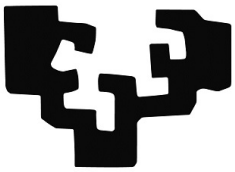
eman ta zabal zazu



Universidad del País Vasco
Euskal Herriko Unibertsitatea

SPIN - Speech Interactive Research Group

Contributions to Attributed Probabilistic Finite State Bi-Automata for Dialogue Management

Thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy by:

Manex Serras Saenz

Supervisor Prof. María Inés Torres
Speech Interactive Research Group
Universidad del País Vasco (UPV/EHU)

Supervisor Ph.D. Arantza del Pozo
Speech and Natural Language Technologies Department
Vicomtech Foundation, Basque Research and Technology Alliance (BRTA)

Manex Serras Saenz
Contributions to Attributed Probabilistic Finite State Bi-Automata for Dialogue Management
Supervisors: Prof. María Inés Torres and Ph.D. Arantza del Pozo
University of the Basque Country
SPIN - Speech Interactive Research Group
Vicomtech
Speech and Natural Language Technologies Department

Declaration

I, Manex Serras, declare that this thesis titled, 'Contributions to Attributed Probabilistic Finite State Bi-Automata for Dialogue Management' and the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Manex Serras Saenz

Abbreviations

A-PFSBA: Attributed Probabilistic Finite State Bi-Automata

ADL: Average Dialogue Length

AP: Attributed Path

AR: Augmented Reality

BAUM: Bi Automata User Model

CMU: Carnegie Mellon University

CPI: Control Panel Interface

DA: Dialogue Act

DB: Data Base

DD: Domain Database

DDM: Domain Database Middleware

DL: Deep Learning

DM: Dialogue Manager

DS: Dialogue System

DSTC: Dialogue State Tracking Challenge

DSTC2: Dialogue State Tracking Challenge 2

HDM: Hybrid Decision Making

HMI: Human-Machine Interaction

HOL: Hybrid Online Learning

IL: Incremental Learning

KNSS: K-Nearest State Smoothing

LSTM: Long Short Term Memory

MB: Mapping Blackboard

MDP: Markov Decision Process

ML: Machine Learning

MLP: Multi Layer Perceptron

MP: Maximum Probability

MPA: Mobile Phone Application

MPP: Maximum Probability Path

MSA: Maximum Scoring Action

NLU: Natural Language Understanding

NLG: Natural Language Generation

MNB: Multinomial Naive Bayes

NSS: Nearest State Smoothing

OL: Online Learning

PA: Passive Aggressive

PFSBA: Probabilistic Finite State Bi-Automata

PM: Pruning Model

POMDP: Partially Observable Markov Decision Process

QM: Quality Metric

RC: Random Choice

RF: Random Forest

RL: Reinforcement Learning

RS: Random Sampling

SDS: Spoken Dialogue System

SE: Search Engine

STT: Speech to Text

SUS: System Usability Scale

SVM: Support Vector Machines

TC: Task Completion

TCP: Task Completion Path

TSA: Thresholded Scoring Action

TTS: Text to Speech

UM: User Model

UX: User Experience

VAD: Voice Activity Detection

Abstract

Task-oriented Spoken Dialogue Systems (SDSs), also known as Conversational Assistants, have been generating a great deal of interest in recent years, as they can be used to automate repetitive and low-value tasks and processes which use a natural communication channel. One basic component of every task-oriented SDS is the Dialogue Manager (DM), which is responsible for tracking the current state of the conversation and for deciding the next action of the system.

This dissertation intends to improve a data-driven framework based in stochastic finite-state transducers for DM modelling in task-oriented SDSs: the Attributed Probabilistic Finite State Bi-Automata (A-PFSBA). Several contributions are presented that enhance the A-PFSBA based DM in different aspects. First, its model generalisation mechanism is improved to better employ context, the semantic relation between dialogue states and the spatial relations of the dialogue state space. Second, the A-PFSBA theoretical framework is extended for policy-making. In the same way, multiple policies with different degrees of complexity are implemented following this formulation. Third, a simple-yet-effective algorithm is proposed to incrementally learn an initial DM, which can be adjusted to work under uncertainty. Finally, the potential of the A-PFSBA framework to be deployed in data scarcity and zero-data scenarios and its capability to bridge the gap between data-driven and rule-based paradigms for DM development is tested.

The presented contributions have been validated using two well-known corpora: the Let's Go corpus and the Dialogue State Tracking Challenge 2 corpus. In order to validate the viability of the A-PFSBA framework in industrial scenarios, three applications that employ the A-PFSBA formulation and which have been validated by real users are also presented.

Acknowledgement

Lehenik eta behin, nire familiakoei eta senitartekoei eskerrak eman nahi dizkiet, bereziki gurasoei, arrebari, eta osaba Antxoni. Saenz-tarren usadioei jarraituz, orain dela urte batzuk hasi nintzen tesi deritzon mendi honetan gora, zeinetan ez diren falta izan aldapa zailak eta harri irristakorrak. Bada, hemen naiz jada, uste baino gogorragoa egin zaidan arista honen bukaeran. Gailurrean. Hau ezinezkoa izango litzateke zuen babesik, pazientziarik eta elkartasunik gabe. Argi izan aurrean duzuen tesi hau ez dela zuen lanaren, izanaren eta erakutsitakoaren islada besterik. Aitona Antonio Serrasek esango luken bezala, lurra ona zen, baita ereindako hazia ere.

También agradecer profundamente a mis directoras, María Inés Torres y Arantza del Pozo, ya que sin ellas este trabajo habría sido imposible de realizar. Cabe destacar que la palabra "*directoras*" cobra especial relevancia, ya que han sabido dirigirme con elegancia y asertividad a enfocar mis esfuerzos hacia un trabajo efectivo.

No me olvido del departamento de Tecnologías del Habla y del Lenguaje Natural, testigos de los innumerables cafés que ha conllevado la realización de este trabajo. También agradecer a todos los compañeros de viaje y faena de Vicomtech, que han contribuido a crear un clima de trabajo y compañerismo fabuloso. Ha sido un placer y un privilegio crecer como profesional y como persona junto a vosotros.

I would also like to thank Arash Eshghi and Oliver Lemon for giving me the opportunity to intern at the Heriot Watt University. Being part of such an amazing and welcoming team was a great experience that I will never forget.

Horrez gain, kuadrilakoei eta lagunei (betidanik egon direnei, eta baita bidean zehar egindakoei) ere eskerrak eman nahi dizkiet. Bide luze honetan zehar aldamean izan zaituztedalako, bizkar gainean neraman zama arinagoa eginez.

