BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

TravelX: NLP model for a real-world task-oriented intelligent assistant

– ITSG report –

Team members
Carausu Catrinel Ioana
Morariu Bogdan
Software Engineering, group 258

2019

Abstract

Natural Language Processing (NLP) has always been a challenging task in Artificial Intelligence, gaining increasing popularity due to new models emerging in the literature, that either solve new areas of the problem, or provide better solutions for algorithms already consecrated.

One such recently born model is Googleś Bert (Bidirectional Encoder Representations from Transformers), that has created excitement in the field, by providing state-of-the-art results for a various number of NLP tasks.

In this paper we proposed a real-life use case that demonstrates the use and performance of the Bert model in Intent Classification, Slot Filling and Question Answering. Furthermore, these three models have been integrated to create an intelligent assistant able to suggest accommodations for tourists, based on some stated criteria. Once an accommodation has been suggested, our system provides capabilities to answer the user's questions about the given property. We used transfer learning to adapt and fine-tune pre-trained Bert models to our custom datasets, and were able to obtain results of 92.59% for intent classification and slot filling, and of 90.14% for question answering.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 What? Why? How?

Intelligent assistants, especially in the form of task-oriented chatbots, have gained popularity in many areas of application, but especially in e-commerce. Messaging platforms enhanced by conversational AI systems provide fast response to common user questions, or automate certain business processes (refunds, orders, setting appointments).

We propose the integration of a task-based chatbot in a business flow specific of the travel industry, namely appointing and booking a hotel, when creating a travel plan.

We are addressing this problem by transposing it into the following NLP specific tasks: intent classification, slot filling and question answering. We intend to use the Bert [2] model as primary building block for our intelligent system. The core task of our conversational AI will be to request some key information from the user, such as the location and check-in date for the accommodation, in order to be able to perform a search for the most fitting properties. Once a property is found, the user will be able to request more data about it, data that will be looked up in the hotel's description. Another capability of out bot will also be to entertain simple conversation with the user (greet, thank).

The problem of chatbots and conversational AI has been addressed extensively in the literature, together with some possible applications. Our goal is to employ a rather new NLP model to build our intelligent assistant, and to study the applicability of this model for our three tasks: intent classification, slot filling and question answering. The results we obtained are relatively close to the state-of-the-art.

1.2 Paper structure and original contributions

The research presented in this paper advances the theory, design, and implementation of several particular models.

The main contribution of this report is to present an intelligent algorithm for solving the problem of intent classification, slot filling and question answering in the context of an intelligent assistant.

The second contribution of this report consists of building an intuitive, easy-to-use and user friendly software application. Our aim is to build an algorithm that will help tourists find accomodations for the destinations where they intend to travel to. This is done in a user-friendly manner, supporting also some basic elements of casual conversation.

The third contribution of this thesis consists of comparing the performance obtained by our custom Bert model in the slot filling task, with respect to the state-of-the-art results provided by industrial-level libraries such as SpaCy [1]. We also examined the benefits of creating a separate model for intent classification and one for slot filling, versus one single model that solves both tasks simultaneously.

The present work contains xyz bibliographical references and is structured in five chapters as follows.

The second chapter contains our problem definition and describes the input and output data of our system.

The third chapter presents the related work and state-of-the-art results that our model is based upon.

Chapter 4 details the algorithm design in much more detail. It presents basic concepts about the functioning of the BERT model and how we adapted it for our purposes. The used technologies are also stated and described briefly.

Chapter 5 states the experimental results obtained, focusing first on the data set and data pre-processing, then on the evaluation techniques used and finally on a comparison between our models.

Finally, chapter 6 presents our conclusions and future work, where we acknowledge how our model can be improved and extended, and we talk about the possible integrations with chatbot smart devices like Amazon Alexa.

Chapter 2

Scientific Problem

2.1 Problem definition

The goal of our chatbot system is to automate the process of finding suitable accommodation for a vacation, by trying to accede in the best possible way to the user's needs. We defined a set of mandatory features that the user must provide in order for the system to start a hotel search.

Our system tries to fill in the values for these features by processing the text responses given by the user, in order to identify if any necessary research criteria has been provided in the current reply. If the user does not provide it beforehand, the bot will be able to ask specifically for the missing pieces of information required for the search. The search is performed using a custom hotel API, the results being sorted by relevance and rating. Once a hotel has been found, the user can address further questions in order to obtain information about the accommodation.

As benefits and motivation for the automation of the hotel booking process, the first that comes to mind is the fact that, in spite of the numerous websites that provide booking capabilities, many customers still prefer to go personally to a travel agency to request more information about the accommodations. This means the preferred form of knowledge gathering is through conversation. A chatbot provides a familiar way of communicating and a familiar interface for the user.

By having an intelligent assistant, many of the FAQs that would keep a human consultant busy, can be automatically answered, in order to speed up the booking process and to employ the consultant's time when it is really necessary. Task based chatbots operate in a closed domain, which means the conversation flows can be abstracted to a limited number of scenarios. This allows for great accuracy of the intelligent agents.

As disadvantages, it is worth mentioning that the difference between conversing with a human interlocutor versus a chatbot can still be very apparent at times, and that can be frustrating for some users. Machines are still having a hard time understanding all nuances of our natural language, and using certain subtle expressions or phraseology can be misleading for the AI, that can provide erroneous or implausible answers.

We define the input and output sets for our application in the following sections.

2.2 Input for the intent classifier and slot filling component

The text message provided by the user through the UI, in natural language.

2.3 Output of the intent classifier and slot filling component

Output of the intent classifier and slot filling component The classifier provides a dictionary of key-value pairs containing the feature(s) name(s) and their corresponding value(s), extracted from the current user reply. These features are then processed by a state machine that will produce a more user friendly response (text response) to be sent back to the user.

2.4 Input for the question answering component

The question provided by the user through the user interface (UI), in natural language, and the hotel description retrieved through the API search (in natural language).

2.5 Output for the question answering component

A fragment of text extracted from the hotel description that answers the question best (has obtained the best confidence score).

