NLP approach to writing a booking service

Vladislav Dordiuk
Ekaterinburg, Russia
vladislav0860@gmail.com

Abstract

This report contains an overview on NLP approaches to create a booking service.

I. Introduction

The integration of Natural Language Processing (NLP) techniques has long been crucial in reshaping user-computer interactions within various domains. In the late 1960s, early NLP systems, known as Natural Language Interfaces to Databases (NLIDB), revolutionized user interactions with databases [1]. Systems like LUNAR [2], PLANES [3], PRECISE [4], WASP [5], and NALIX [6] and others emerged as pioneers, simplifying access to databases by enabling users to communicate with them directly in natural language, without the need to learn complex query syntax for each database. These systems operated by interpreting user inputs and translating them into database queries, utilizing methods such as pattern-matching, syntax analysis, and semantic understanding.

The advance of hardware capabilities and the availability of open datasets over time changed the used NLP methodologies as the researchers began leveraging machine learning approaches in NLP. This evolution opened new possibilities for the integration of advanced NLP systems into e-commerce, particularly manifesting as AI-driven chatbots. Today, these NLP-powered chatbots are heavily used in e-commerce, simplifying customer interactions. They function as real-time customer service assistants, engaging with users, providing product information, aiding in purchases, and swiftly resolving inquiries and complaints. The 24/7 availability of these AI-driven systems has increased their popularity among businesses, ensuring seamless customer support and significantly enhancing user experience [7], [8].

These modern systems have moved beyond their origins in database interactions, now employing advanced machine learning algorithms. These algorithms learn from user interactions, continuously improving their understanding of language nuances and user intent [9]. Consequently, they offer personalized and context-aware interactions, enhancing user satisfaction and contributing to the growth of e-commerce platforms.

This work is focused on creating an AI-powered booking service that would provide an interface for automated flight reservation by combining the ideas from [10], [11], [12] and [9] as they solve similar tasks.

II. Methods

A. Architecture

As can be seen in the papers [11], [12], AI-powered chat systems that use databases usually utilize several ML models to solve different tasks. Namely, such tasks include intent recognition (IR) and named entity recognition (NER). Besides the NLP models, such systems implement specific scenarios that activate depending on user's responses. In this work, we implement a custom architecture inspired by [11], where such system is used in combination with large database for hotel booking.

Our custom architecture is shown in Fig. 1. It operates on a straightforward principle: all inputs undergo classification by an IR model into four classes: intent to book, off-topic request, expression of gratitude, or intention to end the conversation. If the input is tagged as an intent to book, it then proceeds through the NER model and a subsequent post-processing pipeline. Three other classes are not involved in booking, however, they are needed to handle different user's inputs.

B. Data

In this work, we used a total of six different datasets. The first dataset was generated with proprietary third-party software and consists of 120 labelled messages with intent to book a flight. It was used to test NER models. It includes 20 direct requests, 20 neutral requests, 20 casual requests, 20 very casual requests in lowercase, 20 formal requests, and 20 requests with missing data, which is labelled as "Unspecified". Each request has four key points that NER model must retrieve, they are: user's name, city of departure, city of destination and date of flight.

All other datasets were combined to train the IR model for the four label classification task. They include Airline Travel Information System (ATIS) dataset [13] with flight booking requests, Ubuntu Dialogue Corpus [14] and DailyDialog [15] with casual conversations, and two generated datasets, one with expressions of gratitude, the other with the intent to end conversation.

