Real-world Conversational AI for Hotel Bookings

Bai Li
University of Toronto
SnapTravel
Toronto, Canada
bai@cs.toronto.edu

Nanyi Jiang
SnapTravel
Toronto, Canada
leon@snaptravel.com

Joey Sham
SnapTravel
Toronto, Canada
joey@snaptravel.com

Henry Shi
SnapTravel
Toronto, Canada
henry@snaptravel.com

Hussein Fazal
SnapTravel
Toronto, Canada
hussein@snaptravel.com

***Abstract*—In this paper, we present a real-world conversational AI system to search for and book hotels through text messaging. Our architecture consists of a frame-based dialogue management system, which calls machine learning models for intent classification, named entity recognition, and information retrieval subtasks. Our chatbot has been deployed on a commercial scale, handling tens of thousands of hotel searches every day. We describe the various opportunities and challenges of developing a chatbot in the travel industry.**

***Index Terms*—conversational AI, task-oriented chatbot, named entity recognition, information retrieval**

I. Introduction

Task-oriented chatbots have recently been applied to many areas in e-commerce. In this paper, we describe a task-oriented chatbot system that provides hotel recommendations and deals. Users access the chatbot through third-party messaging platforms, such as Facebook Messenger (Figure 1), Amazon Alexa, and WhatsApp. The chatbot elicits information, such as travel dates and hotel preferences, through a conversation, then recommends a set of suitable hotels that the user can then book. Our system uses a dialogue manager that integrates a combination of NLP models to handle the most frequent scenarios, and defer to a human support agent for more difficult situations.

The travel industry is an excellent target for e-commerce chatbots for several reasons:

1) Typical online travel agencies provide a web interface (such as buttons, dropdowns, and checkboxes) to enter information and filter search results; this can be difficult to navigate. In contrast, chatbot have a much gentler learning curve, since users interact with the bot using natural language. Additionally, chatbots are lightweight as they are embedded in an instant messaging platform that handles authentication. All of these factors contribute to higher user convenience [1].
2) Many people book vacations using travel agents, so the idea of booking travel through conversation is already familiar. Thus, we emulate the role of a travel agent, who talks to the customer while performing searches on various supplier databases on his behalf.
3) Our chatbot has the advantage of a narrow focus, so that every conversation is related to booking a hotel. This constrains conversations to a limited set of situations, thus allowing us to develop specialized models to handle hotel-related queries with very high accuracy.

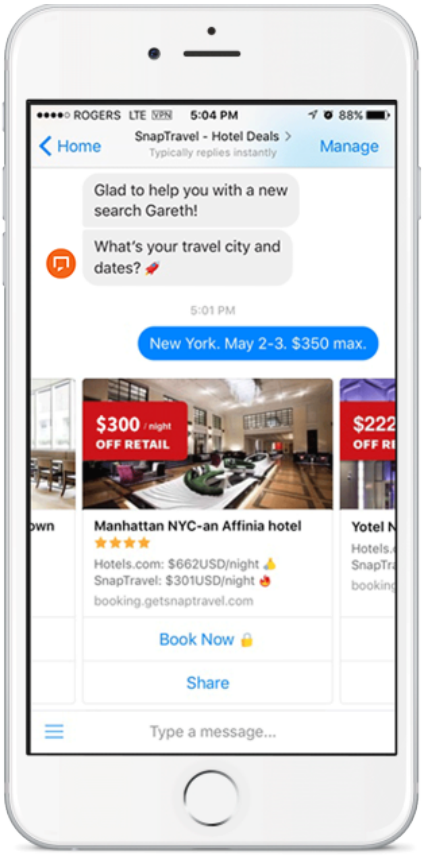

Fig. 1. Screenshot of a typical conversation with our bot in Facebook Messenger.

The automated component of the chatbot is also closely integrated with human support agents: when the NLP system is unable to understand a customer's intentions, customer support agents are notified and take over the conversation. The agents' feedback is then used to improve the AI, providing valuable training data (Figure 2). In this paper, we describe our conversational AI systems, datasets, and models.