Chapter 3

State of art/Related work

Real-world Conversational AI for Hotel Bookings [6] is a commercialized system that uses various intelligent models to recommend a hotel to the user. It communicates with the user through messages and it can be integrated with modern tools like messengers and AI assistants. It uses three Natural Language Processing models to handle most tasks and the chat-bot can switch to a human support agent if the user requests it. The data set used was split into two databases. The first database contained information of 9000 hotels the developers had partnered with.The second database of the system was created by experienced travel agents from conversations with the customers. The conversations were then analysed and labeled by the same agents to make them machine learning suitable. The system uses three natural language processing models in its architecture, for the process of recommending a hotel to the user. The steps of the state machine are the following: identify the intent of the user, extract information from the user messages about his search, ask for more information if needed, search for the top 3 hotels in the database. The flow is split into 3 separate, sequential machine learning tasks: Intent Classification, Named Entity Recognition (NER) and Information Retrieval (IR). For the Information Retrieval task three algorithms were tested: Uni-gram machine for baseline, Averaged GloVe + feed forward and BERT + fine-tuning. After training BERT with 9k hotel descriptions, they obtained a accuracy of 96% on the task of query search based on 2 parameters, overtaking the second solution by about 10%. The system is currently deployed in the real world as a e-commerce chat-bot for the travel industry. To track the chatbots value the distributors use 2 metrics: Number of successful bookings and number of human agent requests. The future work of the paper includes memory allocation optimizations and the addition of a fourth intelligent system for question answering on top of the current flow of the state machine.

The idea of using BERT for Natural Language Understanding tasks is presented in [1]. They compare the performance of BERT on NLU tasks against currently state of the art architectures:

attention-based recurrent neural network models and slot-gated models. The model they proposed is a joint BERT model for intent classification and slot filling. Other joint NLU models approaches include CNNCRF [11], RecNN [4], joint RNN-LSTM [5], attention-based BiRNN [7], and slot-gated attention based model [3]. The performance of the models tested on two public benchmark datasets: ATIS flight reservation dataset [9] and Snips personal voice assistant dataset [2]. The BERT model was pre trained using the datasets from the uncased-BERT baseline. The final accuracy of the system on the ATIS dataset was 97.9% for intent classification and 96.0% for slot filling and on the Snips 98.6% on intent classification and 97.0% on the slot filling task.

Chapter 4

Proposed approach

4.1 Algorithm description

4.1.1 BERT-based model

The transformer architecture [10] replaces well established concepts of recurrent neural networks or convolutional neural networks, in favor of an attention based system.

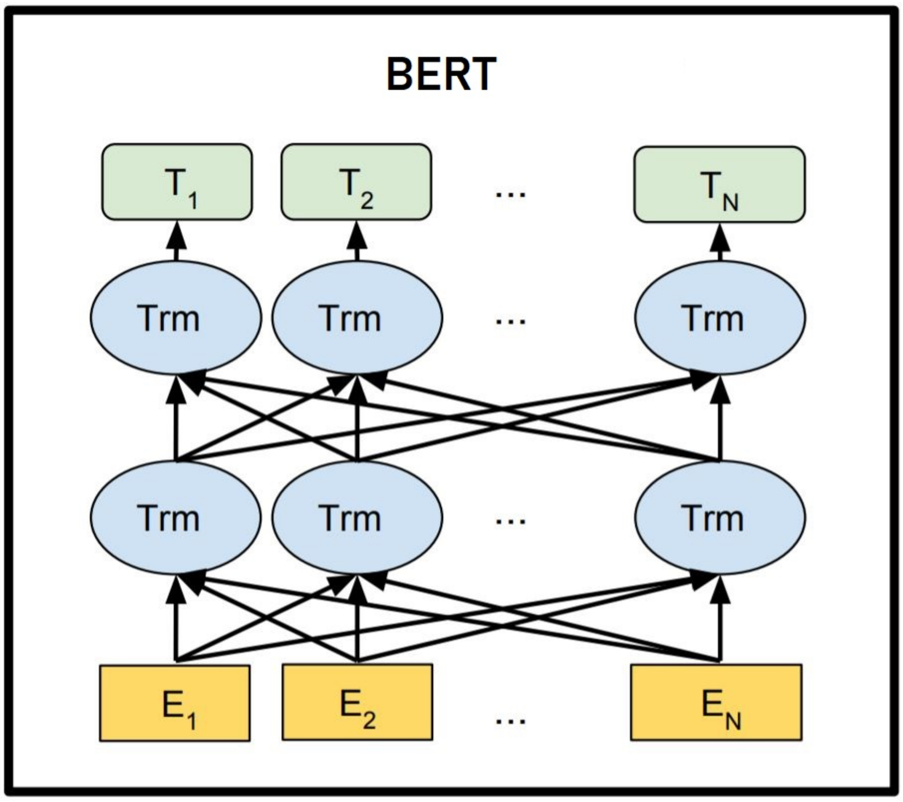
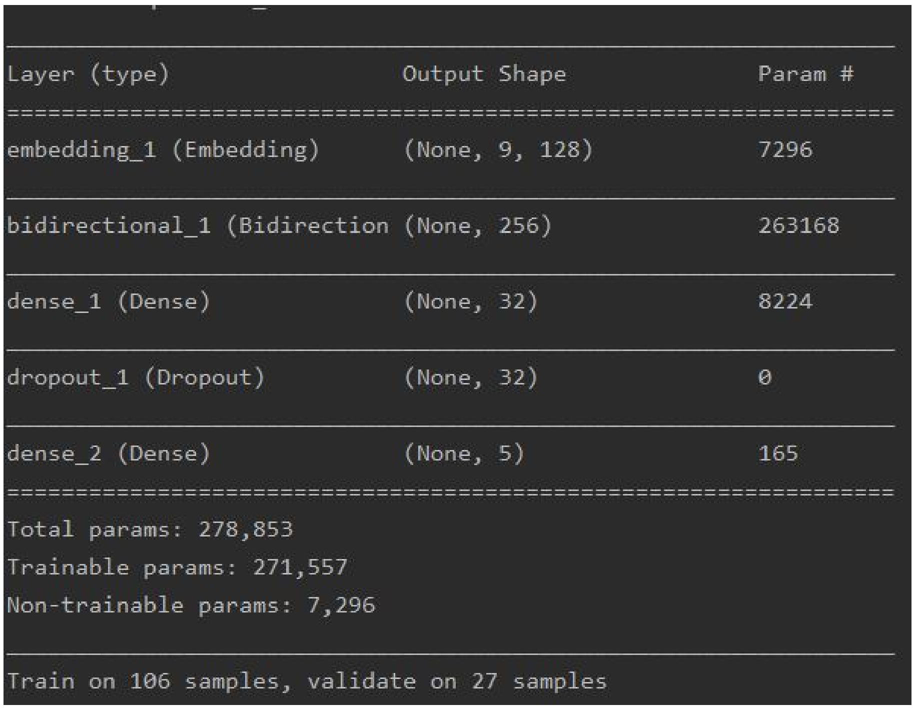


