Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 717–725
Marseille, 11–16 May 2020

MTSI-BERT: A Session-aware Knowledge-based Conversational Agent

Matteo A. Senese, Giuseppe Rizzo, Mauro Dragoni, Maurizio Morisio
LINKS Foundation, Fondazione Bruno Kessler, Politecnico di Torino
matteo.senese@linksfoundation.com, giuseppe.rizzo@linksfoundation.com, dragoni@fbk.eu, maurizio.morisio@polito.it

Abstract

In the last years, the state of the art of NLP research has made a huge step forward. Since the release of ELMo (Peters et al., 2018), a new race for the leading scoreboards of all the main linguistic tasks has begun. Several models have been published achieving promising results in all the major NLP applications, from question answering to text classification, passing through named entity recognition. These great research discoveries coincide with an increasing trend for voice-based technologies in the customer care market. One of the next biggest challenges in this scenario will be the handling of multi-turn conversations, a type of conversations that differs from single-turn by the presence of multiple related interactions. The proposed work is an attempt to exploit one of these new milestones to handle multi-turn conversations. **MTSI-BERT** is a BERT-based model achieving promising results in intent classification, knowledge base action prediction and end of dialogue session detection, to determine the right moment to fulfill the user request. The study about the realization of PuffBot, an intelligent chatbot to support and monitor people suffering from asthma, shows how this type of technique could be an important piece in the development of future chatbots.

Keywords: Multi-turn, Conversational AI, Goal-Oriented, BERT

1. Introduction

Conversational AI is the field of Artificial Intelligence that investigates the techniques to allow machines to interact with us using natural language conversations. Such a task is among the most challenging of the entire NLP field and also one of the hottest topics in nowadays customer care market. The possibility to automate customer care process using an intelligent agent that can understand the user request, assist him and help him to fulfill the final goal is a great desire of companies all around the world. Nevertheless, Conversational AI can also be exploited in medical scenario to help doctors in monitoring the status of their patients and avoid the latter to make long lines at the hospital for trivial sicknesses that do not require the expert intervention. Today the big tech players such as Google and Amazon have already deployed conversational agents capable to assist the user in the fulfillment of daily tasks through the use of voice (Tulshan and Dhage, 2019). Such agents can schedule appointments, control IoT devices in a room, play music and perform searches on the Web. This type of agents simulates what a typical assistant can do, thus taking the name of *virtual assistants*. Virtual assistants like the *Google Assistant* and the *Amazon Alexa* are today widely used in everyday life in an imperative setup, namely a person asks once and the assistant replies and acts right after ("turn on the lights in the kitchen"). Thanks to the increasing demand of IoT devices, this type of assistants are nowadays present in the house of million of customers. They exploit an Automatic Speech Recognition (ASR) module that is able, via Speech-to-Text (STT) technique, to translate speech audio signal to written text. Such written text is then processed and analyzed by a Natural Language Understanding (NLU) module responsible for extracting useful information from text in order to understand what a user asks. All the information are then interpreted and passed to a service manager that dispatches a call to a specific API, in order to perform the desired action. The results are then presented to the user in natural language through a Text-to-Speech (TTS) module that is today able to provide answers in an almost human-like voice.
The total virtual assistants market size is estimated to be USD 2.39 billion in 2018 and it is expected to expand of 40.4% over the forecast period (Graham and Jones, 2016). While the modern technologies seem to push toward a voice-first scenario, a lot of problems remain still unsolved. One limitation resides in multi-turn conversation: a conversational paradigm composed by multiple related interactions between the user and the agent. Most of the times the agents are unable to understand a user question that refers to a previous utterance, thus making impossible to exchange multiple utterances on the same argument. Additionally there is not a mechanism that could allow these agents to lead the conversation whenever is needed (e.g. missing information for the goal achievement). One of the key challenge of a multi-turn technology is the definition and exploitation of the conversational context. This context is whole or part of the conversational history needed to resolve the current user utterance. Correctly defining the context window makes possible to correctly understand the current user utterance by solving pending references. In recent works, we have seen that the majority of models works on a fixed-length context (Adiwardana et al., 2020; Sordoni et al., 2015), which could miss some important pieces of conversation, or on a potentially infinite conversational history like with memory networks (Sukhbaatar et al., 2015). We believe that a reference to a potential infinite context is unnatural (humans rely mainly on short memory during a conversation), introduces noise for the classification and is not scalable for long dialogues (the number of computations depends on the size of the memory). In this paper, we propose a modeling for multi-turn that casts the entire dialogue in sub-conversations having particular features. The contributions of this paper are:

1. The proposal of a modeling for a multi-turn goal-oriented conversation based on single-intent sessions that allows the creation of a dynamic length context. Thus can be integrated in multi-turn systems that need to refer to past utterances. With a dynamic length con-

text the reference is performed only on a finite set of utterances highlighted by the algorithm.

2. The development of a classifier based on BERT(Devlin et al., 2018) to identify and classify each conversational session.

3. The presentation of a CONVersational ontOLOGY to store long term information (extra-session) for the whole dialogue.

The remainder of this paper is structured as follows: in Section 2. we report previous research studies that were inspirational for this work. In Section 3. we detail the modeling of the conversation and our developed classifier and in Section 4. we report the results of the classifier when tested with KVRET dataset (Eric and Manning, 2017). In Section 5. we detail the application of such a methodology in a real case scenario for the classifier utilization and, in Section 6. we conclude the paper with discussions and future work.