Y por último, pa ti pexioxa, collacia na mio llocura y abellugu nos mios momentos más baxos, milenta gracies maittia.

Mila esker bihotzez.

Contents

1 Introduction

1.1 Motivation

Technological advances and the digitalisation of society have drastically changed the way we interact with people, companies and almost everything around us. Apart from the telephone and electronic messages, many new digital communication channels (e.g. WhatsApp, Telegram, VoIP, etc.) have emerged in recent years. The democratisation of the Internet, together with the advent of web platforms, smartphones and hardware with enhanced computation capacity, have given companies and institutions a golden opportunity to communicate with their customers through novel interfaces.

The new digital communication landscape has also brought opportunities to the field of Artificial Intelligence and, more specifically, to the field of Dialogue Systems (DSs) – also known as Conversational Assistants. In this context, the long sought-after endeavour to make natural language-based Human-Machine Interaction (HMI) possible is attracting a great deal of interest. In addition, this goal is far more reachable thanks to improvements in hardware and software, which allow powerful Deep Learning (DL) architectures to be deployed for online use of speech and natural language processing technology.

DS can be used to automate repetitive and low-value tasks and processes in multiple domains using a natural communication channel. In addition, as DS can be easily distributed and escalated to handle several simultaneous users, its commercial interest has grown over the last few years.

1.2 Overview and Limitations of Dialogue Systems

The aim of DSs is to interact with their users by using natural language. They are generally grouped into two categories: **Task-Oriented** systems, which are intended to assist the user to solve and automate certain tasks within a particular application context (e.g. checking bus timetables, finding and booking restaurants); and **Open Domain** systems, which are designed to provide reasonable responses and entertainment to the users (H. Chen et al., 2017). The industrial interest in Task-Oriented DS

is nowadays far superior to Open Domain systems, since they offer the possibility of automating repetitive, low value-added and frequent tasks helping reduce costs, optimise processes, manage knowledge, distribute expertise and dedicate employee time to higher value-added chores (Bohus and Rudnicky, 2005b; Serras, Garcıa-Sardiña, et al., 2020b; Reidsma et al., 2016; Crook, Keizer, et al., 2014; Raux, Bohus, et al., 2006; Pineau, Montemerlo, et al., 2003; Garcıa-Sardiña et al., 2020; Lubold et al., 2016; Reidsma et al., 2016; Graesser et al., 2001; Serras, Garcıa-Sardiña, et al., 2020a).

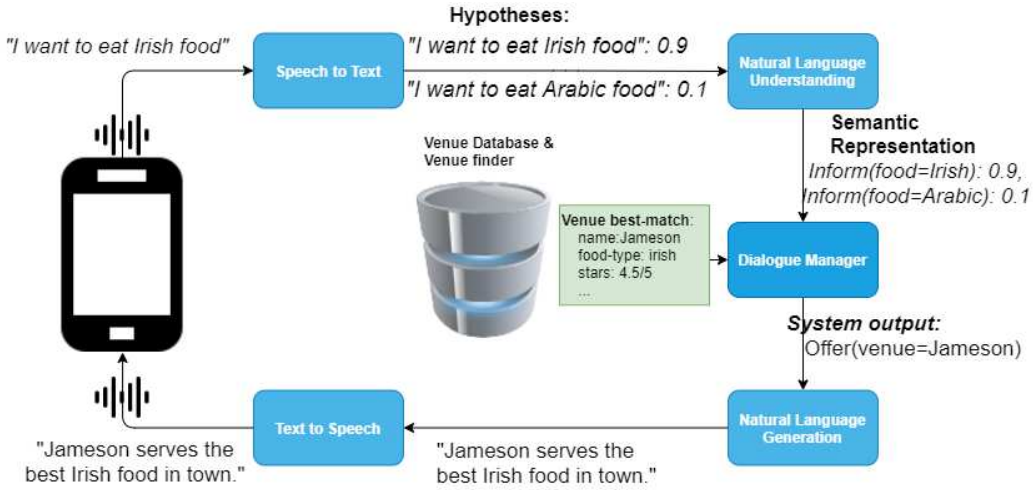


Fig. 1.1.: Illustrative example of the traditional DS pipeline

Traditionally, a DS consists of a pipeline of various technological modules, as shown in Figure 1.1, each having a clearly defined purpose:

- **Speech to Text (STT):** this module transcribes the utterances of the users, returning a set of transcription hypotheses.

- **Natural Language Understanding (NLU):** this component encodes text transcriptions into semantic representations (e.g. "I want pizza" into "inform(food=pizza)"). This way, input text variability is highly reduced in order to be handled by the system.

- **Dialogue Manager (DM):** this module determines the state of the current dialogue, commonly known as *dialogue state*. It is also responsible for consulting external knowledge sources (e.g. databases, profiling systems, and recommendation systems) and for defining the dialogue strategy to follow, commonly known as *dialogue policy*. The dialogue policy determines the response to be given to the user, usually employing a semantic representation.

- **Natural Language Generation (NLG):** this module converts the semantic representation of the response given by the DM into human-interpretable form, usually a text.

- **Text to Speech (TTS):** this module synthesises the text generated by the NLG into audio. This audio response is sent back to the user, then closing the loop.

The layer made up of the STT and TTS modules is characteristic of Spoken Dialogue Systems (SDSs), in which interactions with users are carried out orally. Those DSs without this voice layer are commonly denominated Chatbots and perform written interaction. This dissertation will focus primarily on SDSs. Generally, the inclusion of a STT module in a DS increases the complexity, as the noise channel can have a negative impact on the speech transcription, and this error is propagated to the rest of the modules.

The main topic of this dissertation is related to the DM, the central component responsible for controlling the DS interaction. As mentioned above, its principal tasks involve determining the dialogue state of the current interaction, exploiting external information sources and deciding the response the system should give next in order to meet the users' needs.

Several paradigms have been proposed in the literature for DM development. However, still there is not a mainstream approach capable of solving all the challenges related to dialogue management. This, together with the immediate demand of DSs from the market for process automation, has fostered a clear divergence between industry and academia.

On the one hand, academia proposes data-driven solutions to build technological modules capable of managing dialogue interactions. Despite their adequateness, these techniques require high amounts of tagged data in order to define dialogue policies. As manual dialogue compilation is highly resource demanding, a usual approach is to develop an artificial user or User Model (UM) from a small dataset capable of generating synthetic dialogue samples for training and evaluation purposes (Schatzmann, Weilhammer, et al., 2006).