Figure 4.1: BERT model architecture



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 9, 128)            7296
_________________________________________________________________
bidirectional_1 (Bidirection (None, 256)               263168
_________________________________________________________________
dense_1 (Dense)              (None, 32)                8224
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 165
=================================================================
Total params: 278,853
Trainable params: 271,557
Non-trainable params: 7,296
_________________________________________________________________
Train on 106 samples, validate on 27 samples
```

Figure 4.2: Custom intent classifier model structure

The model uses the encoder decoder pattern combined with a self-attention system to solve common natural language processing tasks. Both the encoder and decoder consist of stacks of 6 hidden layers with similar architecture, separated in sub-layers. The encoder contains a multi-head attention sub-layer and a simple feed-forward layer. The two sub-layers are connected through residual connections followed by a layer normalization. The decoder works similar to the encoder with the addition of an second attention layers, which takes its input from the output of the encoder.

Transformers solve the parallelization incompatibility of current state-of-the-art systems on top of increasing the quality of the output. A transformer system reached a new single-model state-of-the-art BLEU score on the WMT 2014 English-to-German translation task, with a small cost of training compared to the previous best model.

BERT (Bidirectional Encoder Representations from Transformers) is a machine learning architecture, that presents state-of-the-art results on eleven natural language processing tasks [8]. BERT is a pretrained model which can then be optimized for downstream tasks. The implementation of downstream tasks can be done through fine-tuning or feature-based. Feature based approach implies architecture changes to accommodate for additional weight categories. The fine-tunig approach, that BERT uses, adds an additional final layer to accommodate the weights for a specific set of data. The pretraining of the model is done in two steps: masked language model and next sentence prediction. MLM was needed because of the bidirectional nature and full connectivity of the model, without which

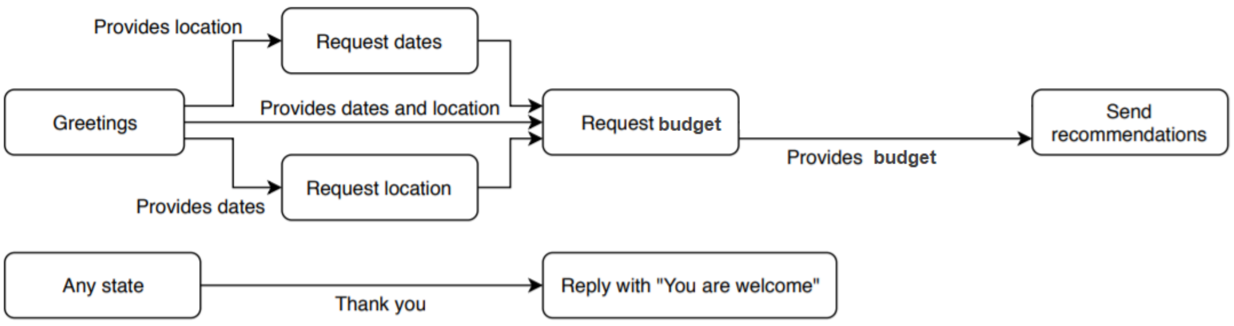


Figure 4.3: Diagram showing part of the state machine, with relevant transitions; this part is invoked when a user starts a new conversation with the bot

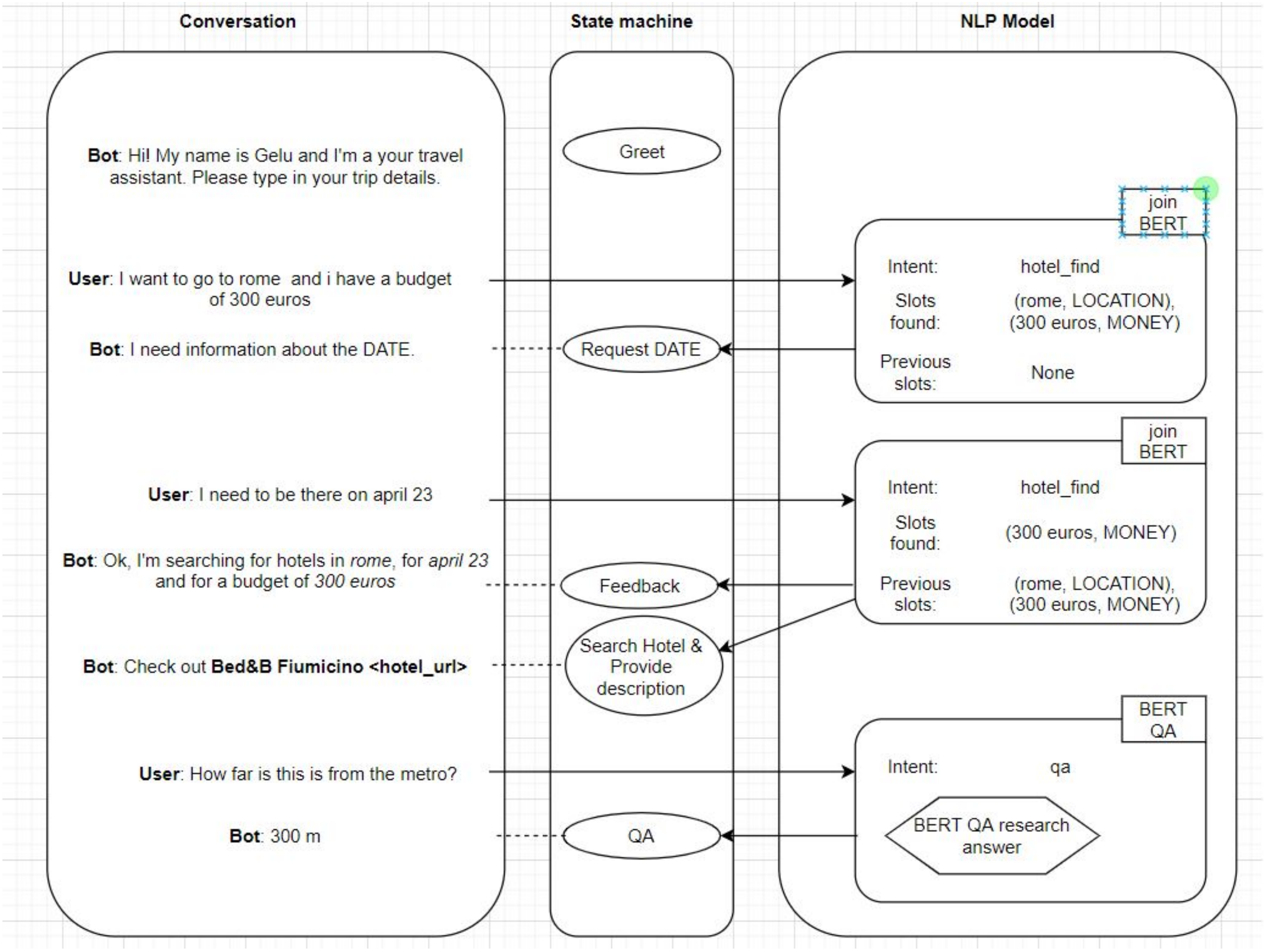

Conversation
State machine
NLP Model
Bot: Hi! My name is Gelu and I'm a your travel assistant. Please type in your trip details.
User: I want to go to rome and i have a budget of 300 euros
Bot: I need information about the DATE.
User: I need to be there on april 23
Bot: Ok, I'm searching for hotels in rome, for april 23 and for a budget of 300 euros
Bot: Check out Bed&B Fiumicino <hotel_url>
User: How far is this is from the metro?
Bot: 300 m
Greet
Request DATE
Feedback
Search Hotel & Provide description
QA
join BERT
Intent: hotel_find
Slots found: (rome, LOCATION), (300 euros, MONEY)
Previous slots: None
join BERT
Intent: hotel_find
Slots found: (300 euros, MONEY)
Previous slots: (rome, LOCATION), (300 euros, MONEY)
BERT QA
Intent: qa
BERT QA research answer

Figure 4.4: 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. Depending on the intent and current conversation state, the BERT QA model may need to be invoked. Then, a response is generated based on output of the models, and the conversation transitions to a different state.