2. Related Work

The language model is one of the fundamental tasks of NLP and it aims to predict the next word in a sentence given all (or part of) the preceding ones. While since 2018 the preferred way to build such models was the use of RNN (Yao et al., 2013), recently numerous approaches based on Transformer (Vaswani et al., 2017) have been largely used due to their ability to better capture long-term dependencies. One of the modern use of language modeling task is to allow transformer-based architectures to learn a contextual representations for the embedding (Peters et al., 2018). Such representation can be used in almost all NLP tasks by enabling a way of doing transfer learning similar to computer vision field. In the last years, a lot of new contextual embedding corpora have been released achieving surprising results in various major NLP tasks such as question answering and named entity recognition. A great breakthrough in NLP was the presentation of the BERT model (Devlin et al., 2018) that for the first time proposes a bidirectional transformer-based masked language model. The use of pre-trained BERT is today almost a must for a lot of NLP works and has inspired a lot of even more powerful architectures (Yang et al., 2019). These new architectures have also been applied in more difficult scenarios such as the multi-turn dialogue, where multiple related sentences are present.
The multi-turn conversational paradigm is a challenging task deserving the attention of various studies in the past years. The major problem in multi-turn is how to allow the agent to correctly reference the past conversational history whenever is needed. Different studies used memory networks to do that (Chen et al., 2016) implementing a way to have a knowledge carryover among the conversational turns. Because of the high demanding for such technologies, the research community organized the Dialogue State Tracking Challenge (Williams et al., 2013). This challenge aims to predict what the user wants, at each conversational turn, in terms of requested slots, search constraint slots and dialogue acts (the tasks depend on the challenge version, until now 8 challenges were made). During these years, different works have been presented by using rule-based, generative or discriminative models (Henderson, 2015). While this challenge aims to handle a multi-turn conversation using end-to-end deep learning models, it works only on single sub-conversations having a specific domain. So it does not take care about the possibility that the user can start various conversations having different domains one after the other, thus making difficult to correctly reference conversational history.
An important challenge of a real multi-turn scenario is to understand for how long the context need to be carried for referencing operations. One possibility is to study the sequence of intents inside a conversation and understanding when an intent change happens (Mensio et al., 2018), in this way it is possible to flush the context when a new intent is detected and so a new transaction starts. Anyway there is a main drawback with this approach, since it can not handle a scenario in which two separate conversations, having the same intent, are one after the other such as an user that want to buy two independent cinema tickets, the utterances related to the second ticket purchase belong to a different conversation. In such a scenario the context of the first conversation is not needed during the analysis of the second one. To overcome this limitation we decided to study the discontinuities of acts inside a multi-turn dialogue. A single intent conversation has a typical pattern of acts, it usually starts with information requests and end with thanks of other related final acts. By studying this pattern is possible to understand when there is a discontinuity of acts inside a single intent conversation (e.g. a final act followed by an inform request).

3. Methodology

This section details the methodology proposed in this paper. We firstly describe the approach followed for the modeling of a multi-turn conversation toward the achievement of the user goal. We report the knowledge component that we called Convology. Then, we describe the task and we illustrate the process. We conclude by reporting the used architecture that we called **MTSI-BERT** that stands for Multi-Turn Single-Intent BERT. **MTSI-BERT** is a joint classifier for intent classification, Knowledge Base action (KB-action) prediction and *End-Of-Session* detection.

3.1. Conversational Session

In a multi-turn goal-oriented conversation several related interactions between a user and an agent are present. The resulting conversation can be modeled as an ordered sequence of question-answer *QA* pairs, each one falling under a particular intent that defines the goal. This definition can be formalized such as in Equation 1, where i represents the timestep of the pair and Q and A are natural language sentences.

$$DIALOG = [(Q^{(i)}, A^{(i)})^{+}],\ \forall i \in [0, N] \qquad (1)$$

In order to find a way to classify intents, we adopted the divide-and-conquer paradigm and divided the entire dialogue in sub-sequences having a single intent and a different subject goal. We call these sub-sequences *Conversational Sessions*. A Session is an ordered sub-sequence of a conversation, containing all the interactions to achieve a desired user goal. An example is presented in Table 1.

Session	Turn	Actor	Sentence
A	1	User	I want to book a flight from SF to LA
A	2	Agent	For which days?
A	3	User	This Saturday at 9:00am
A	4	Agent	Ok!
B	1	User	I want to book a flight from Houston to Berlin
B	2	Agent	Which days do you prefer?
B	3	User	Is it possible on Sunday?
B	4	Agent	There are no flights from Houston to Berlin this Sunday. I am sorry.

Table 1: An example of conversation having two subsequent sessions with the same intent (flight booking) but with a different final goal (different flights). The context carried during the first session is not useful for the fulfillment of the second goal. Understanding when a session ends allow to correctly flush the wrong context and make it easier to reference the correct one.