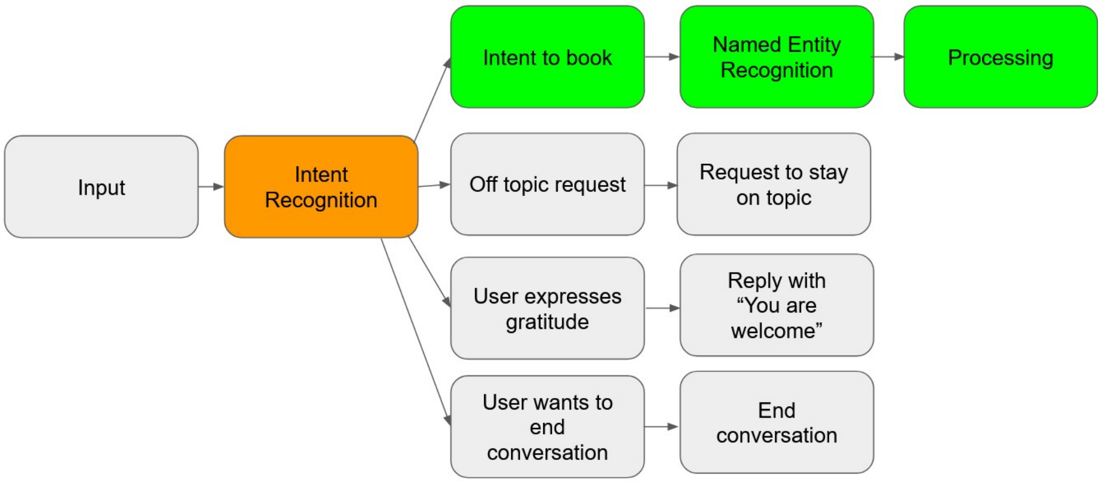

Fig. 1: The logic behind the architecture. The orange block shows IR model that classifies all input information. Green blocks show the processing for booking a ticket.

C. *Models*

In this work there were tested a total 8 different NER approaches. The first one is Regular Expression (RegEx) based model. It doesn't use any machine learning methods and serves as a baseline. Unlike machine learning-based NER models that learn patterns from data, regex-based approaches use explicit predefined rules to recognize entities based on specific patterns or sequences of characters. Other models include pretrained models from HuggingFace [16] python library: the base BERT model, cased and uncased [17], RoBERTa [18] and Electra [19]. Three other models that were used are SpaCy large pipeline [20], SpaCy transformers pipeline [20] and FLERT [21] from FLAIR library. Since some selected NER models are not able to detect dates, the datefinder python library was used to substitute for this function. To all the used models we added date standardization, so it shouldn't matter if the user has typed "12th September 2023" or "September 12, 2023", as all the dates are parsed and transformed to a standard format dd-mm-yyyy.

For the Intent Recognition task we've chosen the pretrained RoBERTa model from HuggingFace, which shows slightly better results [22] than the BERT model, that was used in [11]. The chosen model was fine-tuned during 10 epochs with the aforementioned data using standard HuggingFace sequence classification pipeline. The fine-tuning was performed on a local machine with a single NVIDIA RTX 3060 GPU, which limited the batch size to 32. During the training, AdamW was used as an optimizer with cross entropy as loss function.

III. Results

A. *NER models*

TABLE I: Accuracy of NER models on different subsets of data and the whole dataset. The best results are in bold.

Models	I	II	III	IV	V	VI	Whole dataset
RegEx Baseline	65.0	0.0	60.0	0.0	0.0	35.0	26.66
BERT Base	100.0	100.0	100.0	0.0	95.0	95.0	81.66
BERT uncased	100.0	100.0	100.0	60.0	95.0	95.0	91.66
RoBERTa	100.0	100.0	100.0	60.0	95.0	100.0	92.50
Electra	100.0	100.0	100.0	55.0	95.0	100.0	91.66
SpaCy lg	90.0	10.0	90.0	0.0	55.0	70.0	52.5
SpaCy trf	100.0	90.0	100.0	0.0	50.0	100.0	73.33
FLERT	**100.0**	**100.0**	**100.0**	**95.0**	**100.0**	**100.0**	**99.16**

During the experiments, we've tested 8 different NER models. Table I shows accuracy of each model on all 6 subsets of data, mentioned above. Here we counted as correct predictions only those requests, where all the required key points were correctly identified. As can be seen, the main difficulty for the models was the 4th subset, which contains casual requests in lower case. As the tests have shown, almost all models struggled identifying names of the cities of departure and destination that are not capitalized and marked them with "Unspecified" label.

Table II shows a different perspective on the NER results. As we can see, BERT base, BERT uncased, RoBERTa and Electra have the same accuracy in date recognition, since none of these models can recognize dates natively, and they use datefinder

TABLE II: Accuracy of NER models on different labels from the test dataset. The best results are in bold.