nodes could find their word in the output of the previous nodes. Next sentence prediction proved to be very useful in natural language understanding tasks like question answering.

4.1.2 Custom intent classification model

The current model configuration of the intent classifier was achieved through experimentation. Although not being the best one obtained during the development phase, we had to simplify the model for performance reasons.

The model architecture we settled to is composed of an embedding, a bi-directional and a dense layer with ReLu activation. We included dropout to avoid overfitting and a final dense layer with Softmax that produces the confidence values for each class. We can observe the network structure in Figure 4.2.

4.1.3 Chatbot module

The chatbot module is built as a state machine responsible of keeping track of the mandatory hotel features needed to perform the query over the hotels API.

In the initial state of the bot, it is the first to initiate the conversation by greeting the user and encourage him/her to provide search criteria for the trip.

Once the user responds, the intent of the message is computed, along with potential slots (features) contained. The features detected are persisted in memory and the conversation proceeds until all three key features are obtained. If the user does not provide the search parameters, the bot will request them successively. An example of this state transitions is provided in Figure 4.3, and an example of a chatbot conversation, along with the internal states is provided in Figure 4.4.

When all the search criteria is filled in, the bot will fire a request to a hotel searching API, similar to Booking.com, that will retrieve the description of the hotel that best matched the search, sorted by the user rating. The relevance sorting was not addressed in this scope, but is a point of improvement for future work.

Having provided a the hotel suggestion, the user can address questions to the bot, and the answers will be looked up in the hotel description.

4.2 Tools and technologies used

Among the frameworks and tools we used for the design and development of our system, we enlist the most important ones:

- **Scikit-learn**: python machine learning library;
- **Pytorch**: open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing;
- **Tensorflow**: Pyhton library for building machine learning models;
- **Keras**: Pyhton library that facilitates building neural networks;
- **SciPy**: a Python-based ecosystem of open-source software for mathematics, science, and engineering that has a very mature NLP module;
- **React**: Javascript framework used for developing our user interface;
- **Electron**: allows for the development of desktop GUI applications using web technologies. We used it to be able to run our chatbot both as a web and desktop application;
- **Matplotlib**: provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits
- **Flask**: web framework for building APIs in Python

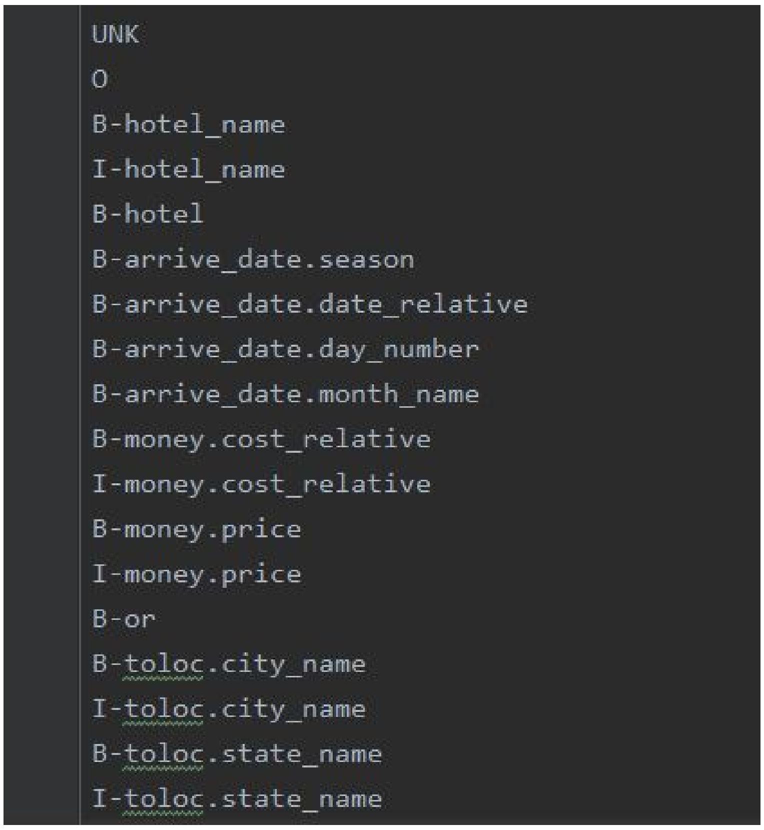
4.3 Feature definition and extraction

We defined five intent classes as follows:

- greet
- thank
- hotel_find
- stop
- assist.

The hotel_find intent can contain a set of 19 features, as listed in Figure 4.5.

Each sentence produced by the user is analyzed from the intent point of view and slots contained. Each prediction retrieves the intent detected and a dictionary containing the pairs of slot names and slot values found inside the phrase, if any.