These sessions are independent from each other by definition and so no context has to be carried among them. This modeling allows simplification of the context reference problem during the development of a real conversational AI. The context to be referenced is internal to each session and so, whenever a session ends, the context can be flushed. If information from past sessions are needed, the use of a suitable knowledge base where to store conversational knowledge can be adopted. The flushing of the session permits a simpler reference operation among conversational turns. A formalization of the session is possible as in Equation 2 where *A* is the identifier of the session. Finally the entire dialog can be rewritten as compositions of sessions as in 3.

$$SESSION_A = [(Q_A^1, A_A^1), (Q_A^2, A_A^2), ..., (Q_A^n, A_A^n)] \tag{2}$$

$$DIALOG = [(Q_A^1, A_A^1), ..., (Q_A^n, A_A^n), ..., (Q_B^1, A_B^1), \\ .., (Q_B^m, A_B^m), ..., (Q_F^1, A_F^1), ..., (Q_F^k, A_F^k), ...] \tag{3}$$

3.2. The Knowledge Component: Convology

MTSI-BERT is supported by a knowledge component, namely Convology (CONVersational ontOLOGY), enabling the management of an effective, efficient, and reliable management of multi-turn conversational sessions.

Convology is a top-level ontology aiming to model the conversation scenario for supporting the development of conversational knowledge-based systems. Convology defines concepts enabling the description of dialog flows, users' information, dialogues and user events, and the real-time statuses of both dialogues and users. Hence, systems integrating Convology are able to manage multi-turn conversations and to decouple the intents recognized within the input provided by users from conceptual information triggering both reasoning activities and the generation of answers and feedback.

The purpose of Convology is two-fold. On the one hand, we want to provide a meta-model fully describing the conversation domain from the conversational agent perspective. On the other hand, we want to support the development of smart applications for supporting users in accessing content of knowledge bases by means of a conversational paradigm.

The ontology contains five top-level concepts: *Dialog*, *Actor*, *ConversationItem*, *Event*, and *Status*.

The *Dialog* concept represents a multi-turn interaction between a *User* and one or more *Agent*. A new instance of the *Dialog* concept is created when a user starts a conversation with one of the agents available within a specific application.

The *Actor* concept defines the different roles that can take part into a conversation. Within Convology, we foresee two main roles represented by the concepts *Agent* and *User*. Instances of the *Agent* concept are conversational agents that interact with users. When Convology is deployed into an application, instances of *Agent* concept represents the different agents involved into the conversations with the users adopting the application. Differently, instances of the *User* concept represents the actual users who are dialoguing with the conversational agent. A new instance of the *User* concept is created when a new user starts a conversation within a specific application (e.g. a new user installs the application for monitoring her asthma conditions).

A *ConversationItem* is an entity taking part into a conversation and that allows to represent relevant knowledge for supporting each interaction. Within Convology, we defined four subclasses of the *ConversationItem* concept: *Question*, *Intent*, *Feedback*, and *DialogAction*. An individual of type *Question* represents a possible question that an instance of type *Agent* can send to a *User*. An *Intent* represents a relevant information, detected within a natural language answer provided by a *User*, that a Natural Language Understanding module is able to recognize and that a reasoner is able to process. Differently from a *Question*, a *Feedback* represents a simple sentence that an *Agent* can send to users and for which it does not expect any reply. Feedback are used for closing a conversation as result of the

reasoning process or simply for sending single messages to users without requiring any further interaction. Instances of the *DialogAction* concept describes the next action that an *Agent* individual has to perform.
The *Event* concept describes a single event that can occur during a conversation. Within Convology, we identified three kinds of events: *EventQuestion*, *EventAnswer*, and *UserEvent*. Instances of these concepts enable the storage of information within the knowledge repository, trigger the execution of the reasoning process, and allow the retrieval of information for both analysis and debugging purposes. An *EventQuestion* represents the fact that a *Question* has been submitted to an *Actor*. On the contrary, the *EventAnswer* concept represents an *Answer* provided by an *Actor*. A *UserEvent* represents an *Event* associated with a specific user. The purpose of having a specific *UserEvent* concept instead of inferring *UserEvent* objects from the *EventQuestion* and *EventAnswer* individuals is that a *UserEvent* does not refer only to questions and answers but also to other events that can occur.
The last branch of Convology has the *Status* concept as top-level entity. This branch contains concepts describing the possible statuses of users, through the *UserStatus* and *StatusItem* concepts, or of dialogues, through the *DialogStatus* concept. Instances of the *UserStatus* concept represent which are the relevant statuses of a *User* that the conversational agent should discover during the execution of a *Dialog*. A *UserStatus* individual is associated with a set of *StatusItem* individuals representing atomic conditions under which a *UserStatus* can be activated. Different strategies can be applied at reasoning time, but they are out of scope of this paper. Finally, a *DialogStatus* individual provides a snapshot of a specific *Dialog* at a certain time.

3.3. The Task

The task we aimed to solve is the classification of a multi-turn conversational session. To correctly classify the session defined in Section 3.1. the intent at first has to be extracted, allowing the agent to formulate a first answer. In this setup, the intent is always expressed in the first utterance of each session. This is due to the goal-oriented structure of the conversations we studied. Each time a new interaction begins, this has to contain information about a particular goal, otherwise the agent is not supposed to help the user (no chit-chat paradigm). Then, in order to identify the first interaction within a session, the end of the preceding one has to be detected. We called this the *End-Of-Session* classification. This task is also the one empowering the context flushing for the agent. *End-of-Session* is the most challenging task in this work, because it is not always clear when a session ends and another starts.
Another important aspect we cared about is the interfacing of the classifier with further knowledge base containing all the knowledge needed to the agent to reach the user goal. Imagine a conversational agent that aims to recommend restaurants based on the user requests and preferences. If the user said *"Please add Abanero to my list of my favourite Mexican restaurants"*, the domain of the request is restaurant but it could be useful for the agent to already have the information about the insert operation, in order to distinguish the action from a more common restaurant search. In order to do that we forced our classifier to classify both the intent and the action to perform on a hypothetical knowledge base in order to achieve correctly the user goal. The action could have different values based on the scenario and can give a different granularity to the single intent, allowing thus a better covering of the tractable topics.