II. Related work

Numerous task-oriented chatbots have been developed for commercial and recreational purposes. Most commercial chat-

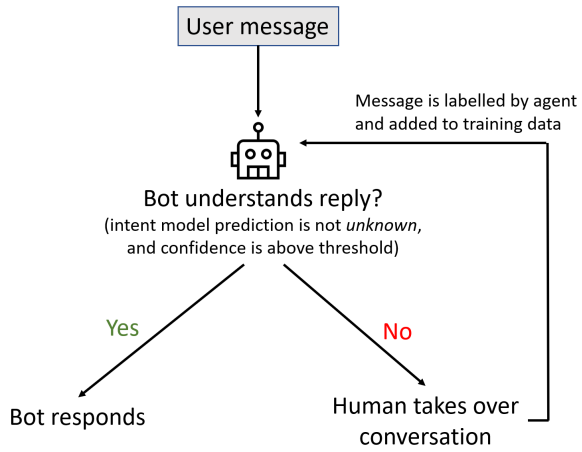

Fig. 2. The intent model determines for each incoming message, whether the bot can respond adequately. If the message cannot be recognized as one of our intent classes, then the conversation is handed to a human agent, and is added to our training data.

bots today use a frame-based dialogue system, which was first proposed in 1977 for a flight booking task [2]. Such a system uses a finite-state automaton to direct the conversation, which fills a set of slots with user-given values before an action can be taken. Modern frame-based systems often use machine learning for the slot-filling subtask [3].

Natural language processing has been applied to other problems in the travel industry, for example, text mining hotel information from user reviews for a recommendation system [4], or determining the economic importance of various hotel characteristics [5]. Sentiment analysis techniques have been applied to hotel reviews for classifying polarity [6] and identifying common complaints to report to hotel management [7].

III. Chatbot Architecture

Our chatbot system tries to find a desirable hotel for the user, through an interactive dialogue. First, the bot asks a series of questions, such as the dates of travel, the destination city, and a budget range. After the necessary information has been collected, the bot performs a search and sends a list of matching hotels, sorted based on the users' preferences; if the user is satisfied with the results, he can complete the booking within the chat client. Otherwise, the user may continue talking to the bot to further narrow down his search criteria.

At any point in the conversation, the user may request to talk to a customer support agent by clicking an "agent" or "help" button. The bot also sends the conversation to an agent if the user says something that the bot does not understand. Thus, the bot handles the most common use cases, while humans handle a long tail of specialized and less common requests.

The hotel search is backed by a database of approximately 100,000 cities and 300,000 hotels, populated using data from our partners. Each database entry contains the name of the city or hotel, geographic information (e.g., address, state, country), and various metadata (e.g., review score, number of bookings).

TABLE I
Some intent classes predicted by our model.

Intent	Description
thanks	User thanks the bot
cancel	Request to cancel booking
stop	Stop sending messages
search	Hotel search query
...	
unknown	Any other message

A. Dialogue management

Our dialog system can be described as a frame-based slot-filling system, controlled by a finite-state automaton. At each stage, the bot prompts the user to fill the next slot, but supports filling a different slot, revising a previously filled slot, or filling multiple slots at once. We use machine learning to assist with this, extracting the relevant information from natural language text (Section IV). Additionally, the system allows universal commands that can be said at any point in the conversation, such as requesting a human agent or ending the conversation.

Figure 3 shows part of the state machine, invoked when a user starts a new hotel search. Figure 4 shows a typical conversation between a user and the bot, annotated with the corresponding state transitions and calls to our machine learning models.

B. Data labelling

We collect labelled training data from two sources. First, data for the intent model is extracted from conversations between users and customer support agents. To save time, the model suggests a pre-written response to the user, which the agent either accepts by clicking a button, or composes a response from scratch. This action is logged, and after being checked by a professional annotator, is added to our training data.