Models	Name	Departure	Destination	Date
RegEx Baseline	72.50	71.66	69.16	48.33
BERT Base	83.33	83.33	82.50	97.50
BERT uncased	100.00	95.00	83.33	97.50
RoBERTa	100.00	95.00	100.00	97.50
Electra	98.33	95.00	100.00	97.50
SpaCy lg	80.00	83.33	83.33	60.00
SpaCy trf	83.33	83.33	83.33	85.83
FLERT	**100.00**	**99.16**	**100.00**	**100.00**

python library for this task. What is notable is that even though datefinder approach is pattern-based, it has shown higher accuracy than SpaCy models, that were trained to extract dates from texts. The overall best results were shown by FLERT model with FLAIR library pipeline, and it was chosen as NER model in the final pipeline. It could successfully recognize most of the labels and only made mistakes while identifying departure city from lowercase casual request.

B. IR model fine-tuning

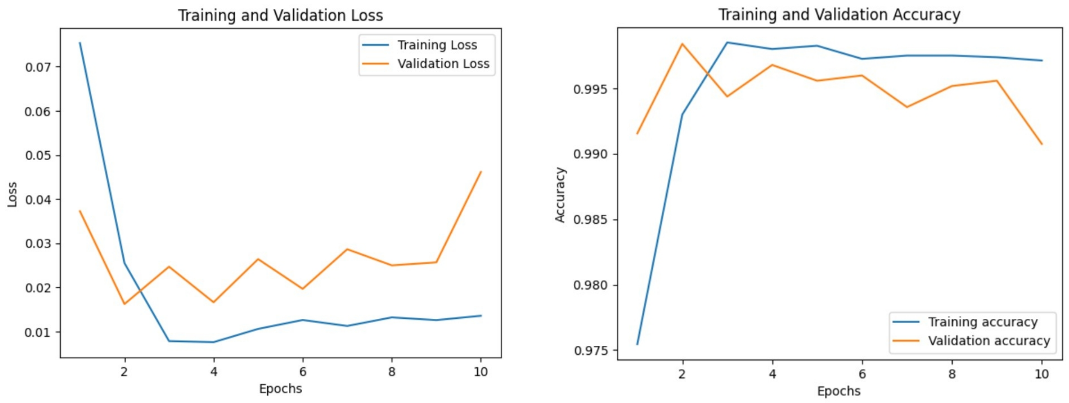

Fig. 2: Loss and accuracy values over 10 fine-tuning epochs.

As it was mentioned before, the model was trained on 32 batch due to the technical limitations. This led to the fact that the weights of the model's head were updating too frequently, and the overfitting could start to show up after the first or second epoch (see fig. 2. Deeper inspection of the first epoch values have shown that the model does, indeed, converge after only a few batches (see fig. 3). The change of learning rate didn't lead to positive results, as the model would either stop converging or generalizing. In order to somehow counter this, the saving of the best model was implemented, meaning that in the final pipeline we use the state of the model that had the lowest loss value, instead of the state from the last epoch.

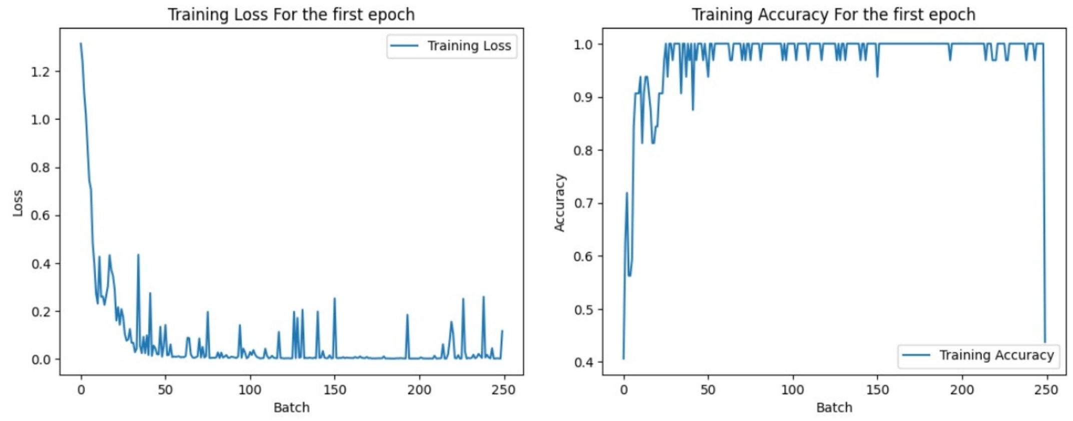

Fig. 3: Loss and accuracy values per batch over the first fine-tuning epoch.