3.4. MTSI-BERT

A joint architecture is a particular design choice consisting in training a single model to perform different prediction tasks together. The main idea behind this architectural decision is the same of transfer learning: the already computed latent space representation can be used for solving similar tasks. Joint NLP models were often used to perform intent classification and NER (Name Entity Recognition), sharing the initial layers of the model, and then using individual end branches. Even BERT was trained as a joint model, aiming to predict both masked words and next sentence flag.
MTSI-BERT reuses the same idea to perform the three different tasks jointly: intent and action classification, *End-Of-Session* prediction. The classifier presents a common shared basis composed of 12 pre-trained BERT encoders followed by different branches, one for each task. BERT encodes each single input token of the sentence in an embedding using self-attention modules that compute how much a given token relates to all the others inside the sentence. Each single embedding produced by BERT is 768 dimensions. The embedding contains information of the single word related to the context in which it is, therefore capturing better its semantic inside that particular sentence and solving word ambiguity. This concept differs from the concepts of the original word embedding (Mikolov et al., 2013; Pennington et al., 2014) and takes the name of contextual embedding.
These embeddings are then passed to an upper level composed of two different branches. The branch on the left (Figure 1) is mounted above the *CLS* token embedding and it is trained to predict the *End-Of-Session*. The branch on the right (Figure 1) is used for the classification of both intent and action. The first part of the right branch is a bidirectional LSTM, shared between the two tasks, that takes as input each word contextual embedding. The decision of sharing the same LSTM is born observing that action and intent are often dependent (e.g. a scheduling intent has a bias toward insert action). The output of the LSTM is a 1536 dimensional vector resulting from the concatenation of the right hidden state and the left one. This vector represents the embedding of the entire input sequence and is then used by two upper branches, one for the intent and one for the action.
Each individual branch presents a 3-layer *FFNN* (Feed Forward Neural Network) (Fine, 2006) not shared among the others. This FFNN permits the branch to better specialize on the single task. The parameters learned during training are task-oriented, without them it is more difficult for the output layer to extract the correct features for that particular task starting from the shared representations.
All the branches contain, at the end, a softmax function to output the distributional probability of each class. Figure 1 illustrates the entire structure of **MTSI-BERT**.

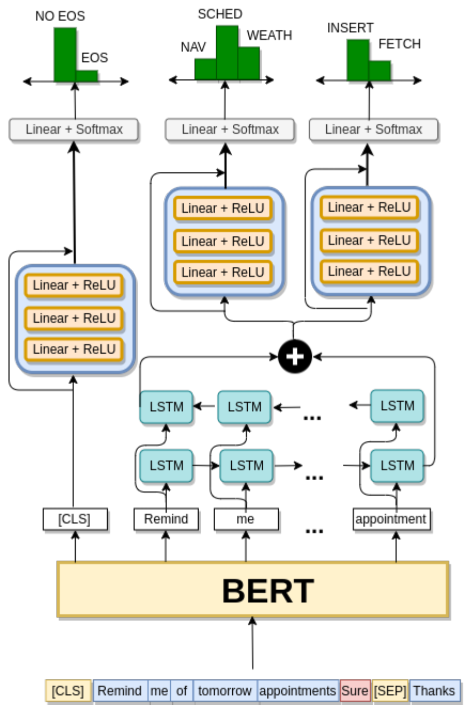


Figure 1: The architecture of MTSI-BERT.

4. Experimental Results

In this section, we report the results and the experimental setup created for this work. The section is divided in four parts. The first is an overview of the dataset we used. The second part contains the metrics used for the model performance computation. The third consists of the description for the optimization process used to train **MTSI-BERT** and the fourth part instead contains the performance scores of the model.

4.1. Dataset

The choice of the dataset was an important step in our work. To correctly apply **MTSI-BERT** we need a dataset containing multi-turn single-intent conversations that allow the adoption of the session. Furthermore the dataset has to present something close enough to the concepts of action and knowledge base. The only dataset found fitting the task is the *Key-Value Retrieval* (KVRET) by Stanford (Eric and Manning, 2017). KVRET is a dataset for the training of multi-turn goal-oriented agents. It contains dialogues between a driver and an agent about three different topics: weather (997 dialogues), navigation (1000 dialogues) and calendar scheduling (1034 dialogues), thus defining the intents. Each dialogue is subdivided in turns that can be "driver" or "assistant", defining the author of a particular utterance. Each turn has also an end of dialogue flag indicating if the current turn is the last one of the dialogue. The data are divided in three sets: the training contains 2425 dialogues, the validation contains 302 dialogues and the test 304 dialogues.
As the name suggests, KVRET main task is about retrieving information requested by the user through natural language interactions, from a knowledge base (e.g. the nearest gas station), for this reason the dataset also contains a fictitious database from which retrieving information. This database is represented as key-value pair and is present for each conversation that requires the agent to extract some knowledge. By using this, we have defined two actions to predict on the knowledge base: insert and fetch. When the knowledge base is present, the action to predict is ”fetch”, otherwise is ”insert”. The total number of fetch versus insert is 2012 vs 413 in the training set, 242 vs 60 in the validation set and 256 vs 48 in the test set. Finally, the concept of session can be adopted by concatenating together different dialogues to form a bigger dialogue where each session is single-intent. In this way, it is possible to predict the end of session by concatenating a session with the first utterance of a random one, which expresses a new user request. The total number of intra-session versus *End-Of-Session* is 6406 vs 2415 in the training set, 748 vs 301 in the validation set and 810 vs 302 in the test set.
The dataset structure is illustrated in Figures 2a and 2b.