On the other hand, industrial DSs generally employ handcrafted decision rules for dialogue management due to the lack of annotated data. These methods are hard to escalate to complex scenarios. Also, they have difficulty in handling uncertainty (e.g. when there are multiple transcription hypotheses given by the STT).

Within the technological context above, this dissertation aims to improve a data-driven framework based in stochastic finite-state transducers for dialogue management. Contributions are made in different aspects, such as the model generalisation mechanisms, policy-making and incremental learning. In addition, the potential of the dialogue management framework to bridge the gap between data-driven and rule-based paradigms for DM development is also explored. Note that the presented work targets only Task-Oriented DSs.

1.3 Summary of Contributions

The contributions of this thesis focus on improving and testing the Attributed Probabilistic Finite State Bi-Automata (A-PFSBA) framework proposed to model dialogue interaction in (M Inés Torres, 2013) for DM development in several scenarios. In particular, the following contributions are presented:

- The A-PFSBA framework is tested on two well-known scenarios to determine its validity: Let's Go, a bus information retrieval system; and the Dialogue State Tracking Challenge 2 (DSTC2), a restaurant finder application. The first one provides a challenging corpus collected with real users. The second one provides an statistical NLU which enables to test the capability of the A-PFSBA framework to handle uncertainty and provides user-related goals for better user modelling.

  - An A-PFSBA DM is implemented for the first time on the Let's Go domain and a mirroring UM is built for evaluation purposes.

  - In the DSTC2 domain, the implemented A-PFSBA DM includes a speech recognition error simulation model to test the robustness of the framework against channel noise. In addition, two goal-conditioned UMs are implemented to emulate the behaviour of the users: one based on the A-PFSBA framework and the other one based on a DL architecture that improved the state of the art at that time.

- Three novel methods are proposed to improve the A-PSFBA framework's model smoothing strategy, which allows the DM to transparently recover when unknown situations appear in the dialogue and generalise to new interactions.

  - K-nearest State Smoothing: exploits the nearest dialogue states to better contextualize the recovery strategy.

  - State Pruning: prunes non-useful dialogue states for dialogue recovery.

  - Spatial Relation Learning: learns spatial relations (i.e. similarities, distances and metrics) to better represent the dialogue state space

- The theoretical definition of the A-PFSBA framework is extended to exploitation policies.

- A method is proposed to incrementally improve the A-PFSBA DM model as new dialogues are carried out with the users. It is adapted to interactions under uncertainty.

- A methodology capable of integrating expert rule based DMs and A-PFSBA based DMs is presented, which has the potential to incrementally learn a data-driven DM in zero-data domains.

- Finally, the A-PFSBA framework has been used to build three different DS applications.

  - An Industrial Maintenance prototype, whose goal is to guide the operator in performing a monthly maintenance task over a Universal Robot's gripper combining spoken interaction and Augmented Reality.

  - A Gerontological Data Registration system, which interacts with a Knowledge Base in order to register information related to the tasks carried out by personnel in a nursing home.

  - A Citizen Support application, which aims to help citizens consult the most frequent administrative procedures, subsidies/grants and information concerning the different departments of the Public Administration.

All the presented contributions are experimentally validated in the Let's Go and/or the Dialogue State Tracking Challenge 2 dataset.

1.4 Outline

The remainder of the thesis is organised as follows:

Chapter 2 describes the background on DMs where the A-PFSBA framework is described in detail.

Chapter 3 introduces the corpora used during this dissertation and the baseline systems.

Chapter 4 introduces the A-PFSBA model smoothing and the contributions to improve it.

Chapter 5 details the policy formulation over the A-PFSBA framework and implements and tests different policies using this formulation.

Chapter 6 introduces the incremental learning mechanisms and presents an hybridization mechanism to combine rule-based DMs and A-PFSBA-based DMs to incrementally learn a data-driven A-PFSBA DM.

In order to validate the potential of the A-PFSBA framework, three different applications that employ this formulation are presented in **Chapter 7**.

Chapter 8 summarizes the conclusions of this dissertation and sets the guidelines for future research and open topics.

The experimentation and validation of the presented contributions are included in the corresponding Chapters.

2 Background on Dialogue Systems

This chapter briefly reviews the principal methods that have been used to build Dialogue Managers (DMs) through the history of Dialogue Systems (DSs). Rule-based approaches, where experts encode the knowledge and business rules of the systems are described first. Stochastic techniques capable of exploiting the patterns found in annotated data for dialogue planning are then introduced, including the most popular statistical and deep learning approaches. In addition, the statistical Attributed Probabilistic Finite State Bi-Automata (A-PFSBA) framework employed in this thesis is also presented in detail. Then, the most widely used techniques to simulate User Models (UMs) are described. Finally, a summary of the existing commercial DSs and their technological approaches is made and important DM properties that need to be considered in real application scenarios are presented.

2.1 Rule-based systems

Dialogue Systems have been around for a long time. Initial DMs were built by designing and implementing a set of expert rules (Weizenbaum, 1966; Colby et al., 1971; Bohus and Rudnicky, 2009), since dialogue data were not available at the time. Because data scarcity is a frequent issue, rule-based DMs are quite common and have been employed in several systems (Lemon et al., 2002; Serras, Garcıa-Sardiña, et al., 2020a; Bohus and Rudnicky, 2005b; Ward, 1994; Gupta et al., 2005). Multiple rule-based frameworks have been developed to build conversational systems. VoiceXML (Lucas, 2000) implemented one of the first standards. The Let's Go system (Raux, Langner, et al., 2005; Raux, Bohus, et al., 2006) was based on RavenClaw, a plan-based framework with domain-independent error handling (Bohus and Rudnicky, 2003; Bohus and Rudnicky, 2005a; Bohus and Rudnicky, 2009). And more recently, the probabilistic rule based OpenDial (Lison and Kennington, 2016; Lison, 2015) framework has been implemented.

In recent years, several frameworks have arisen to build DSs, such as DialogFlow, Watson, RASA, Xenioo etc. (Janarthanam, 2017; Khatri et al., 2018; Bocklisch et al., 2017; Cañas and Griol, 2020). Most of these frameworks employ rule-based or deterministic automata-based DMs allowing different mechanisms to build them. Thanks to these frameworks, the popularity and adoption of Chatbots in industrial environments have risen dramatically. Early experiments have been done

to adapt statistical managers onto industry-level frameworks (Cañas and Griol, 2020), but, for now and to our knowledge, these functionalities only exist in beta/early implementations of DialogFlow X and RASA.

Apart from these early works, the rule-based DMs are the predominant ones on industrial Chatbots and DSs. The rules are usually easier to implement, audit, understand and do not require to have a statistical/artificial intelligence background as the data-driven DMs requires. These factors might explain the predominance of rule-based DM.