Second, we employ professional annotators to create training data for each of our models, using a custom-built interface. A pool of relevant messages is selected from past user conversations; each message is annotated once and checked again by a different annotator to minimize errors. We use the PyBossa[1] framework to manage the annotation processes.

IV. Models

Our conversational AI uses machine learning for three separate, cascading tasks: intent classification, named entity recognition (NER), and information retrieval (IR). That is, the intent model is run on all messages, NER is run on only a subset of messages, and IR is run on a further subset of those. In this section, we give an overview of each task's model and evaluation metrics.

A. Intent model

The intent model processes each incoming user message and classifies it as one of several intents. The most common intents are *thanks*, *cancel*, *stop*, *search*, and *unknown* (described in

[1] https://pybossa.com

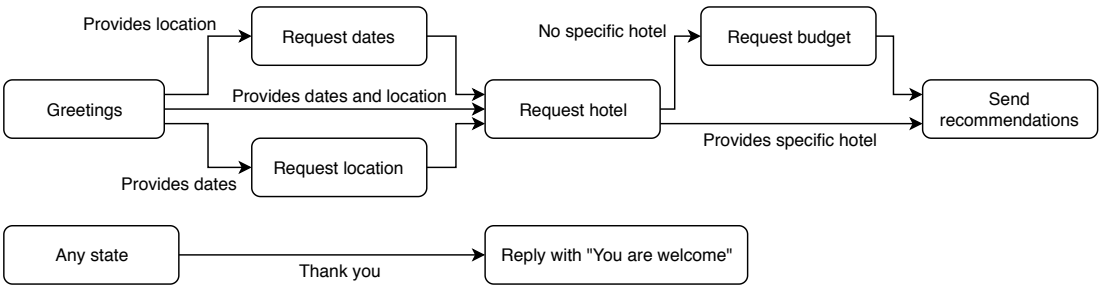

Fig. 3. Diagram showing part of the state machine, with relevant transitions; this part is invoked when a user starts a new search for a hotel.

