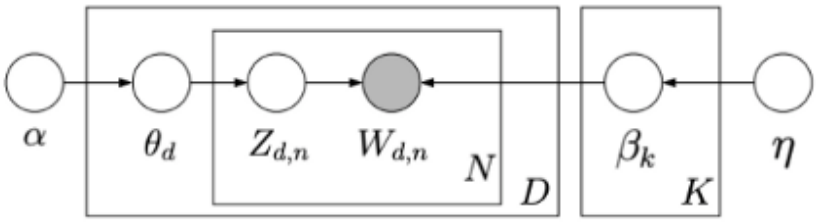
Intent Discovery

The first challenge in this problem is to discover existing topics (intents) from the enormous messaging corpus without prior knowledge. One might think of using

embedding techniques to generate message-level clusters and thus topics. However a key assumption here is that only one primary topic exists in a message, which does not hold for Airbnb data. On Airbnb, people tend to set up context before they start to type core messages, and it is common to have one message containing several pieces of information that are not quite relevant to each other.

Here is an example. A guest actually wants to ask how they could store the luggage for early check-in. But they may tell the host about their arrival time first before asking the real check-in question. For humans, it is relatively easy to decompose the topics and figure out that the key topic is "early check-in possibility". For embedding methods however, neither one single embedding vector nor algebraic aggregations of several different embedding vectors could represent the key topic. What we really need is an algorithm that can detect distinct underlying topics, and decide which one is the primary one based on probability scores.

Thus LDA becomes a natural choice for our purposes. First, LDA is a probabilistic model, which gives a probabilistic composition of topics in a message. Second, LDA assumes each word is drawn from certain word distribution that characterizes a unique topic, and each message can contain many different topics (see Figure 2 below for a graphical model representation along with the joint distribution of the variables). The word distribution allows human judgement to weigh in when deciding what each topic means.



$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d) \prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n})$$

Figure 2: A graphical model representation of LDA by David Blei et al. along with the joint probabilities of the observed (shaded nodes) and hidden (unshaded nodes) units

Figure 3 shows a 2D visualization of the generated topics using pyLDAvis. We determined the number of topics (hyperparameter *K*) in LDA to be the one generating the highest coherence score on the validation set.

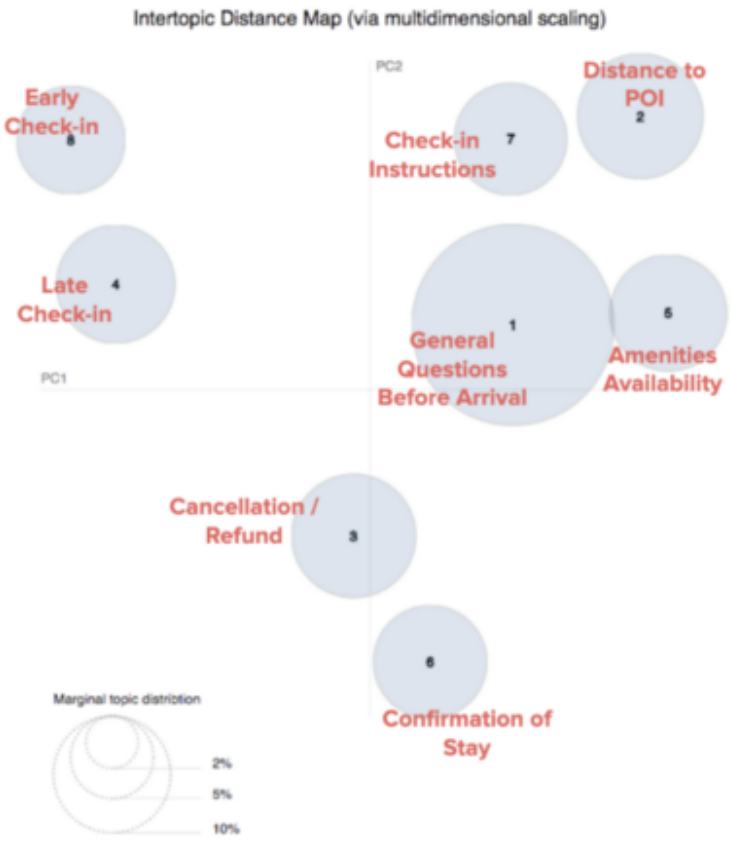


Figure 3: A 2D visualization of inter-topic distances calculated based on topic-term distribution and projected via principal component analysis (PCA). The size of the circle is determined by the prevalence of the topic.