4.2. Evaluation Metrics

To evaluate the overall performance of **MTSI-BERT** we opted for evaluating the performance of the model on each single task. As performance score we use the *F-Measure* (Van Rijsbergen, 1979) (also referred as *F1* score). The *F-Measure* is a measure defined as function of precision and recall, balancing between the two (it weighs the precision of the model with its recall). In this way, it is possible to give a meaningful measure with unbalanced classes (which is the case for KVRET dataset). The formula for *F-Measure* is reported in (4).

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4}$$

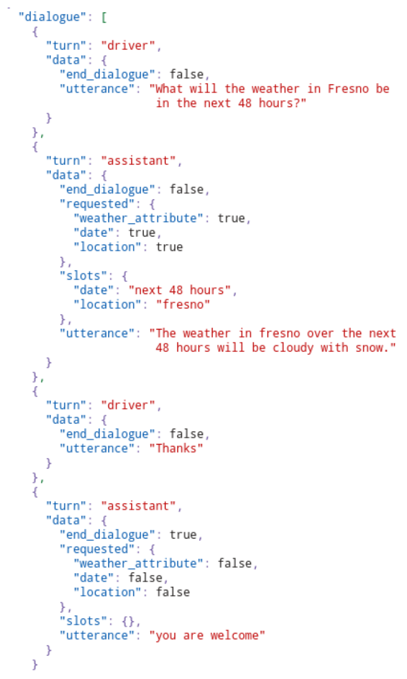
While these metrics are defined for binary classification problems, the extension to multi-class is trivial by considering binary problems for each class c of the type: class c versus not class c. Once all the precision scores for each class have been computed, an average of them have to be made. The scores for this work take into consideration the macro average only for *F-Measure*, due to presence of unbalanced classes. With macro average the F1-score for each class is computed first and then all the scores are averaged. This type of average is used in case of unbalanced class, since it penalizes more errors on the minority class. Equation 5 reports the macro average.

$$F1_{macro\_avg} = \frac{F1_{class1} + F1_{class2} + F1_{class3}}{3} \tag{5}$$

Regarding the *End-Of-Session*, the prediction is performed between the current session and a random one. To correctly visualize the final accuracy, we compute and report also the mean value and the standard deviation for the score.

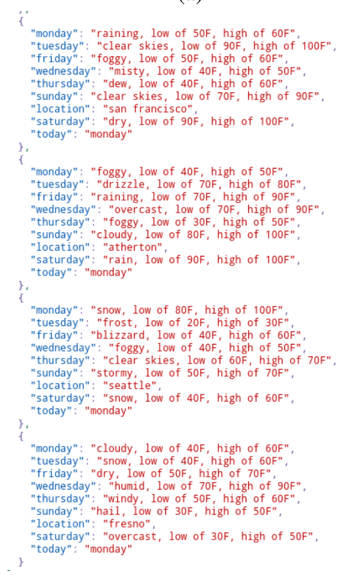
4.3. Optimization

The experiments start with a 20 epochs training to test the convergence of the model loss, then the epochs were raised to reach 100.
The loss used for the training of the model is a *Cross-Entropy Loss*. This loss penalizes the network to produce the expected class value maximizing the confidence,



```
"dialogue": [
  {
    "turn": "driver",
    "data": {
      "end_dialogue": false,
      "utterance": "What will the weather in Fresno be
                    in the next 48 hours?"
    }
  },
  {
    "turn": "assistant",
    "data": {
      "end_dialogue": false,
      "requested": {
        "weather_attribute": true,
        "date": true,
        "location": true
      },
      "slots": {
        "date": "next 48 hours",
        "location": "fresno"
      },
      "utterance": "The weather in fresno over the next
                    48 hours will be cloudy with snow."
    }
  },
  {
    "turn": "driver",
    "data": {
      "end_dialogue": false,
      "utterance": "Thanks"
    }
  },
  {
    "turn": "assistant",
    "data": {
      "end_dialogue": true,
      "requested": {
        "weather_attribute": false,
        "date": false,
        "location": false
      },
      "slots": {},
      "utterance": "you are welcome"
    }
  }
```