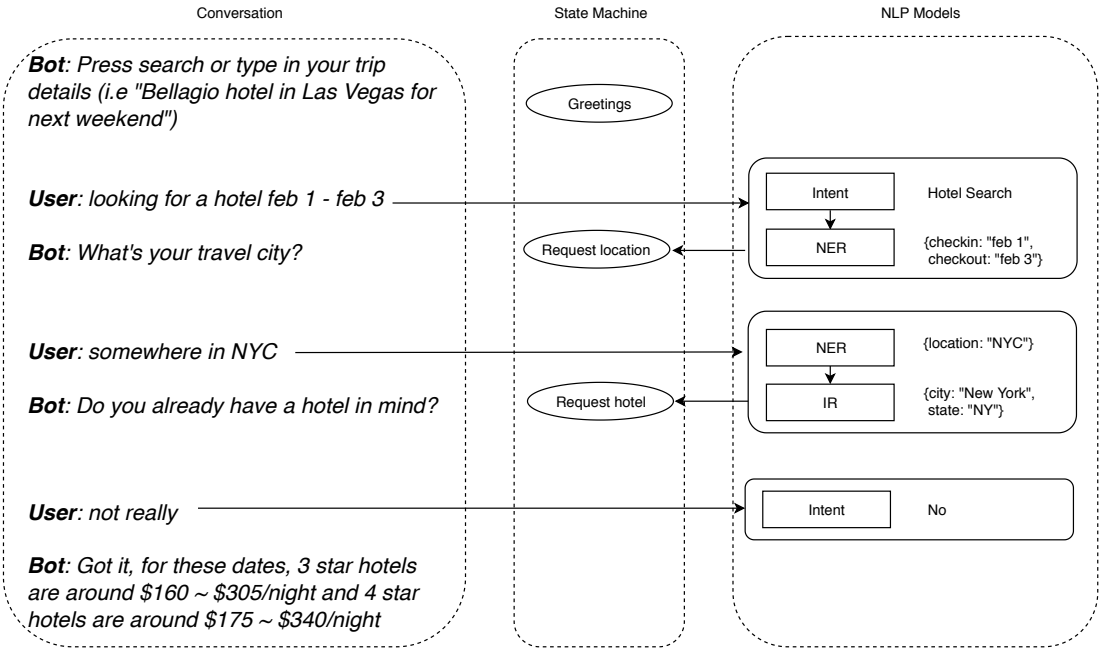

Fig. 4. Example of a conversation with our bot, with corresponding state transitions and model logic. First, the user message is processed by the intent model, which classifies the message into one of several intents (described in Table I). Depending on the intent and current conversation state, other models (NER and IR) may need to be invoked. Then, a response is generated based on output of the models, and the conversation transitions to a different state.

Table I); these intents were chosen for automation based on volume, ease of classification, and business impact. The result of the intent model is used to determine the bot's response, what further processing is necessary (in the case of *search* intent), and whether to direct the conversation to a human agent (in the case of *unknown* intent).

We use a two-stage model; the first stage is a set of keyword-matching rules that cover some unambiguous words. The second stage is a neural classification model. We use ELMo [8] to generate a sequence of 1024-dimensional embeddings from the text message; these embeddings are then processed with a bi-LSTM with 100-dimensional hidden layer. The hidden states produced by the bi-LSTM are then fed into a feedforward neural network, followed by a final softmax to generate a distribution over all possible output classes. If the confidence of the best prediction is below a threshold, then the message is classified as *unknown*. The preprocessing and training is implemented using AllenNLP [9].

TABLE II
RESULTS OF NER MODEL

Entity Type	Precision	Recall	F1
Hotel	0.84	0.53	0.65
Location	0.85	0.89	0.87
Hotel + Location	**0.94**	**0.99**	**0.96**

We evaluate our methods using per-category precision, recall, and F1 scores. These are more informative metrics than accuracy because of the class imbalance, and also because some intent classes are easier to classify than others. In particular, it is especially important to accurately classify the *search* intent, because more downstream models depend on this output.

B. *Named entity recognition*

For queries identified as *search* intent, we perform named entity recognition (NER) to extract spans from the query

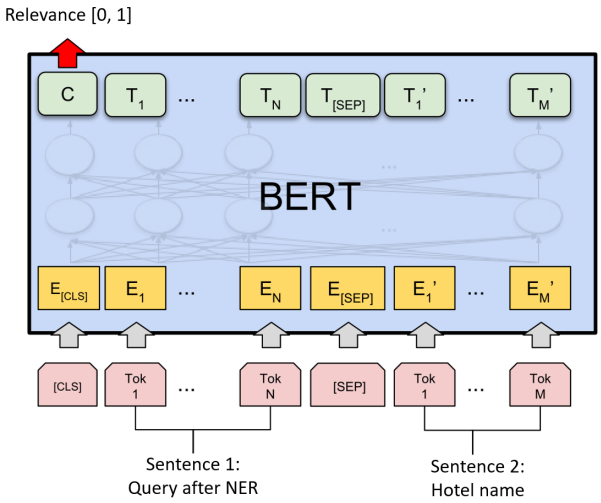

Fig. 5. BERT model for IR. The inputs are tokens for the user query (after NER) and the official hotel name, separated by a `[SEP]` token. The model learns to predict a relevance score between 0 and 1 (i.e., the pointwise approach to the learning-to-rank problem). Figure adapted from [13].

representing names of hotels and cities. Recently, neural architectures have shown to be successful for NER [10], [11]. Typically, they are trained on the CoNLL-2003 Shared Task [12] which features four entity types (persons, organizations, locations, and miscellaneous).

Our NER model instead identifies hotel and location names, for example:

- *"double room in the cosmopolitan, las vegas for Aug 11-16"*,
- *"looking for a resort in Playa del carmen near the beach"*.