Due to time constraints, we didn't invest much time in methods like doc2vec and BERT. Even though these methods have constraints as mentioned above, they do take the word order into account and could be attractive alternatives for intent discovery purposes. We remain open to these methods and plan to revisit them at a later time.

Labeling: From Unsupervised to Supervised

Labeling is a critical component of Phase 2, as it builds up a key transition from an unsupervised solution to a supervised one. Even though a sketch of the intent space in Phase 1 has already been detected, we do not have full control of the granularity due to its unsupervised nature. This is particularly problematic if certain Airbnb product needs to address specific message intents which may not have been detected in Phase 1. It is also hard to evaluate the efficacy of LDA results for each message without a clearly predefined intent label for that message as the ground truth.

Just as intent discovery, the first challenge of labeling is to determine what labels to define. More importantly, we need to ensure that the quality of the labels is high. Our solution is to perform an iterative process that starts from the topics discovered from LDA, but leverages product feedback to generate a final set of labels. First, we *pilot labeled* a small sample by having each message labeled by multiple people to evaluate labeling quality. We then refined the label definitions based off the *inter-rater agreement* for each intent label, and kicked off formal labeling with a much larger data size. During the formal round, each message is reviewed once for the majority of the data. We keep a small portion of messages that are labeled by multiple reviewers so that we could *identify the limits in prediction accuracy that our model could achieve due to human-level error*. Each message is completely anonymized with Personally identifiable information (PII) scrubbed throughout the process.

In terms of labeling resource, we secured our internal product specialists who were able to provide high-quality labeling service for the message data. The labeling service turned out to be much more customizable and reliable compared with third-party vendors, and also exemplified a great collaboration between different organizations of a company.

During the labeling process, we found that about 13% of our target messages has multi-intent. **Multi-intent** is a situation where people ask questions with two or more different intents in one single message. When multi-intent occurred, we asked our specialists to assign each specific intent to the corresponding sentences. Sentences assigned with one single intent were used as an independent training sample when building the intent classification model. We demonstrate how they are handled in real-time serving in the *Productionization* section (Figure 6).