(a)



```
,,
{
  "monday": "raining, low of 50F, high of 60F",
  "tuesday": "clear skies, low of 90F, high of 100F",
  "friday": "foggy, low of 50F, high of 60F",
  "wednesday": "misty, low of 40F, high of 50F",
  "thursday": "dew, low of 40F, high of 60F",
  "sunday": "clear skies, low of 70F, high of 90F",
  "location": "san francisco",
  "saturday": "dry, low of 90F, high of 100F",
  "today": "monday"
},
{
  "monday": "foggy, low of 40F, high of 50F",
  "tuesday": "drizzle, low of 70F, high of 80F",
  "friday": "raining, low of 70F, high of 90F",
  "wednesday": "overcast, low of 70F, high of 90F",
  "thursday": "foggy, low of 30F, high of 50F",
  "sunday": "cloudy, low of 80F, high of 100F",
  "location": "atherton",
  "saturday": "rain, low of 90F, high of 100F",
  "today": "monday"
},
{
  "monday": "snow, low of 80F, high of 100F",
  "tuesday": "frost, low of 20F, high of 30F",
  "friday": "blizzard, low of 40F, high of 60F",
  "wednesday": "foggy, low of 40F, high of 50F",
  "thursday": "clear skies, low of 60F, high of 70F",
  "sunday": "stormy, low of 50F, high of 70F",
  "location": "seattle",
  "saturday": "snow, low of 40F, high of 60F",
  "today": "monday"
},
{
  "monday": "cloudy, low of 40F, high of 60F",
  "tuesday": "snow, low of 40F, high of 60F",
  "friday": "dry, low of 50F, high of 70F",
  "wednesday": "humid, low of 70F, high of 90F",
  "thursday": "windy, low of 50F, high of 60F",
  "sunday": "hail, low of 30F, high of 50F",
  "location": "fresno",
  "saturday": "overcast, low of 30F, high of 50F",
  "today": "monday"
}
```

(b)

Figure 2: (a) A sample of conversation from KVRET. (b) The database attached to conversation containing the knowledge to handle the requested task. (a).

without taking care about how the remaining probability is distributed among the other classes. The classes were weighted in loss computation by a factor of $\frac{support_C}{support_x}$, where x is the class taken in consideration and C the most frequent one. This in order to avoid bad model behaviours in unbalanced dataset. In the loss computation also a L_2 penalization (weight decay) is introduced with a factor of 0.1, thus penalizing high weight values.

Joint models typically produce more than one probability distribution. Since a deep neural network can be trained to optimize one single loss, the three different tasks of **MTSI-BERT** must produce, at the end, only one loss to minimize. There are several ways to combine losses together, in this work we opted for using a sum. The sum ensures to give, from the loss perspective, the same importance to all the three tasks. Equation 6 shows the final loss of **MTSI-BERT**.

$$\mathscr{L}_{joint} = \mathscr{L}_i + \mathscr{L}_a + \mathscr{L}_{eos} \tag{6}$$

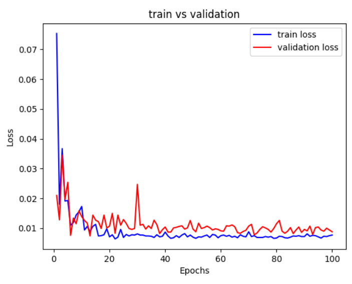
Two different learning rates were chosen, one for BERT and one for the upper level. The reason behind this decision is that BERT, since it is pretrained, is in a good local minimum already, thus moving weights too fast could cause to loose that minimum. On the contrary, the upper level has to be trained from scratch, thus a bigger learning rate is suggested. The learning rate of BERT was chosen to be $5e^{-5}$ (as the paper suggests) and the one for the upper level was set to $1e^{-3}$. The learning rate was then decreased, after certain milestones, by multiplying it by 0.5, thus helping the model to converge. The output of BERT was passed through a dropout layer with probability 0.5. The dropout, together with the weight decay, was used to avoid overfitting phenomenon. The chosen optimization method is *Adam* (Kingma and Ba, 2014) with a *Mini-batch Gradient Descent* (Ruder, 2016). The mini-batch size was set to 16. It is important to notice that here the batch size represents the number of dialogues and not the number of sentences.

MTSI-BERT was developed in *PyTorch* (Paszke et al., 2017) deep-learning library together with *Transformer* (Wolf et al., 2019) Python package, that provides the pretrained BERT. Results of the training phase are shown in Figures 3a and 3b.

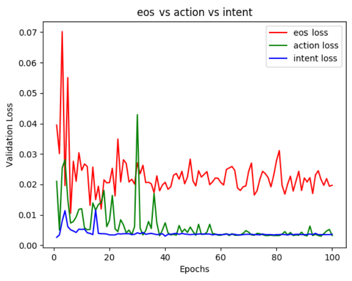
The chart in Figure 3a shows the mean value of $\mathscr{L}_i$, $\mathscr{L}_a$ and $\mathscr{L}_{eos}$ (the losses for each single task). From this chart, it can be noticed how both the training and the validation losses decrease together toward a lower bound error rate smaller than 0.01, therefore ensuring the convergence of the model and the non-presence of overfitting. The chart in Figure 3b shows each single loss on the validation set. Both action and intent losses (the green and the blue ones) converge to a lower error rate compared with the *End-Of-Session* loss and become more stable after 60 epochs. The *End-Of-Session* instead is more unstable and converges to a higher error rate, thus confirming to be the most difficult task among the three.

MTSI-BERT is trained to make it consistent with a multi-turn conversational scenario. Such scenario requires the agent to wait for the first user interaction of the session and understand the intent and the KB-action. It has then to formulate a coherent answer for the user and then come back to wait for the new user interaction. While the answer generation is not part of this work, the main focus remains on the whole session classification. The new user interaction could be related to the previous question-answer pair or not. In the first case a dialogue continuation is happening generating a multi-turn conversation. In the case the new user interaction is not related with the previous exchanged utterances, then a new session begins, with its intent and its KB-action. To correctly flush the context of the past conversation, the agent has to be aware about the end of the session. Additionally, it has to trigger the classification of KB-action and intent for the first utterance of the new session.