Despite their popularity and usefulness, rule-based systems have some drawbacks. First, they are hard to escalate. In addition, they lack flexibility to adapt to unforeseen situations and uncertainty, such as Speech To Text errors (STT) and/or Natural Language Understanding (NLU) incertitude. Bridging rule-based and data-driven paradigms to build DMs if really interesting in order to solve these issues.

2.2 Dialogue as a Probabilistic Interaction

Dialogue can also be modelled as a probabilistic interaction between a user and the system. Any two-party dialogue $\mathbf{z}$ can be viewed as a sequence of system and user responses $\mathbf{z} = (u_0, a_0, u_1, a_1, \cdots)$ where u are the user inputs and and a the system actions. A turn can be seen as a tuple of user input a system action $t = (u_t, a_t)$ Then, the probability of a user giving a response at turn t can be roughly explained as:

$$P(u_t|a_{t-1}, u_{t-1}, \cdots u_0, G_u) \tag{2.1}$$

Where G_u is the goal the user wants to fulfill during the interaction (e.g. making an appointment with the doctor, retrieving a bus route, etc.). Also, the probability of the system giving a determined response at time $t+1$ can be modelled as:

$$P(a_{t+1}|u_t, a_t, u_{t-1}, \cdots u_0, G_s) \tag{2.2}$$

Where G_s is the goal of the system, which most often is to satisfy the goal of the user G_u through the dialogue. Usually, and to avoid maintaining all the previous actions to estimate the probabilities of the next user and system actions, a dialogue state is maintained at each turn q_t, which usually summarizes the dialogue history so far (Griol et al., 2008; Paek and Pieraccini, 2008; Gašić, Jurčíček, Keizer, et al., 2010; Ghigi and M Inés Torres, 2015; Levin et al., 1998).

In practice, multiple probabilities need to be combined in order to estimate the next action of the system a_{t+1}. This is because the pipeline of a Spoken Dialogue System (SDS) consists of multiple technological modules, as described in Chapter 1, all of which need to be taken into account in order to generate a system response.

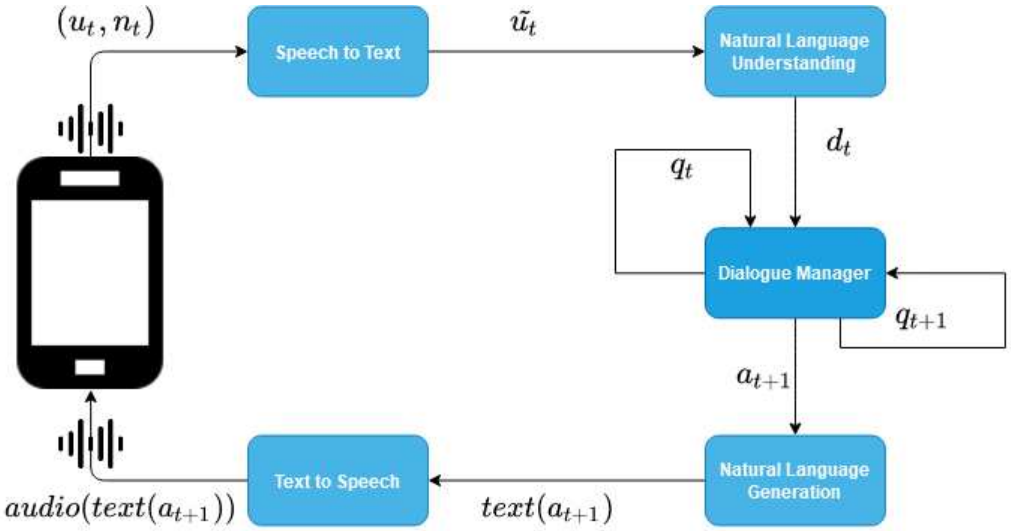


Fig. 2.1.: Items to be taken into account to estimate the next system action

Figure 2.1 depicts the items that need to be taken into account in order to estimate the next system action a_{t+1} in a SDS. The whole interaction works as follows:

1. In the current dialogue state q_t, the user sends a spoken signal u_t with an associated channel noise n_t to the STT module.

2. The STT component converts such speech signal into text hypotheses, $\tilde{u_t} = \tilde{u_t}^1, \tilde{u_t}^2, \cdots$. The received speech signal is perturbed according to the received noise n_t.

3. The NLU module then decodes the received transcription hypotheses $\tilde{u_t}$ into semantic labels that denote the communicative intention of the user d_t. Note that the goal of the NLU is to reduce the variability of the text-representations, so the system is able to understand different user expressions encoding the same communicative intention.

4. Using the current dialogue state q_t and the semantic decoding of the user input d_t, the DM, decides on the next system action a_{t+1} to carry out and updates the dialogue state q_{t+1} to be used in the next turn. This response is usually returned at the semantic level.

5. Finally, the Natural Language Generation (NLG) and Text to Speech (TTS) modules convert the next system action a_{t+1} into text and speech form, respec-

tively. These processes are deterministic, i.e. there is no uncertainty as the system response is fully determined by the DM.

In this scenario, the SDS needs to estimate the following joint probability:

$$P(q_{t+1}, a_{t+1}, u_t, d_t | q_t, \tilde{u_t}) \tag{2.3}$$

Given the modular architecture of the SDS, it can be assumed that some of the items are independent and so, the joint probability can be split into:

- **User Response Model:** $P(u_t|a_t, q_t)$ represents the response of the user to a specific system action in a given dialogue state. Note that since the user goal G_u is hidden for the SDS, it cannot be used for this estimation.

- **Speech to Text:** $P(\tilde{u_t}|a_t, q_t, u_t, n_t)$, which needs to estimate the transcription of the user's speech signal conditioned by the last system response, the current dialogue-state, the user's utterance and the channel noise.

- **Language Understanding:** $P(d_t|\tilde{u_t}, q_t, a_t)$ decodes the semantic meaning d_t of the transcription of the user's spoken input, by using semantic tags.

- **Response Selection:** $P(a_{t+1}|d_t, q_t, G_s)$ where the system decides the next response according to some system-goal G_s, which is defined by the task to be fulfilled. In order to select this response, the decoded semantic tags and the current dialogue state are used.

- **Dialogue State Iteration:** $P(q_{t+1}|q_t, d_t, a_{t+1})$ where the dialogue state is updated using the current dialogue state q_t, the decoded semantic actions of the user d_t and the latest system response a_{t+1}. The updated dialogue state q_{t+1} is then stored in the dialogue manager, for the next interaction turn.