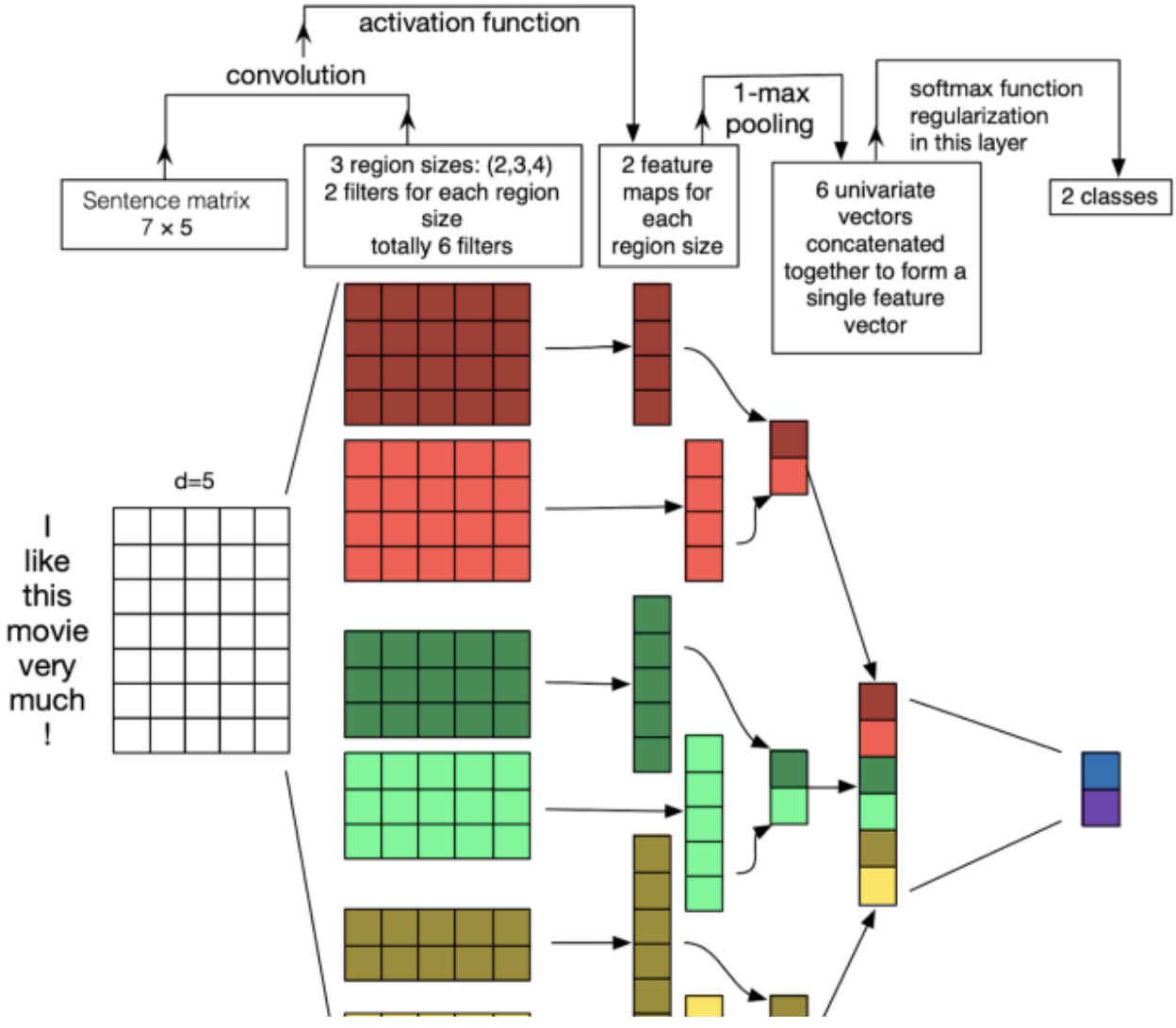
Intent Classification with CNN

Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) have been very popular methods for NLP tasks. In this work we focus on CNN due to its implementation simplicity, reported high accuracy, and especially fast speed (at both training and inference time). Piero Molino et al., 2018 showed that Word CNN performs less than 1% worse than the Char C-RNN on the same dataset and hardware, while being about 9 times faster during both training and inference. In our case, it takes us 10 minutes on average for the validation error to converge while it takes 60 minutes on

average for RNN to converge to the same level. This results in a much slower model iteration and development when taking hyperparameter tuning into account.

In terms of model accuracy, Wenpeng Yin, et al.,2017 did a thorough comparison of CNN and RNN on different text classification tasks. They found that CNN actually performs better than RNN when the classification is determined by some key phrases rather than comprehending the whole long-range semantics. In our case, we usually do not need to have the full context of a guest's message in order to identify the intent of their question. Rather, the intent is mostly determined by the key phrases such as "how many beds do you have?" or "is street parking available?"

After extensive literature review, we decided to adopt Yoon Kim,2014 and Ye Zhang et al.,2016, where a simple one-layer CNN followed by one 1-max pooling layer was proposed. Unlike the original work, we designed 4 different filter sizes each with 100 filters.



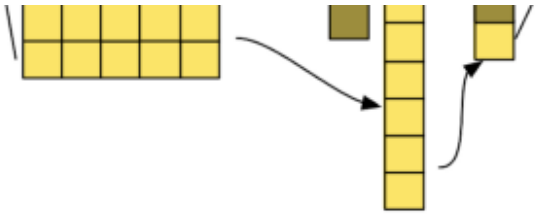


Figure 4: Illustration of a CNN architecture for sentence classification from Ye Zhang et al.