To correctly adapt the agent to work in this way, the input



(a)



(b)

Figure 3: (a) Loss trend on validation and training set. No overfitting is present since we can notice from the plot that the training and the validation losses decrease together. (b) Comparison of loss trend among the three tasks on the validation set. The *End-Of-Session* (here reported in red) shows itself as the most difficult one.

was shaped to simulate this type of scenario. **MTSI-BERT** was trained with a QAQ triplet as input. Such triplet consists in the concatenation of the last three exchanged utterances, where Q represents the user question and A the agent answer. **MTSI-BERT** is able to learn the relations existing between the new question of the user and the preceding exchanged *QA* pair. This task is possible because it exploits, for this task, the *CLS* embedding of BERT. This is the embedding used in the BERT original task to predict the next sentence flag. It contains information about the similarity between what comes before the *SEP* and what comes next. Figure 4 shows the input for **MTSI-BERT** at each timestep.

4.4. Results

We developed three models, each performing one of the tasks for which **MTSI-BERT** was trained: intent, action and *End-Of-Session*. They consist of a bidirectional LSTM that receives *spaCy* (Honnibal and Montani, 2017) pretrained embedding and output, through a softmax, the distributional probabilities for the three tasks. These models are used to better show the performance of **MTSI-BERT** versus other non-joint models that perform the same task and we consider our reference. Table 2 reports the results of the reference mode, a state of the art intent discriminator, and our **MTSI-BERT**.

MTSI-BERT performs better in action and *End-Of-Sessions*. The intent discrimination is already solved by the reference, however this requires further investigation given the small number of intents in the KVRET dataset. The most promising results are on the *End-Of-Session*, whose improvement is an additional 3% from the reference.

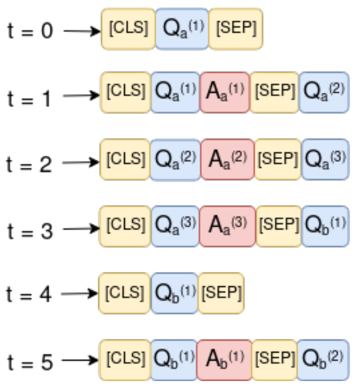


Figure 4: The input of MTSI-BERT. Here the dialogue is composed of two sessions A and B. The MTSI-BERT receives the first sentence of the session and then subsequent triplets of the conversation. When the session end is detected at timestep 4, the triplet is reset and the model is asked to classify the first sentence of the next session.

Model	**F1(Intent)**	**F1(Action)**	**F1(EOS)**
MTSI-BERT	1.00	1.00	0.9938 $\pm$ 0.0005
(Mensio et al., 2018)	0.9987	-	-
Reference	1.00	0.9937	0.9638 $\pm$ 0.0006

Table 2: Results on the test set of *KVRET*.

Since Table 2 contains results which are very close to the perfect prediction, we have come to the conclusion that the task on this particular dataset is fairly easy to be solved and then even the reference model is good enough to be used. In particular, we have noticed three following weakness of KVRET for our tasks. It contains only 3 intents, thus forbid a better topics covering, important to simulate a real conversation. The possible actions we have extracted are only fetch and insert, as a binary task it is quite trivial to be solved. Last, the dialogues end very often with the thanks of the user or the greetings of the assistant, then making the end of session easy to detect. To really assess the capabilities of MTSI-BERT, new tests on a more challenging dataset have to be performed.

5. Real case scenario

In this chapter, we propose a real case scenario of **MTSI-BERT**. *PuffBot* is a chatbot for the support of asthmatic people. It has two main features: help patients to keep under control their asthma and help doctors to monitor easier their health status. *PuffBot* assists patients while conversing and infers their emergency code. The patient tells to *PuffBot* the last events related to his asthma. Then it infers his status with some reasoning processes on the understood information and on the patient generalities such as practiced sports, smoker habits and allergies that the patient lists during the initial registration phase. An emergency code is a label, defined with the Trentino healthcare department, that represents the status of the patient from a medical

perspective. After the emergency code prediction, *PuffBot* suggests a possible advice to the patient. We have defined, along with the domain experts, 4 different codes. The green emergency code is associated with a normal patient situation. Possible events associated with this category are the absence of cough or cough during sport activities. The typical suggestion is to continue with the current therapy. The yellow emergency code is associated with mild cases, for which a doctor intervention is typically not needed. Events like cough with rhinitis, cough during night and other types of coughs belong to this case. The suggestion for a yellow code is to increase the number of inhaler puffs. *PuffBot* comes back to monitor the status after a while and, if it gets worse, the doctor intervention is suggested. The orange emergency code is the last *PuffBot* can handle and is the one that associated to potentially dangerous situations. Events like persistence cough and sense of chest constriction requires to follow a particular therapy defined with domain experts during the development of the chatbot. If the situation gets worse, doctor intervention is needed. The red code refers to an a emergency situation for which the doctor visit is needed. *PuffBot* suggests the patient to call a doctor and continue to monitor the patient after the doctor visit. From a design point of view, *PuffBot* relies on a multi-turn conversational paradigm and a knowledge base where all domains (macro-categories of conversational topics identified by a set of intents and related slots) and conversational knowledge are stored. The knowledge base relies on *Convology*.
In this scenario, the use of **MTSI-BERT** helps the development of the agent. The conversations between the patient and *PuffBot* can be divided in sessions, each one regarding a particular patient intent. For instance a possible session can be the patient talking about the cough episode of the last week, or his struggling to breathe of the last night. Then the information about these sessions have to be correctly extracted and stored inside the knowledge base, in order to infer the patient emergency code when the conversation ends. Thus to correctly store the information of each session, the agent needs to be aware when a particular session ends, allowing then the storing of final understood information and the flush of the past context (which resume is available in the knowledge base).