```
UNK
O
B-hotel_name
I-hotel_name
B-hotel
B-arrive_date.season
B-arrive_date.date_relative
B-arrive_date.day_number
B-arrive_date.month_name
B-money.cost_relative
I-money.cost_relative
B-money.price
I-money.price
B-or
B-toloc.city_name
I-toloc.city_name
B-toloc.state_name
I-toloc.state_name
```

Figure 4.5: Slot classification features

4.4 Improvements and optimisations

The first version of the application had a very big response-time for the predictions and API calls. We performed a memory footprint reduction task, aiming to decrease resource utilization of both the models and of the state machine, by reducing variable and object allocation and decreasing the network depth. Response time improvements:

3.8 seconds avg. -> 1.1 seconds avg.

(request completion-time measured in Chrome dev tools).

Chapter 5

Application (numerical validation)

We present in this chapter the results obtained on the custom intent classifier and as well as on the BERT-based model (joined intent and slot filling), and a comparison with state-of-the-art results.

5.1 Methodology

- As our intent classification and slot filling tasks are essentially classification problems, for evaluation we use precision, recall and F1 scores, together with validation loss. For the question answering task we measure the F1 score and the exact matches percentage.
- We aim to get better insight on the advantages of BERT versus a simple classifier in the context of intent classification, as well as to test its performance on smaller and real world data.

5.2 Data

As mentioned in the previous section, we performed transfer-learning and fine-tuning on pre-trained BERT models, that were evaluated on state-of-the-art datasets. For the intent classification and slot filling task, we used a pre-trained model and trained it on the ATIS [9] and Snips [2] datasets. For the question answering task we used a pre-trained BERT model on SQuAD 2.0 []. We then performed transfer-learning and trained these models on custom data sets that we created.

The dataset was one of our main issues since we did not find an open-source one that satisfied our requirements. We managed to gather a sample on 100 data entries and to model them for our training and evaluation tasks, but this small amount of training data was one of the reasons why our model did not reach better accuracy.

5.2.1 Data preparation

When dealing with natural language, data can be very complex and unorganised. As such, simplifying as best as possible the dataset is an essential endeavour in order to reach good accuracy. For both our intent classification and slot filling tasks, we applied a series of data pre-processing techniques.

- **data cleaning**: the raw data obtained from the future user needs to be cleared of all the factors that could create noise. As such, we eliminated punctuation and special characters and tokenized the sentences into words. We then lowercased all words;
- **lemmatization**: this process ensures that the words are more valuable and more easily recognisable, by removing any trace of inflections from a word. An example of such a process is the transformation of plural forms into singular.