To prepare for the embedding layer, we pre-trained word embeddings based on large out-of-sample Airbnb messaging corpus. We performed careful text preprocessing, and found that certain preprocessing steps such as tagging certain information are especially helpful in reducing noise as they normalize information like URLs, emails, date, time, phone number, etc. Below is an example of the most similar words for the word `house` generated by word2vec models trained without and with such preprocessing steps:

```
# top similar words for “house” from model trained WITHOUT extra preprocessing steps
[('nuts', 0.9999956488609314),
 ('quinceaera', 0.9999409317970276),
 ('kaladi', 0.9999126195907593),
 ('troubles', 0.9998962879180908),
 ('rafford', 0.9998959302902222),
 ('ridden', 0.9998912215232849),
 ('marne', 0.9998871088027954),
 ('malinois', 0.999783456325531),
 ('home', 0.9997491836547852),
 ('jullie', 0.9997130632400513)]

# top similar words for “house” from model trained WITH extra preprocessing steps
[('cottage', 0.8544829487800598),
 ('guesthouse', 0.832261323928833),
 ('cabin', 0.8258084058761597),
 ('casita', 0.7988039255142212),
 ('townhouse', 0.7875640392303467),
 ('condo', 0.7709039449691772),
 ('bungalow', 0.7637494802474976),
 ('apartment', 0.7626785039901733),
 ('property', 0.7332403659820557),
 ('yurt', 0.718646228313446)]
```

top_similar_words.py hosted with ❤ by GitHub view raw

Most similar words for “house” generated by word2vec models trained without / with extra preprocessing steps

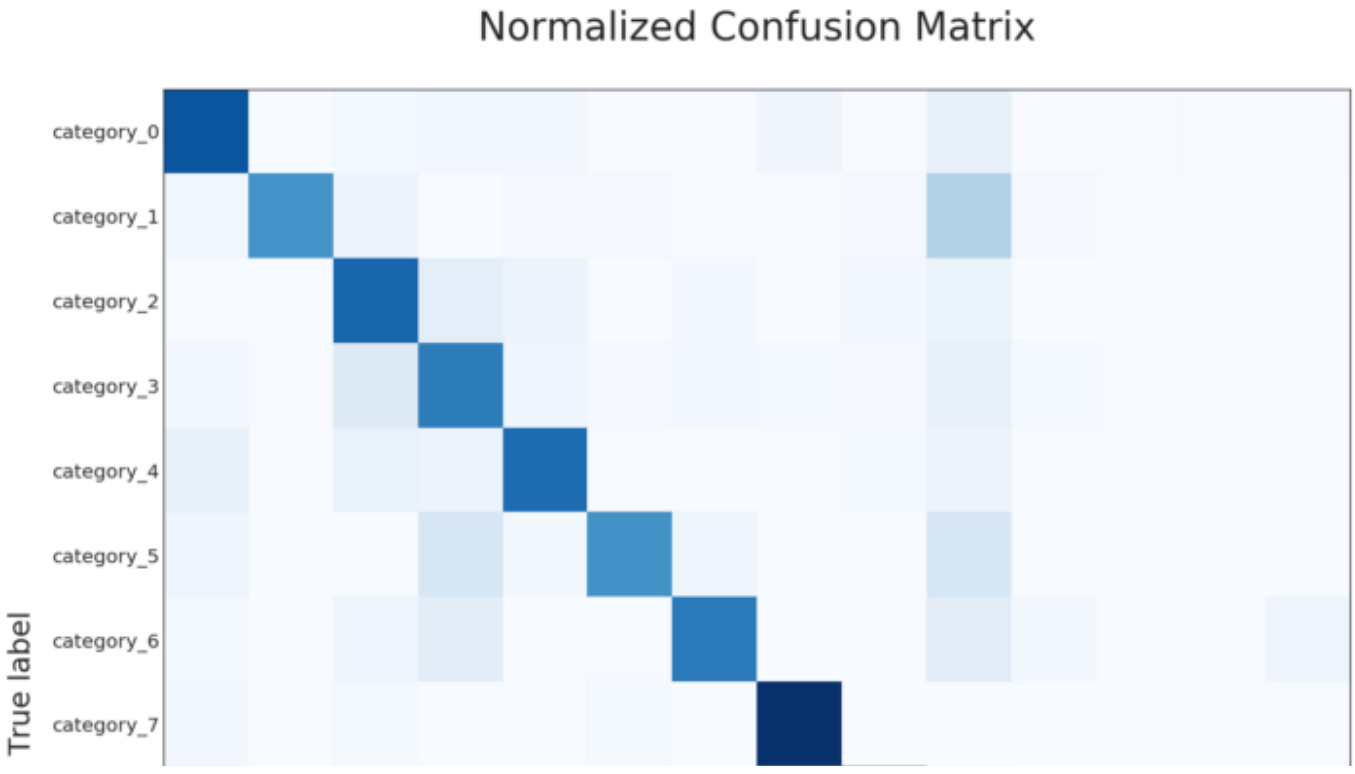
To be consistent, we used the same preprocessing steps throughout training word embeddings, offline training for message intent classifier, as well as online inference for real-time messages. Our to-be-open-sourced Bighead Library made all these feasible.