We use SpaCy[2] to train custom NER models. The model initialized with SpaCy's English NER model, then fine-tuned using our data, consisting of 21K messages labelled with hotel and location entities. Our first model treats hotels and locations as separate entities, while our second model merges them and considers both hotels and locations as a single combined entity type. All models are evaluated by their precision, recall, and F1 scores for each entity type. The results are shown in Table II.

The combined NER model achieves the best accuracy, significantly better than the model with separate entity types. This is expected, since it only needs to identify entities as either hotel or location, without needing to distinguish them. The model is ineffective at differentiating between hotel and location names, likely because this is not always possible using syntactic properties alone; sometimes, world knowledge is required that is not available to the model.

C. Information retrieval

The information retrieval (IR) system takes a user search query and matches it with the best location or hotel entry in our database. It is invoked when the intent model detects a *search* intent, and the NER model recognizes a hotel or location named entity. This is a non-trivial problem because the official name of a hotel often differs significantly from what a user typically searches. For example, a user looking for the hotel *"Hyatt Regency Atlanta Downtown"* might search for *"hyatt hotel atlanta"*.

TABLE III
RESULTS OF IR MODELS

Model	Top-1 Recall	Top-3 Recall
Unigram matching baseline	0.473	–
Averaged GloVe + feedforward	0.680	0.869
BERT + fine-tuning	**0.895**	**0.961**

We first apply NER to extract the relevant parts of the query. Then, we use ElasticSearch[3] to quickly retrieve a list of potentially relevant matches from our large database of cities and hotels, using tf-idf weighted n-gram matching. Finally, we train a neural network to rank the ElasticSearch results for relevancy, given the user query and the official hotel name.

Deep learning has been applied to short text ranking, for example, using LSTMs [14], or CNN-based architectures [15], [16]. We experiment with several neural architectures, which take in the user query as one input and the hotel or city name as the second input. The model is trained to classify the match as relevant or irrelevant to the query. We compare the following models:

1) **Averaged GloVe + feedforward**: We use 100-dimensional, trainable GloVe embeddings [17] trained on Common Crawl, and produce sentence embeddings for each of the two inputs by averaging across all tokens. The sentence embeddings are then given to a feedforward neural network to predict the label.
2) **BERT + fine-tuning**: We follow the procedure for BERT sentence pair classification. That is, we feed the query as sentence A and the hotel name as sentence B into BERT, separated by a `[SEP]` token, then take the output corresponding to the `[CLS]` token into a final linear layer to predict the label. We initialize the weights with the pretrained checkpoint and fine-tune all layers for 3 epochs (Figure 5).

The models are trained on 9K search messages, with up to 10 results from ElasticSearch and annotations for which results are valid matches. Each training row is expanded into multiple message-result pairs, which are fed as instances to the network. For the BERT model, we use the uncased BERT-base, which requires significantly less memory than BERT-large. All models are trained end-to-end and implemented using AllenNLP [9].

For evaluation, the model predicts a relevance score for each entry returned by ElasticSearch, which gives a ranking of the results. Then, we evaluate the top-1 and top-3 recall: the proportion of queries for which a correct result appears

[2]https://spacy.io

[3]https://www.elastic.co

as the top-scoring match, or among the top three scoring matches, respectively. The majority of our dataset has exactly one correct match. We use these metrics because depending on the confidence score, the chatbot either sends the top match directly, or sends a set of three potential matches and asks the user to disambiguate.

We also implement a rule-based unigram matching baseline, which takes the entry with highest unigram overlap with the query string to be the top match. This model only returns the top match, so only top-1 recall is evaluated, and top-3 recall is not applicable. Both neural models outperform the baseline, but by far the best performing model is BERT with fine-tuning, which retrieves the correct match for nearly 90% of queries (Table III).