TABLE III: Metrics of IR model performance on different labels from the test dataset.

Intent	Precision	Recall	f1-score	N of instances
Booking	1.0	1.0	1.0	632
Miscellaneous	1.0	1.0	1.0	1454
Gratitude	1.0	0.99	1.0	213
End_conversation	1.0	0.99	1.0	186

The table III shows high performance of the fine-tuned model, as all the metrics show very positive results, which is confirmed by the confusion matrix in fig 4. The fine-tuned model have achieved nearly 100% accuracy on the whole test dataset, and there are only a few misclassifications of miscellaneous label, intent to express gratitude, and intent to end conversation. However, such big values might be caused by different factors. For example, the similar patterns that are in the data (e.g. “I want to fly to Boston” and “I want to fly to London”). Also, since two of five datasets (intent of gratitude and intent to end conversation) were automatically generated, the data leak is also possible, however it shouldn’t affect the booking intent and miscellaneous class.

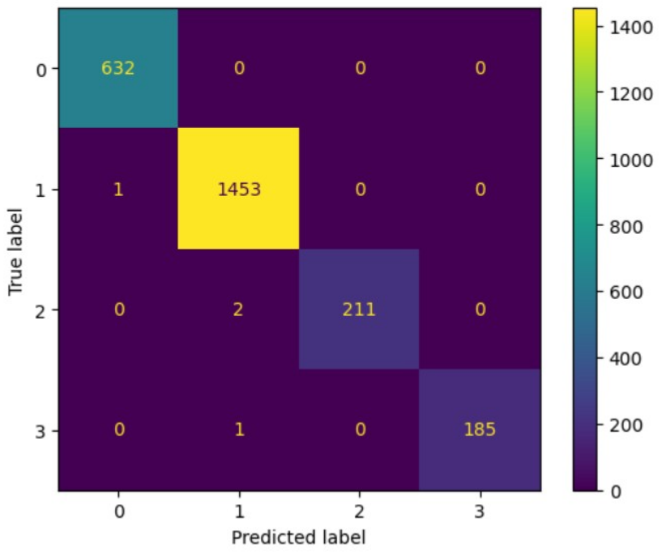

Fig. 4: Confusion matrix. Numbers 0-3 represent labels, which are booking intent, miscellaneous label, intent to express gratitude, and intent to end conversation respectively.

C. *Final architecture*

For the final architecture of the booking service, a custom pipeline that would incorporate the best performing NER model and fine-tuned IR model was created. The main idea was shown in fig. 1, and fig. 5 shows the processing of the inputs that were labelled by IR model as booking intent. First, the input is processed by NER model that tries to extract user’s name, city of departure, city of destination and the date of flight from the input. If none of those are identified, the model asks the user to include these data points in their request. However, in order to start selecting a flight, the city of departure and city of destination is already enough. In this case, the model would list the dates of matching flights, the most appropriate of which is selected by the user. If the user has already specified the date in the original request, this step is performed automatically. After finding the matching flight, the model would ask for user’s name in order to make reservation. After all the required data is collected, the model asks the user to check if everything is in order and if the user confirms, the model finishes booking the flight.

IV. Discussion

This project provides a custom pipeline as a solution for a problem of automated flight booking. The architecture is based on real-world existing apps for e-commerce, such as hotel booking and ticketing. The proposed app uses IR and NER machine learning modes for information retrieval in combination with post-processing methods such as date standardization and scripted data structuring.

The train and test data for this project were assembled from parts of the existing datasets and generated data that were manually labelled. The robustness of the proposed method can me further increased by expanding the data for fine-tuning the