Finally, the next system action a_{t+1}, generally represented using semantic tags, is converted into a text representation and then synthesised into speech. As these two steps are deterministic, once the system action is defined they are not part of the probabilistic formulation.

The above notation has introduced the decoding of users and system responses as semantic tags d_t, a_t. In the context of user-system interactions in DS, these semantic tags are commonly known as Dialogue Acts (DAs) (Hancher, 1979; Asher and Lascarides, 2001; Core and Allen, 1997). DA are denoted by intention tags (e.g. *inform, request, confirm*) which can contain information objects known as slots (*food, address*), with their corresponding atomic values (e.g. [*Chinese, Japanese, Italian,*

...], [*fake street 123*, ...])[1]. The main objective of the DA-based representation is to convert the textual user input, which has a high variability, into a small set of DA without losing explainability – i.e. semantic tags describe the communicative intention of users' text without ambiguities **within the domain of the DS**. For example, the DA tag *'request|price'* describes the following user query texts in a restaurant-finder domain:

- How much does it cost?
- And the price?
- How many dollars?
- How affordable is it?

Then, when performing the conversion from text to DAs ($\tilde{u_t} \rightarrow d_t$), the variability of user inputs is highly reduced, so the DM can handle it. Figure 2.3 in Section 2.3 shows a dialogue where each utterance is encoded with the intention tags *Hello, Inform, Request, Offer* that contain the slots *food-type, neighbourhood, venue*, where each information object has its associated value.

The dialogue state concept q_t is also introduced in the notation above. The dialogue state represents where the dialogue is at a given turn t of the interaction. As the dialogue becomes larger, it is computationally expensive to maintain all the information of previous turns and to estimate the probability of the system response of the current turn a_t. A common practice to avoid this problem and keep the core information in each dialogue is to encode the transitive content (e.g. bus departure, desired food, selected flight etc.) of the dialogue in a summary space (Gašić, Jurčíček, Keizer, et al., 2010; Serras, Marıa Inés Torres, et al., 2017; Iñigo Casanueva et al., 2017; Griol et al., 2008) that serves as the dialogue state. This dialogue state is then updated turn by turn according to the user inputs d_t and the system responses a_t.

2.2.1 Markov Decision Processes

In the late 90's the Markov Decision Process (MDP) theoretical framework was proposed for building stochastic DMs (Levin et al., 1997; Levin et al., 1998). MDPs are discrete-time stochastic directed graphs (Bellman, 1957) which are often employed in control processes (White III and White, 1989; Sezer, 2018; Gocgun et al., 2011). Following the Markovian property, they only use the current time state for decision

[1] System nowadays refer to these concepts as Intents, Entities and Entity-values.

making, with no information regarding previous states. A MDP is fully represented by the following 4-Tuple (S, A, T, R) (Levin et al., 1997; Levin et al., 1998; Roy et al., 2000):

- A set of states $S = \{s_0, \cdots, s_1\}$
- A set of actions $a \in A$
- A transition function which describes the probability of transitioning from one state s to another s' by means of a system action a: $T : S \times A \rightarrow S$ where $T(s, a, s') = P(s_{t+1} = s' | a_t = a, s_t = s)$
- A set of immediate rewards $R(s, s', a)$ which are gained after transitioning from s to s' by using action a.

In the context of a DS, a state s is the dialogue state which encodes the known information in the ongoing interaction by using discrete variables. The set of actions A are the possible system responses and the transition function $T(s, a, s')$ are the probabilities of transitioning from one state to another, through the user's responses to the selected system action a.

MDP-based DS make the following assumptions (White III and White, 1989; Levin et al., 1998):

- **Markovian property:** when an action a_t is taken at time t while in state s_t and the MDP dialogue state changes to s_{t+1}, the Markovian property, i.e. that the stochastic process is memoryless, is satisfied:

$$P(s_{t+1} | s_t, s_{t-1}, \cdots, s_0, a_t, a_{t-1}, \cdots, a_0) = P(s_{t+1} | s_t, a_t)$$

- **Session Reward:** the second assumption defines the reward of the entire stochastic process as the *session reward* $\mathcal{R}$. The *session reward* is the sum of all the individual rewards gathered throughout a dialogue: $\mathcal{R} = \sum_{t=0}^{T} R(s_{t+1}, s_t, a_t)$

2.2.1.1 Policy learning in Markov Decision Processes

Based on the two assumptions above, an optimal policy Π can be achieved by maximizing the expected cumulative rewards E when faced with a known dialogue state s_t for an infinite time horizon:

$$E[\sum_{t=0}^{\inf} \gamma^t R(s_{t+1}, s_t, a_t)]$$

Where a_t is the action chosen by the policy $\Pi(s_t)$ at the current dialogue state. The expectation E is sampled from the distribution $s_{t+1} \sim T(s_t, a_t, s_{t+1})$. γ is the discount factor, which ensures that short-term rewards are preferred to long-term ones.

This expected reward E can be used to search for an optimal policy $\Pi(s)$ to map between dialogue states and system actions, in ideal situations where the transition probability matrix is fully known. Given the Markovian property, only the last dialogue state is taken into account for decision making, so the policy-matrix can be written as $\Pi(s, a) = P(a_t = a|s_t = s)$. In order to maximize the expected reward E, an optimization problem is defined over some functions. The *value function* $V^{pi}(s)$ is a recursive function that calculates the expected reward from an starting dialogue-state s, given some policy π:

$$V^{\pi}(s) = \sum_{a} \pi(s, a) \sum_{s'} T(s, a, s')[R(s, s', a) + V^{\pi}(s')]$$

Then, we can define the $\mathcal{Q}$-function as:

$$\mathcal{Q}^{\pi}(s, a) = \sum_{s}^{'} T(s, a, s')[R(s, s', a) + V^{\pi}(s')]$$

Which returns the expected reward if action a is taken from state s'. Then, $V^{\pi}(s)$ can be re-written as:

$$V^{\pi}(s) = \sum_{a} \pi(s, a) \mathcal{Q}^{\pi}(s, a)$$

Then, the optimal dialogue policy $\pi^*(s)$ is the deterministic policy which gives the maximum value function $V^*(s)$:

$$V^*(s) = max_{\pi} V^{\pi}(s) \; \forall s \in S$$

Where S is the set of all dialogue states. The optimal value function can be found by solving:

$$V^*(s) = max_a \sum_{s'} T(s,a,s')[R(s,s',a) + V^*(s')]$$

Similarly, the optimal $\mathcal{Q}$ function can be found by solving:

$$\mathcal{Q}^*(s,a) = \sum_{s'} T(s,a,s')[R(s,s',a) + max_a \mathcal{Q}(s',a')]$$

and then the required optimal dialogue policy is given by:

$$\pi^*(s) = argmax_s \mathcal{Q} * (s,a)$$

In practice, as the state space S can be very large, the transition probability matrix $T(s,a,s')$ is not fully observed. In addition, due to user behaviour uncertainty and/or because of the small amount of available data samples Z (usually $|S|^2 >> |Z|$), the transition probabilities can be difficult to estimate. In this situation, a common methodology involves developing a simulated User Model which will serve as a training environment, optimizing the policy π by using Reinforcement Learning (RL) or dynamic programming (Sutton and Barto, 1998; Schatzmann, Thomson, Weilhammer, et al., 2007; Schatzmann, Thomson, and S. Young, 2007a; S. J. Young, 2000).