D. External validation

Each of our three models is evaluated by internal cross-validation using the metrics described above; however, the conversational AI system as a whole is validated using external metrics: agent handoff rate and booking completion rate. The agent handoff rate is the proportion of conversations that involve a customer support agent; the booking completion rate is the proportion of conversations that lead to a completed hotel booking. Both are updated on a daily basis.

External metrics serve as a proxy for our NLP system's performance, since users are more likely to request an agent and less likely to complete their booking when the bot fails. Thus, an improvement in these metrics after a model deployment validates that the model functions as intended in the real world. However, both metrics are noisy and are affected by factors unrelated to NLP, such as seasonality and changes in the hotel supply chain.

V. Conclusion

In this paper, we give an overview of our conversational AI and NLP system for hotel bookings, which is currently deployed in the real world. We describe the various machine learning models that we employ, and the unique opportunities of developing an e-commerce chatbot in the travel industry. Currently, we are building models to handle new types of queries (e.g., a hotel question-answering system), and using multi-task learning to combine our separate models. Another ongoing challenge is improving the efficiency of our models in production: since deep language models are memory-intensive, it is important to share memory across different models. We leave the detailed analysis of these systems to future work.

Our success demonstrates that our chatbot is a viable alternative to traditional mobile and web applications for commerce. Indeed, we believe that innovations in task-oriented chatbot technology will have tremendous potential to improve consumer experience and drive business growth in new and unexplored channels.

VI. Acknowledgment

We thank Frank Rudzicz for his helpful suggestions to drafts of this paper. We also thank the engineers at SnapTravel for building our chatbot: the conversational AI is just one of the many components.

References

[1] L. C. Klopfenstein, S. Delpriori, S. Malatini, and A. Bogliolo, "The rise of bots: A survey of conversational interfaces, patterns, and paradigms," in *Proceedings of the 2017 Conference on Designing Interactive Systems*. ACM, 2017, pp. 555–565.

[2] D. G. Bobrow, R. M. Kaplan, M. Kay, D. A. Norman, H. Thompson, and T. Winograd, "Gus, a frame-driven dialog system," *Artificial intelligence*, vol. 8, no. 2, pp. 155–173, 1977.

[3] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu *et al.*, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 530–539, 2015.

[4] K. Zhang, K. Wang, X. Wang, C. Jin, and A. Zhou, "Hotel recommendation based on user preference analysis," in *2015 31st IEEE International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 2015, pp. 134–138.

[5] A. Ghose, P. G. Ipeirotis, and B. Li, "Designing ranking systems for hotels on travel search engines by mining user-generated and crowdsourced content," *Marketing Science*, vol. 31, no. 3, pp. 493–520, 2012.

[6] H.-X. Shi and X.-J. Li, "A sentiment analysis model for hotel reviews based on supervised learning," in *2011 International Conference on Machine Learning and Cybernetics*, vol. 3. IEEE, 2011, pp. 950–954.

[7] W. Kasper and M. Vela, "Sentiment analysis for hotel reviews," in *Computational linguistics-applications conference*, vol. 231527, 2011, pp. 45–52.

[8] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, vol. 1, 2018, pp. 2227–2237.

[9] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer, "AllenNLP: A deep semantic natural language processing platform," in *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, 2018, pp. 1–6.

[10] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proceedings of NAACL-HLT*, 2016, pp. 260–270.

[11] M. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semi-supervised sequence tagging with bidirectional language models," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2017, pp. 1756–1765.

[12] E. F. T. K. Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, 2003.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[14] D. Wang and E. Nyberg, "A long short-term memory model for answer sentence selection in question answering," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, vol. 2, 2015, pp. 707–712.

[15] A. Severyn and A. Moschitti, "Learning to rank short text pairs with convolutional deep neural networks," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. ACM, 2015, pp. 373–382.

[16] H. He, K. Gimpel, and J. Lin, "Multi-perspective sentence similarity modeling with convolutional neural networks," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1576–1586.

[17] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.