- **data splitting**: we divided out data into training (80%) and evaluation (20%) sets, to obtain actual results of how the model performs on unfamiliar data.
- **data integration**: in our joined BERT approach, we composed an extensive dataset combining both information about the intents and information about the slots that need to be filled in each phrase.

5.3 Results

Comparing the results obtained by our two approaches, the custom intent classifier with SpaCy slot filling, and the joined BERT we observe that while our custom intent classifier did perform better on intent classification alone, it obtained poorer results when integrated with the slot filling agent.

We defined four intent classes for our custom intent classifier. The results obtained after 100 epoch are presented in Figures 5.1 and 5.2.

We obtained an accuracy of 87% for the custom intent classifier and of 92.59% for the Joined BERT model. The BERT model behaved better in a real-life usage simulation of the application.

As a further analysis, we computed the confidence intervals for the accuracy of our two approaches. The computations follow Equation 5.1. The results of the comparison are shown in Table 5.1, computed on validation datasets of 30 (n=30) samples with p1=0.87 and p2=0.9259.

Confidence intervals are a way of quantifying the uncertainty of an estimate, in our case, the accuracy, by providing upper and lower bounds for that parameter, as well as likelihood. Generally speaking, we can assert that smaller confidence intervals denote a more precise estimate, while larger confidence intervals show a less precise estimate.

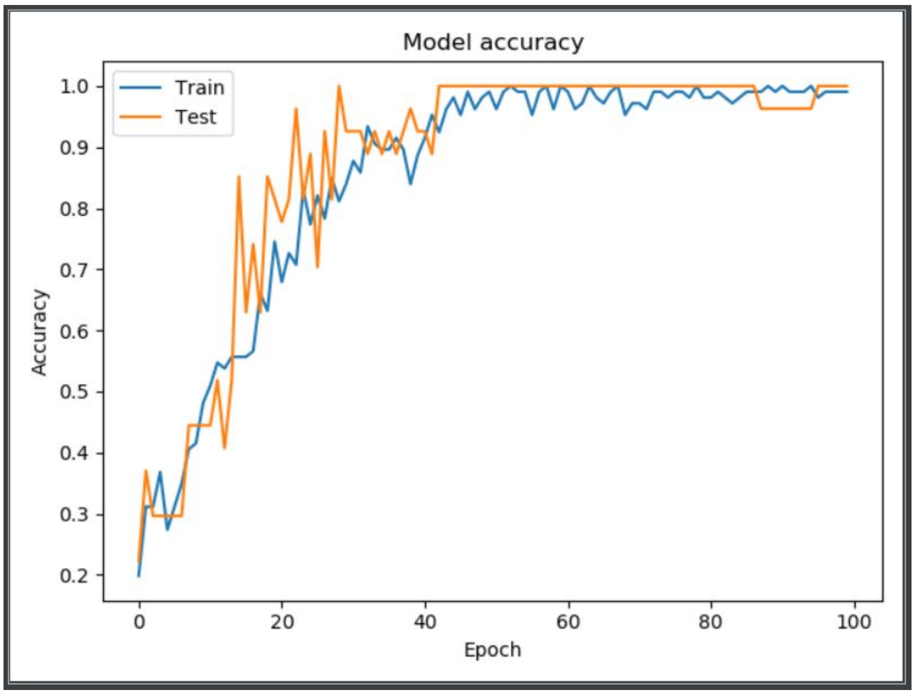


Figure 5.1: Custom intent classifier accuracy after 100 epochs

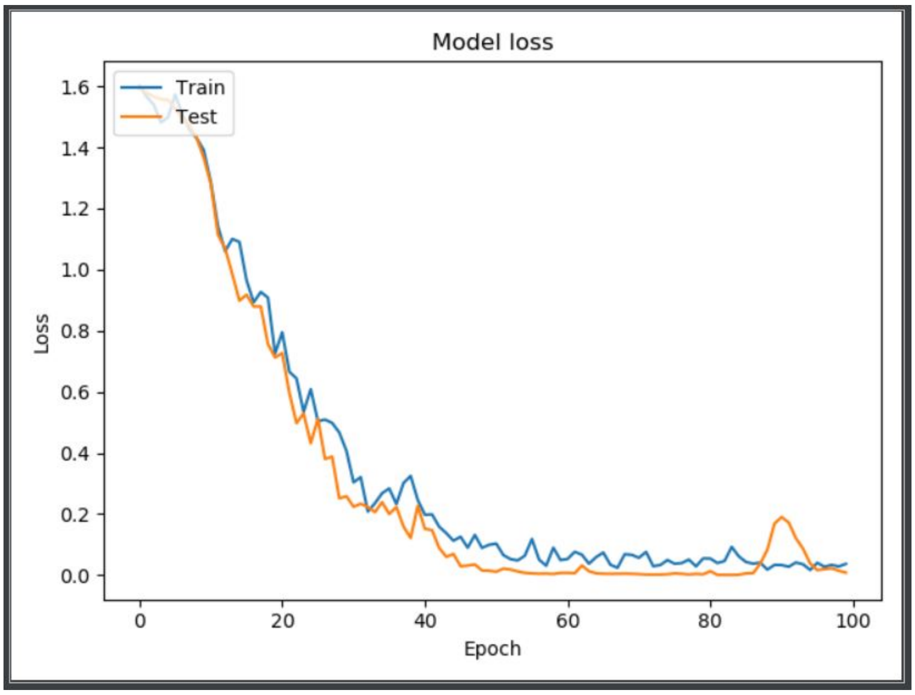


Figure 5.2: Custom intent classifier validation loss after 100 epochs

$$P \in [p - z(p(1-p)/n)^{1/2}, p + z(p(1-p)/n)^{1/2}] \tag{5.1}$$

Confidence intervals				
P = 1 - α	z	Interval bi-LSTM (I1)	Interval Bert (I2)	I1 $\cap$ I2
99.9%	3.3	[0.5281, 1.0]	[0.7665, 1.0]	[0.7665, 1.0]
99.0%	2.577	[0.5856, 0.9744]	[0.8028, 1.0]	[0.8028, 0.9744]
98.5%	2.43	[0.5962, 0.9638]	[0.8095, 1.0]	[0.8095, 0.9638]
97.5%	2.243	[0.6104, 0.9496]	[0.8185, 1.0]	[0.8185, 0.9496]
95.0%	1.96	[0.6318, 0.9282]	[0.832, 1.0]	[0.832, 0.9282]
90.0%	1.645	[0.6556, 0.9044]	[0.8471, 1.00]	[0.8471, 0.9044]
85.0%	1.439	[0.6712, 0.8888]	[0.8569, 0.9947]	[0.8569, 0.888]
75.0%	1.151	[0.6929, 0.8671]	[0.8707, 0.9809]	$\emptyset$

Table 5.1: Confidence intervals for bi-LSTM and Bert models, and their comparison.

Following the study of these confidence intervals, we can affirm with highest probability of 75.0% that the Bert model performs better for our problem and case study. Furthermore, there is a 95% likelihood that the confidence interval [0.832, 1.0] covers the true classification accuracy of the Bert model on unseen data.

A visual representation of the confidence intervals is depicted in Figure 5.3. It is noticeable that the smaller confidence intervals for Bert also denote that the algorithm produces more precise estimations.

5.4 Back-end Python application