The overall accuracy of the Phase-1&2 solution is around 70% and outperforms the Phase-1 only solution by a magnitude of 50–100%. It also exceeds the accuracy of predicting based on label distribution by a magnitude of ~400%.

Trip Stage	Phase-1 & 2	Phase-1 Only	Predict by Label Distribution
Pre-trip	68%	43%	10%
On-trip	71%	32%	12%

Table 1: Comparison on overall accuracy between Phase-1&2, Phase-1 Only, and Predict by Label Distribution. Pre-trip: before a trip starts. On-trip: during a trip.

We evaluated classification accuracies class by class, especially when the dataset was imbalanced across different classes. Figure 5 is the confusion matrix for the on-trip model mentioned above. We masked the actual category name with `category_1`, `category_2`, etc. due to confidentiality. As one may see, a clear diagonal pattern can be found, which means the majority of the class predictions matches the ground truth.



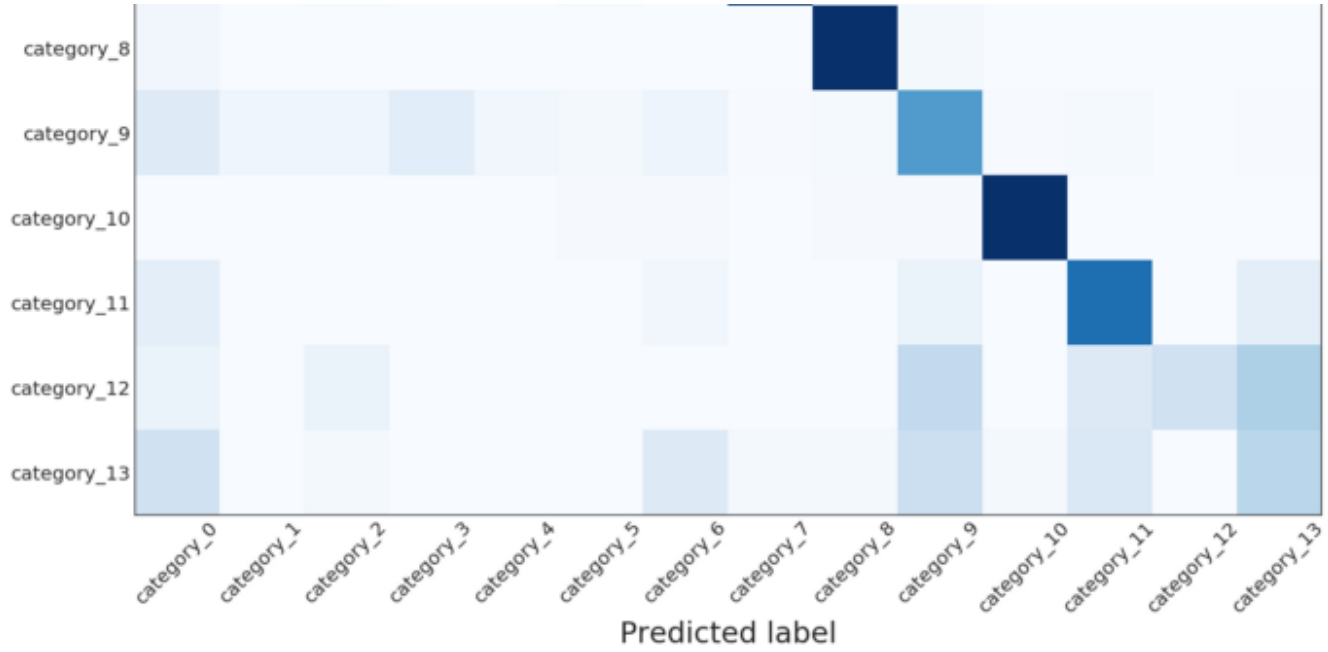


Figure 5: The normalized confusion matrix for the on-trip model results

Table 2 shows some example categories that are well predicted. In these categories, the key phrases are strong indicators of message intent that the CNN model captures very well.

Intent Category	Precision	Recall
Parking Situation	90%	92%
Internet Availability and Access	86%	86%
Check-in & Check-out Time Negotiation	74%	84%
Amenities Questions	79%	82%

Table 2: Example categories that are well predicted

Table 3 below shows some example categories that are not well predicted.

Intent Category	Precision	Recall
Recommendations for Generic Activities	40%	56%
Recommendations for Specific Activities	48%	33%

Table 3: Example categories that are not so well predicted

There were two primary root causes for the misclassifications:

1. Human errors in labeling. For example, some labelers mistakenly think that “Do you have recommendations on hiking or boat tours?” is a general question, but this type of questions are considered specific questions in our categories.

2. Label ambiguity. For example, “Could you recommend some things to do in the area? We were looking to go to a public beach or lake”, can be labeled as a generic question because the first sentence, “Could you recommend some things to do in the area?”, is a general ask. However the next sentence in the same message, “We were looking to go to a public beach or lake”, apparently has very specific intent. The message does not neatly fit into either label (specific or generic) as a whole.

Productionization — Online Serving

We productionized our framework using Bighead, a comprehensive ML infrastructure tool developed by the ML Infrastructure Team at Airbnb. The models are served through Deep Thought, the online inference component of Bighead. There will be a separate blog post introducing Bighead in more details — stay tuned!