MDP-based dialogue systems have been used in the literature due to their interpretable theoretical framework and usability (Eshghi et al., 2017; Levin et al., 1998; S. J. Young, 2000; Levin et al., 2000; Singh et al., 2000). Nevertheless, one of their main drawbacks is that for a full control of the process it needs to be fully observed, which, in practice, does not happen in DSs – where either channel noise corrupts the output of the STT with errors or the NLU outputs multiple hypotheses at the same time. In order to solve this limitation, Partially Observable Markov Decision Processes (POMDPs) (Parr and Russell, 1995; Russell et al., 1995) were introduced within the PARADISE framework (Walker et al., 1997), as an enhanced approach for statistical DS modelling (Fromer, 1998).

2.2.2 Partially Observable Markov Decision Processes

As stated previously, DSs need to handle uncertainty due to diverse factors such as STT induced-errors and/or NLU uncertainty. POMDPs were proposed (Roy et al., 2000) as a variant of the well-established MDP theoretical framework and were established as a popular statistical framework for DM modelling (Roy et al., 2000;

Gašić, Jurčıček, Keizer, et al., 2010; Schatzmann, Thomson, and S. Young, 2007b; Gašić, Breslin, et al., 2013; Paek and Pieraccini, 2008; Williams and S. Young, 2007; Zhang et al., 2001; Lei et al., 2019; S. Young et al., 2010).

In order to enhance the initial MDP framework, the POMDP framework adds the following elements:

- A set of observations $O = \{o_1, o_2, \cdots, o_{|O|}\}$ that the system can receive from the environment. For the DS, these observations are estimations of the user dialogue acts.

- An observation probability $O(o, s, a) = P(o|s, a)$, which encodes the probability distribution of a given observation when the state s and the action a are known.

- The belief state b_i, which is a probability distribution that encodes the probability of being at a given dialogue-state $s_i \in S$, since it is assumed that the current state is not fully observable.

Then, the POMDP works as follows. At each time-step the system is at some unobservable state $s_t \in S$, whose belief state is denoted as b_t. Based on the belief state, the machine selects an action $a_t \in A$, receives a reward $r(s_t, a_t)$ and transitions to a new, also unobserved, dialogue state s_{t+1} which depends only on s_t and a_t. Then, the machine receives an uncertain observation o_{t+1} which is dependant on s_{t+1} and a_t. Finally, the belief state distribution is updated:

$$b_{t+1}(s_{t+1}) = \frac{P(o_{t+1}|s_{t+1}, a_t, b_t)P(s_{t+1}|a_t, b_t)}{P(o_{t+1}|a_t, b_t)}$$

The estimation of the probability $P(s_{t+1}|a_t, b_t)$ requires to compute the probability $P(s_{t+1}|a_t, b_t, s_t)$ for each dialogue state $s \in S$:

$$b_{t+1}(s_{t+1}) = \frac{P(o_{t+1}|s_{t+1}, a_t) \sum_{s_t \in S} P(s_{t+1}|a_t, s_t, b_t)P(s_t|a_t, b_t)}{P(o_{t+1}|a_t, b_t)}$$

Due to the computational cost of this operation when the state-space Q increases, which makes it intractable for the online consumption required by DSs, initial approaches for dialogue management using POMDPs were employed in proof of concepts (S. Young et al., 2010; Paek and Pieraccini, 2008; S. Young, 2006). In order to allow POMDPs to be deployed into more realistic production environments, the compression of the belief state has been widely researched (S. Young et al., 2010;

Pineau and Thrun, 2001; Williams and S. Young, 2007; Williams and S. Young, 2005; Thomson, Schatzmann, et al., 2007). Often, these compression methods require a domain ontology or a handcrafted partition/clustering of the dialogue states (S. Young et al., 2010). The need of tracking the dialogue state employing real-time mechanisms motivated the Dialogue State Tracking Challenges (Williams, Henderson, et al., 2014; Henderson, Thomson, and Williams, 2014; Kim, D'Haro, et al., 2016; Kim, D'Haro, et al., 2017). These challenges focused on tracking the current dialogue state s_t by using supervised models such as Recurrent Neural Networks, Conditional Random Fields, Decision Trees and so on (Williams, Raux, et al., 2016; Henderson, Thomson, and S. Young, 2014). These methods are faster when performing the dialogue state inference, but they require tagged data to learn from.

2.2.3 Deep Learning for Dialogue Management

In recent years, Deep Learning (DL) approaches have been used in several fields of Natural Language Processing such as named entity recognition, translation, text classification, speech translation and others (Arzelus et al., 2018; Etchegoyhen et al., 2020; Azpeitia et al., 2020; Perez et al., 2019; Pablos et al., 2020). Such models have been used for open-domain DS (Adiwardana et al., 2020; Roller et al., 2020). These models are trained over a large dialogue corpora and usually consist of models with billions of parameters, which makes their efficient training a challenging task. Some transfer learning mechanisms were successfully implemented to turn language models intro chatbots (Wolf et al., 2019), where language representation models such as BERT and GPT can be used (Radford, Narasimhan, et al., 2018; Radford, J. Wu, et al., 2019; Devlin et al., 2018). Note that these open-domain systems do not aim at completing a task.

When facing task-oriented scenarios other challenges arise, such as the need to take into account the user goal and the state of the task to be performed, so the DM can plan accordingly. To this end, several proposals can be found which employs Neural Networks or DL. The RL paradigm, widely used in POMDPs, is used for DL models (Cuayáhuitl, S. Yu, et al., 2016; Cuayáhuitl, 2017; Z. Lipton et al., 2018; Inigo Casanueva et al., 2018; Budzianowski et al., 2017). In addition, the dialogue management problem is handled as a sequence generation, where the network receives the sequence of the turns of the dialogue and it needs to predict the next system response (Crook and Marin, 2017; Layla et al., 2016; Serban et al., 2015; Wen et al., 2016). Other approaches employ generative-discriminative training mechanisms to model the DM as a generative network (Li et al., 2017). The challenge of modelling a DM can also be interpreted as a response retrieval

mechanism, where the next system response needs to be selected from a set of possible answers (Serras, Maria Inés Torres, et al., 2019b; Zhu et al., 2017; Bordes et al., 2016).