The back-end application comprehends both the AI modules, the chatbot itself (state-machine) and the Web API written in Flask. We illustrate the main architecture and relations between the application models below.

5.5 User interface

As far as our user interface is concerned, we created a Front-End application that connects to our Back-End in python. The application was written in React and encapsulated in the Electron framework in order for it to be used as both a web and desktop application.

The interface is a simplistic one, as we believe minimalism is an important factor towards user friendliness. It presents a chat window with an input box and a button for sending the messages. Animations are available for when the chatbot is processing ("thinking") the response, and the window is scrolled automatically when new messages (that would exceed the viewport) are added.

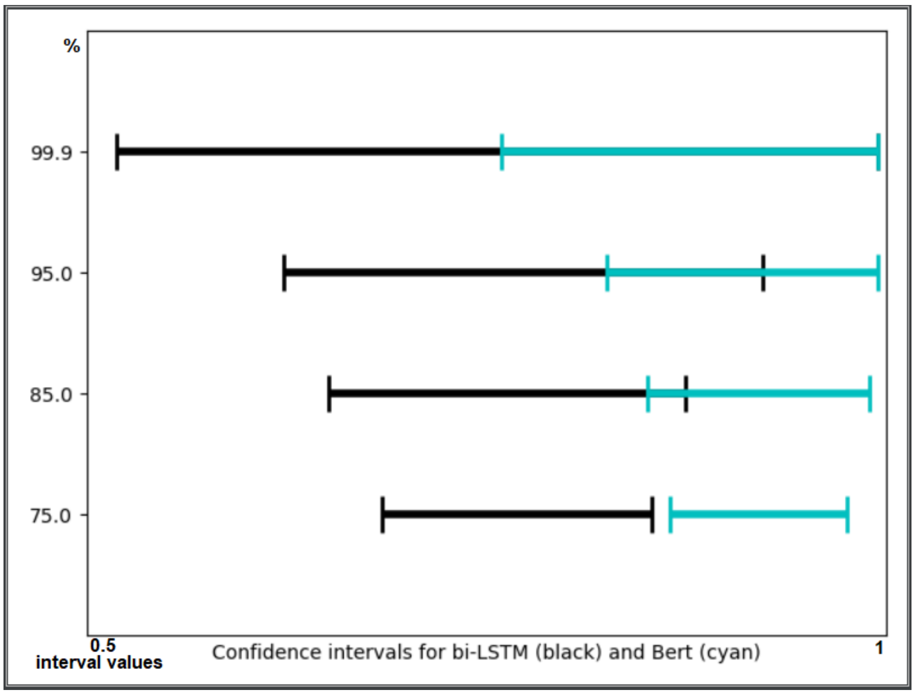


Figure 5.3: Confidence intervals for bi-LSTM and Bert. We notice that, with 75% probability, Bert performs better for the task at hand. The smaller intervals for Bert also denote that the algorithm produces more precise estimations.

5.6 Discussion

Our two approaches, the custom intent classifier with SpaCy slot filling, and the joined BERT presented in 4.1, have unraveled interesting findings. While our custom intent classifier did perform better on intent classification alone, it obtained poorer results when combined with the slot filling task. Correlating the two datasets into one, that was fed to the joined BERT model has reached the best accuracy, being able to make more insightful connections between the intent and the slots.

We did not reach state of the art results, mostly due to our lack of data. We would like to employ more time to extend the datasets as well as adding more intents and slots to see how well the two models scale.

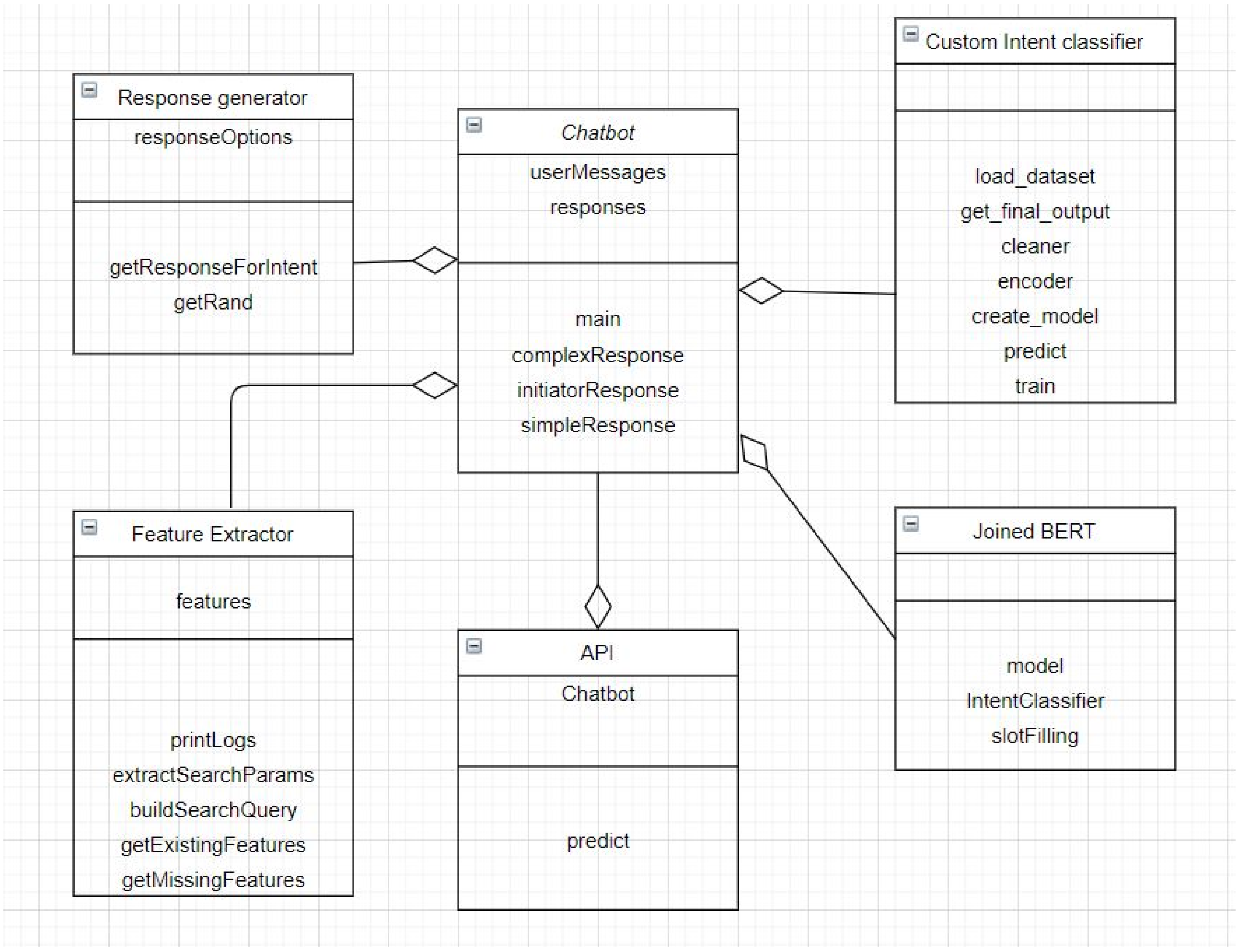


Figure 5.4: The Back-end application architecture

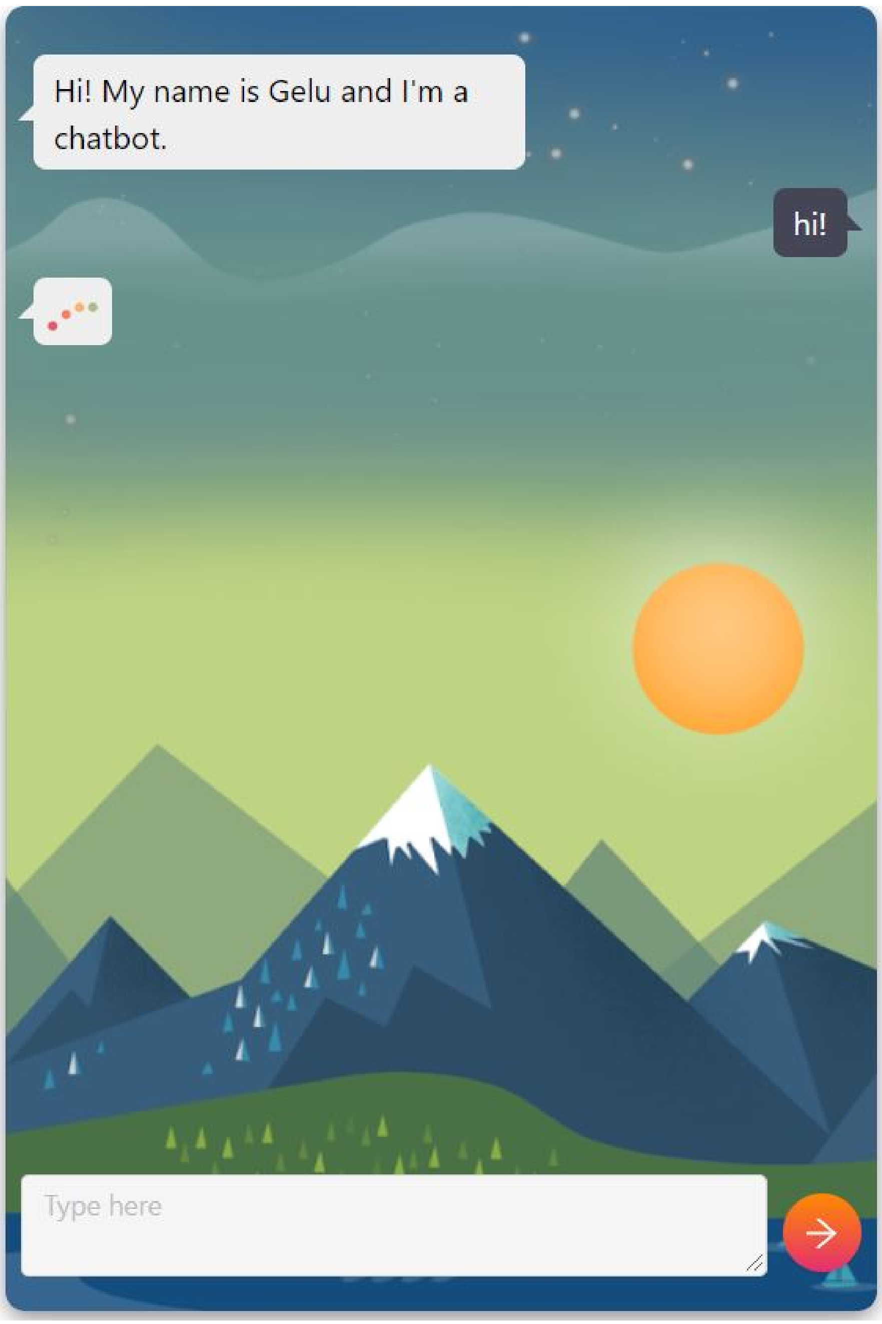


Figure 5.5: The application user interface

Chapter 6

Conclusion and future work

In this paper we have demonstrated the applications of the BERT model for solving several NLP tasks: intent classification, slot filling and question answering.

We analyzed a custom built classifier using a bi-LSTM model against the BERT based implementations and connected them through a state-machine application with another intelligent agent able to perform question answering on the result obtained from the first model. Results were positive for both models, but did not reach state-of-the-art accuracy.

Furthermore, one of the goals we intended to achieve through this paper was to demonstrate the use of such models in a real-world application context, namely by building a task-oriented chatbot that helps users find accommodations for their vacations. The proof of concept illustrated within this research demonstrates that an intelligent assistant can be successfully used for a practical use case and is fairly user-friendly.

One of the shortcomings that impacted our results was the lack of a comprehensive hotels dataset, such that we had to create our own dataset and have a limited volume of training data.