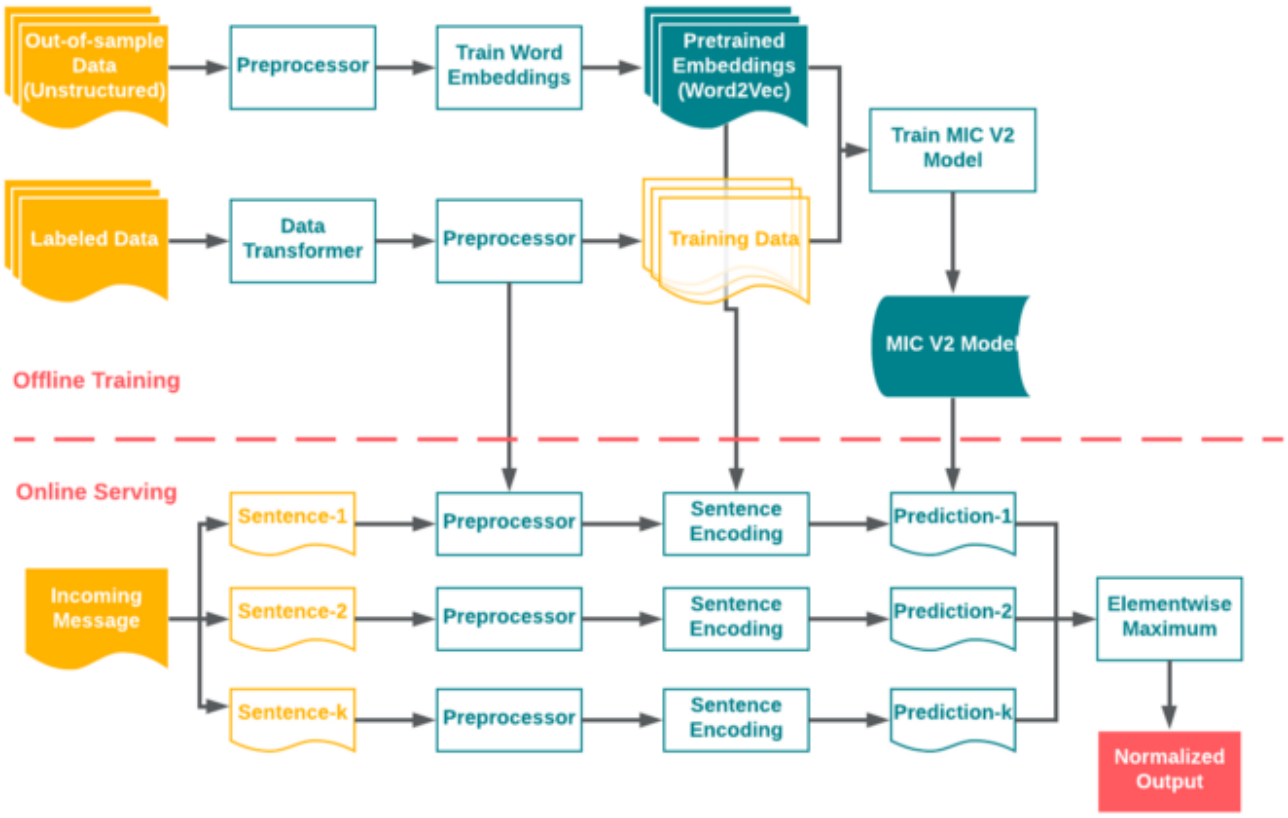


Figure 6: The offline training & online serving workflow of Phase II.

Applications

Here is a glimpse of some of the applications that are either happening or are being planned for the near future.

- Predicting customer support issues leveraging message intent history
- Guiding through cancellation / payment / refund process by identifying such intents early
- Improving booking experience by identifying guest concerns
- Providing instant smart response by identifying guest / host needs

Conclusion

We have developed a framework for message intent understanding that evolved from intent discovery to intent classification using unsupervised and supervised learning techniques. The work empowers a variety of product applications that facilitate a seamless communication experience through Airbnb's messaging platform. Below are a few takeaways:

- Unsupervised learning can be a powerful tool to provide labels for a supervised learning solution.
- Text preprocessing could play a critical role when training word embeddings using customized text corpus.
- Label quality is the key to model performance. Figuring out the right ways to reduce human error in the labeling process could have tremendous impact on your model accuracy if the bottleneck is label accuracy in your problem in the first place.

Up next, we plan to further improve our solution from several aspects:

- Improve unsupervised learning results for intent discovery by experimenting with more language representation models such as doc2vec, BERT
- Improve labeling efficiency through enhanced labeling tools with semi-supervised learning
- Improve labeling quality with more rigorous definitions and professional training

- Explore hosts' intents beyond those of the guests

As we deepen our understanding of Airbnb's text data, we are constantly identifying new areas where we can leverage this technology to improve the Airbnb product. We also plan to support other languages to assist our communities worldwide.

Acknowledgement

This work is in collaboration with *John Park* and *Sam Shadwell*. We would also like to thank *Joy Zhang, Peter Gannon, Patrick Srail, Junshuo Liao, Andrew Hoh, Darrick Brown, Atul Kale, Jeff Feng, Peggy Shao, Cindy Chen, Wei Han, Alfredo Luque, Roy Stanfield, Joshua Pekera* for their support and feedback throughout this project! We'd also like to thank *Xiaohan Zeng, Dai Li,* and *Rebecca Rosenfelt* for their kind help in proofreading!

Airbnb is always seeking outstanding people to join our team! If you are interested in working on problems such as the one in this post, please check out our open positions in Data Science and Analytics, and send your application!

Thanks to Xiaohan Zeng, Shijing Yao, Darrick Brown, Jeff Feng, and Dai.

Machine Learning NLP Data Science AI