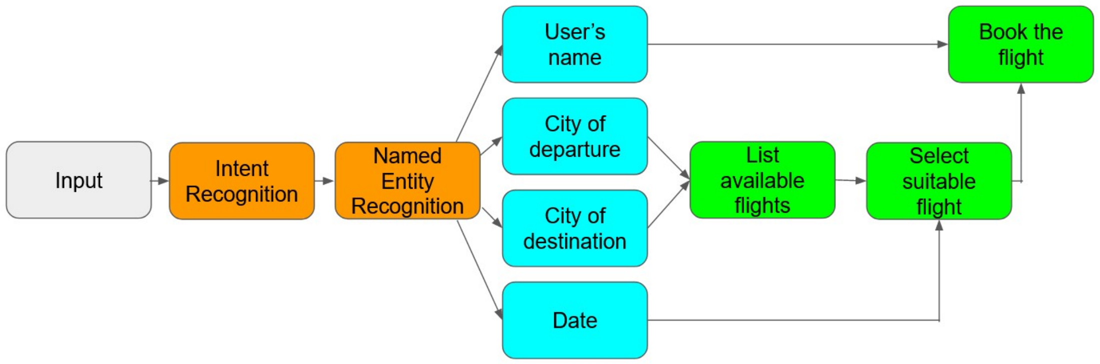

Fig. 5: Data processing for the inputs with intent of booking. ML models are marked with orange, significant values are marked with light-blue, and functions are marked with green.

IR model and adding more diversity to it. Also, the user experience can be improved by adding more labels to IR model and if possible implementing state of the art conversational models. This would make human-bot interaction much more pleasant

References

[1] M. Mony, J. M. Rao, and M. M. Potey, "An overview of nlidb approaches and implementation for airline reservation system," *International Journal of Computer Applications*, vol. 107, no. 5, 2014.
[2] W. A. Woods, "Progress in natural language understanding: an application to lunar geology," in *Proceedings of the June 4-8, 1973, national computer conference and exposition*, 1973, pp. 441–450.
[3] D. L. Waltz, "An english language question answering system for a large relational database," *Communications of the ACM*, vol. 21, no. 7, pp. 526–539, 1978.
[4] A.-M. Popescu, A. Armanasu, O. Etzioni, D. Ko, and A. Yates, "Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability," in *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, 2004, pp. 141–147.
[5] Y. W. Wong, *Learning for semantic parsing using statistical machine translation techniques*. Computer Science Department, University of Texas at Austin, 2005.
[6] Y. Li, H. Yang, and H. Jagadish, "Constructing a generic natural language interface for an xml database," in *International Conference on Extending Database Technology*. Springer, 2006, pp. 737–754.
[7] S. Shah, H. Ghomeshi, E. Vakaj, E. Cooper, and S. Fouad, "A review of natural language processing in contact centre automation," *Pattern Analysis and Applications*, vol. 26, no. 3, pp. 823–846, 2023.
[8] M. Cristian, S. Christian, and T. Dumitru-Tudor, "A study in the automation of service ticket recognition using natural language processing," in *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 2019, pp. 1–6.
[9] B. Setiaji and F. W. Wibowo, "Chatbot using a knowledge in database: human-to-machine conversation modeling," in *2016 7th international conference on intelligent systems, modelling and simulation (ISMS)*. IEEE, 2016, pp. 72–77.
[10] M. S. Wandhare, M. Gaikwad, S. Dighe, Y. Jadhav, and S. Karwa, "Artificial conversation entity-a review."
[11] E. Handoyo, M. Arfan, Y. A. A. Soetrisno, M. Somantri, A. Sofwan, and E. W. Sinuraya, "Ticketing chatbot service using serverless nlp technology," in *2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*. IEEE, 2018, pp. 325–330.
[12] e. a. Sumeet Pate, "Human-to-machine conversation modeling for movie ticket booking using nlp," in *International Journal for Research in Engineering Application Management (IJREAM)*. IJREAM, 2018, pp. 11–14.
[13] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington, "The atis spoken language systems pilot corpus," in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.
[14] R. Lowe, N. Pow, I. Serban, and J. Pineau, "The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems," *arXiv preprint arXiv:1506.08909*, 2015.
[15] Y. Li, H. Su, X. Shen, W. Li, Z. Cao, and S. Niu, "Dailydialog: A manually labelled multi-turn dialogue dataset," *arXiv preprint arXiv:1710.03957*, 2017.
[16] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.
[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[18] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.
[19] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.
[20] Explosion AI, "spaCy 3: Industrial-strength natural language processing in Python," https://spacy.io.
[21] S. Schweter and A. Akbik, "Flert: Document-level features for named entity recognition," *arXiv preprint arXiv:2011.06993*, 2020.
[22] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, "Beyond accuracy: Behavioral testing of nlp models with checklist," *arXiv preprint arXiv:2005.04118*, 2020.