6. Conclusion and Future Work

In the proposed work, we have introduced a challenge in the domain of multi-turn conversational agents. Multi-turn conversational paradigm consists of a simulation of real conversations between humans, where different utterances can be exchanged in order to fulfill the desired goal. Such paradigm has proven to be difficult to manage since it requires to solve a lot of problems in the natural language understanding field, specifically related the contextual understanding task. For instance, the co-reference resolution has to be done easily, together with the carry of the dialogue context through the conversation.
To analyze this paradigm from a more practical perspective, some approximations of the real world were done. We divided the entire dialogue in subsections called sessions. Each session is characterized by a single intent to be fulfilled through a single action on a knowledge base, which is a structured representation for data that enables dialogue tracking, fast data retrieval and reasoning processes. Then a model to classify these sessions was proposed. *MTSI-BERT* is a joint model, based on BERT, for intent and KB-action classification within a dialogue session and *End-Of-Session* detection. Such model has reached very good results on *KVRET* dataset, a dataset containing dialogues between a driver and an integrated car assistant. Thus providing a fertile soil for further studies. Anyway the obtained results are influenced by KVRET. Infact, this dataset contains only 3 intents, 2 KB-actions and a repetitive pattern for the *End-Of-Session* (the conversations end almost always with thanks of greetings of one of the two actors). MTSI-BERT can be improved by training it on a more challenging dataset and by integrating, together with the other tasks (e.g. response generation), the *Name Entity Recognition*. The session classification was done by assuming correct agent response. Wrong agent replies affect the way this model performs and this can lead to unexpected system behaviour. The agent reply must not only be grammatically correct, but it needs to be also semantically coherent. A way to generate a natural language text given some constraints (Miao et al., 2019) (e.g. the knowledge base features extracted after the reasoning process) will require further studies. A dialogue tracker system has also to be implemented. Another core problem in such scenario is to find a way, for the agent, to carry on the conversation dynamically if some information is still missing. We believe this is an important module to design.
Finally an use case was described for which MTSI-BERT is a useful component for its piloting. *PuffBot* is a chatbot for helping and monitoring people suffering from asthma. It is based on a multi-turn scenario and it has a knowledge base for saving all the status information about the patient and the information extracted from the conversation, in order to infer its emergency code and support him accordingly. We are also experimenting MTSI-BERT to the deployment of a conversational agent meant to support migrants by lowering the bureaucracy burden when seeking for actionable information to ask for a residence permit or family reunion, do tax declarations.
Future studies will cover other related challenges such as the *Dialogue State Tracking Challenge* (DSTC), Reinforcement Learning for dialogue policy generation and NLG (Natural Language Generation).

Acknowledgments

- Computational resources were provided by HPC@POLITO, a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it). Testing operations were performed using the Google Colab platform.

- The authors acknowledge the funding received from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 870980 "Enabling immigrants to easily know and exercise their rights", Call: H2020-SC6-MIGRATION-2018-2019-2020, Topic: DT-MIGRATION-06-2018-2019.

7. Bibliographical References

Adiwardana, D., Luong, M.-T., So, D. R., Hall, J., Fiedel, N., Thoppilan, R., Yang, Z., Kulshreshtha, A., Nemade, G., Lu, Y., et al. (2020). Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Chen, Y.-N., Hakkani-Tür, D., Tür, G., Gao, J., and Deng, L. (2016). End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Interspeech*, pages 3245–3249.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Eric, M. and Manning, C. D. (2017). Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414*.

Fine, T. L. (2006). *Feedforward neural network methodology*. Springer Science & Business Media.

Graham, C. and Jones, N. B. (2016). Intelligent virtual assistant's impact on technical proficiency within virtual teams. *International Journal of Virtual and Personal Learning Environments (IJVPLE)*, 6(1):41–61.

Henderson, M. (2015). Machine learning for dialog state tracking: A review. In *Proceedings of The First International Workshop on Machine Learning in Spoken Language Processing*.

Honnibal, M. and Montani, I. (2017). spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 7.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mensio, M., Rizzo, G., and Morisio, M. (2018). Multi-turn qa: A rnn contextual approach to intent classification for goal-oriented systems. In *Companion Proceedings of the The Web Conference 2018*, pages 1075–1080. International World Wide Web Conferences Steering Committee.

Miao, N., Zhou, H., Mou, L., Yan, R., and Li, L. (2019). Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.-Y., Gao, J., and Dolan, B. (2015). A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Tulshan, A. and Dhage, S., (2019). *Survey on Virtual Assistant: Google Assistant, Siri, Cortana, Alexa: 4th International Symposium SIRS 2018, Bangalore, India, September 19–22, 2018, Revised Selected Papers*, pages 190–201. 01.

Van Rijsbergen, C. J. (1979). Information retrieval.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Williams, J., Raux, A., Ramachandran, D., and Black, A. (2013). The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Yao, K., Zweig, G., Hwang, M.-Y., Shi, Y., and Yu, D. (2013). Recurrent neural networks for language understanding. In *Interspeech*, pages 2524–2528.