As future work, we would like to improve the accuracy of the current model, by gaining more insightful data. Moreover, adding new intents and slots for other key-features relative to the booking domain would increase the user-friendliness, operability and usefulness of the assistant. Adding more intents would also support more complex dialog flows and provide a more authentic conversation experience.

Another feature that could be added to our system is the capability of the bot to understand when there is the need to request assistance from a specialised operator (human). Then the bot could send the history of the conversation to the operator that will take over the conversation. This feature could be somewhat facilitated by the fact that our system already preserves the chat history locally.

Last but not least, we propose an integration with chatbot smart devices like Amazon Echo (Alexa) that could combine both written and spoken (voice) input.

Bibliography

[1] Qian Chen, Zhu Zhuo, and Wen Wang. Bert for joint intent classification and slot filling. Technical report, Speech Lab, DAMO Academy, Alibaba Group, February 2019.

[2] Alice Coucke, Alaa Saade, Adrien Ball, Theodore Bluche, Alexandre Caulier, David Leroy, Clement Doumour, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Mael Primet, and Joseph Dureau. Snips voice platform: an embedded spoken language understanding system for privateby-design voice interfaces. Technical report, 2018.

[3] Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and YunNung Chen. Slot-gated modeling for joint slot filling and intent prediction. *In NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2*, pages 735–737, June 2018.

[4] Daniel Guo, Gokhan T ur, Wen tau Yih, and Geoffrey Zweig. Joint semantic utterance classification and slot filling with recursive neural networks. *In 2014 IEEE Spoken Language Technology Workshop, SLT 2014, South Lake Tahoe, NV, USA*, pages 554–563, December 2014.

[5] Dilek Hakkani-Tur, G okhan T ur, Asli CÂ¸ elikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and YeYi Wang. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. *In Interspeech 2016, San Francisco, CA, USA*, pages 715–719, September 2016.

[6] Bai L, Nanyi Jiang, Joey Sham, Henry Shi, and Hussein Fazal. Real-world conversational ai for hotel bookings. Technical report, August 2019.

[7] Bing Liu and Ian Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. *In Interspeech 2016, San Francisco,CA, USA*, pages 685–688, September 2016.

[8] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. Technical report, Google AI Language, May 2019.

[9] Gokhan T ur, Dilek Hakkani-T ur, and Larry P. Heck. What is left to be understood in atis? *In 2010 IEEE Spoken Language Technology Workshop, SLT 2010, Berkeley, California, USA*, pages 19–23, December 2010.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[11] Puyang Xu and Ruhi Sarikaya. Convolutional neural network based triangular crf for joint intent detection and slot filling. *In 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic*, pages 78–83, December 2013.