Recent advancements in DL methods and architectures have allowed to reduce the components of the traditional DS architecture into a single module. These end-to-end DSs employ one DL module to understand the user, plan the next system action and create a surface form of the given system response (Zhu et al., 2017; Zhao and Eskenazi, 2016; L. Xu et al., 2019; Liang et al., 2020; B. Liu and Lane, 2018).

Despite the promising results, several bottlenecks still limit the deployment of task-oriented DL architectures in industrial environments. One of the main limitations is the lack of available data and corpora to train this type of systems. So as to overcome this issue, some research has been carried out on transferring knowledge from DL models trained on existing DS datasets to a target domain with low resources by using only a few samples (Shalyminov et al., 2019b; Shalyminov et al., 2019a). Nevertheless, for under-resourced languages, where such kind of datasets do not exist, these approaches are still unfeasible. In addition, their black-box nature and their lack of transparency and auditability are also limiting factors. Due to these reasons, the DL-based DSs still require further research to be considered a suitable method to build industrial-level solutions.

2.3 Modelling Dialogues using Attributed Probabilistic Finite State Bi-Automata

As explained before, a dialogue can be seen as an stochastic exchange of user and system responses. As Figure 2.2 shows, two parallel sequences are generated throughout the dialogue: the responses given by the user u_t with their associated decoding d_t and the responses given by the system a_t.

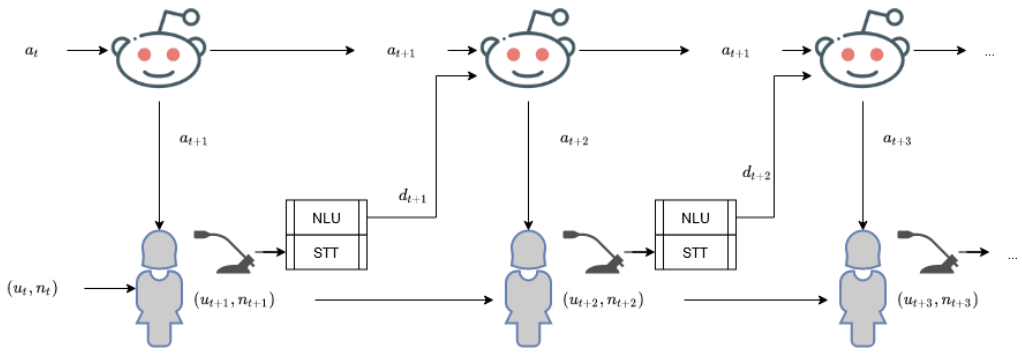


Fig. 2.2.: Dialogue interaction as an exchange of responses d_t and a_t between a user and a system

Then, the dialogue can be represented as a *bi-string* and stochastic regular bi-language (M Inés Torres and Casacuberta, 2011). Let Σ and Δ be two finite alphabets and $\Sigma^{\leq m}$ and $\Delta^{\leq n}$, the finite sets of sequences of symbols in Σ and Δ of length up to m and n. Then, let Γ be an extended alphabet $\Gamma \subseteq (\Sigma^{\leq m} \times \Delta^{\leq n})$ consisting of pair of strings of the sequences $\Sigma^{\leq m}$ and $\Delta^{\leq n}$. Then, a bi-language is a set of strings over the extended alphabet Γ (M Inés Torres and Casacuberta, 2011).

The transitive content (e.g. bus departure, desired food and selected flight) of the dialogue is encoded in the attribute alphabet $\omega \in \Omega$. These attributes are inferred from the bi-string representation of the dialogue $\mathbf{z}$ at each turn.

The A-PFSBA formulation aims at maximizing the probability of model M to generate a given sample of dialogues Z, being $\mathbf{z}$ each of the dialogues that compose sample Z.

$$\hat{M} = \arg\max_M P_M(Z) = \arg\max_M \prod_{z \in Z} P_M(\mathbf{z})$$

As the model learns its structure by maximizing the likelihood to fit the samples, it can also generate dialogue samples, thus resulting in a generative model that encompasses both user and system interactions simultaneously.

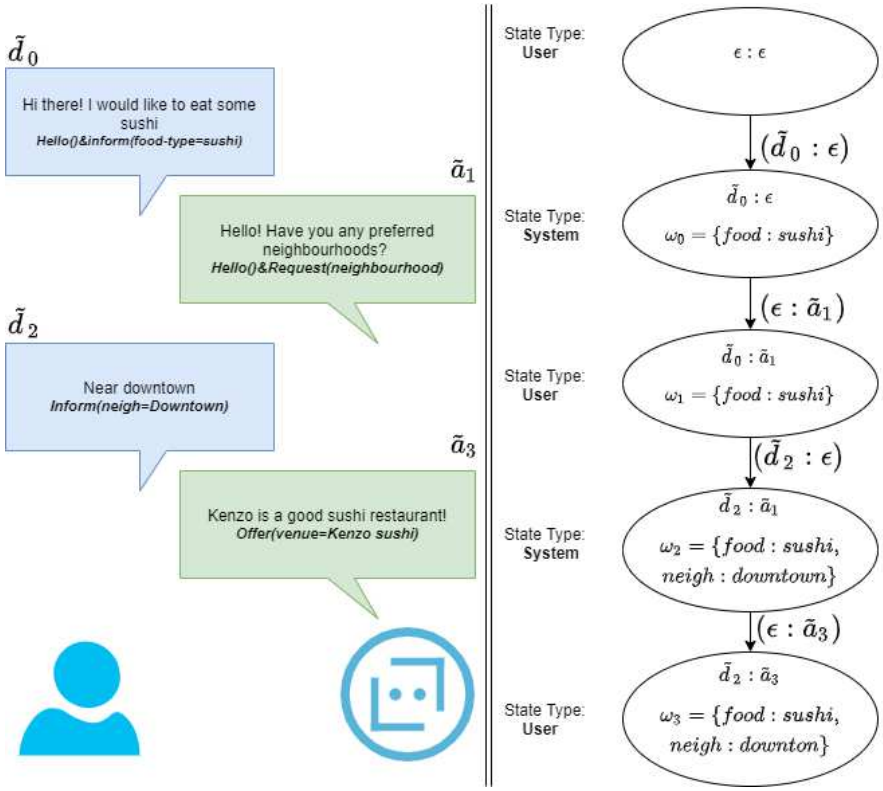


Fig. 2.3.: A-PFSBA model example of a simple dialogue example

The original A-PSFBA model presented in (M Inés Torres, 2013), is defined as $\hat{M} = (\Sigma, \Delta, \Omega, \Gamma, Q, \delta, q_0, P_f, P_t)$ where:

- Σ is the alphabet of user decoded actions, $d \in \Sigma$.
- Δ is the alphabet of system actions, $a \in \Delta$.
- Ω is the alphabet of dialogue attributes $\omega_i \in \Omega$.
- Γ is an extended alphabet $\Gamma \subseteq (\Sigma^{\leq m} \times \Delta^{\leq n})$ that contains the combination of the user decoded and the system actions.
- $Q = Q_S \cup Q_U$ is the set of system and user states labelled by bi-strings and attributes: $[(d_i : a_i), \omega_i] \in \Gamma \times \Omega$.
- $q_0 \in Q_S$ is the unique initial state: $[(\epsilon : \epsilon), \epsilon]$ where ϵ is the empty symbol.
- $\delta \subseteq Q \times \Gamma \times Q$ is the union of two sets of transitions $\delta = \delta_S \cup \delta_U$ as follows:
  - $\delta_S \subseteq Q_S \times \Gamma \times Q_U$ is the set of system transitions of the form $(q, (\epsilon : a_i), q')$ where $q \in Q_S,\ q' \in Q_U$ and $(\epsilon : a_i) \in \Gamma$.
  - $\delta_U \subseteq Q_U \times \Gamma \times Q_S$ is the set of user transitions of the form $(q, (d_i : \epsilon), q')$ where $q \in Q_U,\ q' \in Q_S$ and $(d_i : \epsilon) \in \Gamma$.
- $P_f : Q \rightarrow [0, 1]$ is the final-state probability distribution.
- $P_t : \delta \rightarrow [0, 1]$ defines the transition probability distributions $P_t(q, b, q') \equiv P_t(q',\ b \mid q)\ \forall b \in \Gamma$ and $q, q' \in Q$ such that:

$$P_f(q) + \sum_{b \in \Gamma, q' \in Q} P_t(q, b, q') = 1\ \forall q \in Q$$

  where transition (q, b, q') is completely defined by the initial state q and the transition action b. Thus, $\forall q \in Q,\ \forall b \in \Gamma,\ |\{q' : \{(q, b, q')\}| \leq 1$

Then, over this bi-language notation, any two-party dialogue $\mathbf{z}$ can be viewed as a sequence of user and system actions (M Inés Torres, 2013). Let $d \in \Sigma$ be the finite alphabet of user actions decoded by a NLU component and $a \in \Delta$ the finite alphabet of system actions. Then, $\tilde{d}_i = d_1, \cdots, d_{|\tilde{d}_i|} \in \Sigma^{\leq m}$ represents the decoding of a user input utterance, where each item of the alphabet d_i can be a user dialogue act. Similarly, $\tilde{a_i} = a_1, \cdots, a_{|\tilde{a_i}|} \in \Delta^{\leq n}$ represents the system response. Then, a dialogue

$\mathbf{z}$ can be represented as a bi-string over the extended alphabet $\Gamma \subseteq (\Sigma^{\leq m} \times \Delta^{\leq n})$ where the turn t, z_t is of the form $(\tilde{d}_t : \epsilon)$ for the user turns and of the form $(\epsilon : \tilde{a}_t)$ for the system turns, being ϵ the empty symbol. Finally, every dialogue bi-string $\mathbf{z}$ is modelled as a sequence of dialogue states $(q_0, q_1, ..., q_{|z|}) \in Q$ where q_0 is the initial empty state. Figure 2.3 shows the A-PFSBA model of a simple dialogue example.

2.3.1 Smoothing the model

As the A-PFSBA model $\hat{M}$ is fit to the training data, it may happen that the dialogue state leads to an unseen dialogue-state $q' \notin Q$ when used to model interactions. In this situation, depicted in Figure 2.4, the model is unable to find a possible transition and the dialogue is broken. In order to ensure the generalisation of the model, a smoothing strategy must be defined to rectify the dialogue, avoiding the interaction breakdown and continuing with it.

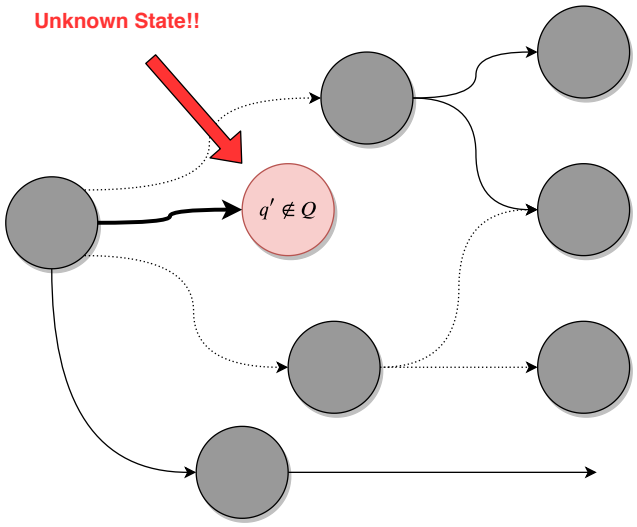


Fig. 2.4.: Dialogue interaction reaching an unknown state q'.

Under the assumption that "*similar dialogue states will trigger similar responses*", smoothing can be done in different ways. The most common method is to redirect the current dialogue state to a similar one seen in the training set $q \in Q$ and use this state to select the next dialogue action. Model smoothing is a cornerstone of the A-PFSBA formulation and the contributions made in this area are presented in Chapter 4.

2.3.2 Integrating external knowledge

A common need when deploying DSs in production environments is to communicate with already existing knowledge bases and services (e.g. databases, recommender systems, and user profiling systems) and inject information of these external services to the DM (Paek and Pieraccini, 2008). It is common for DS-building frameworks to provide mechanisms to communicate with external services and allow data-injection (Janarthanam, 2017; Khatri et al., 2018; Bocklisch et al., 2017; Cañas and Griol, 2020)

Within the A-PFSBA framework, the attribute alphabet Ω is used to integrate external knowledge. In this way, external information can be taken into account for decision making. Formally, external knowledge sources can inject data into a subset of the attributes $\Omega_e \subseteq \Omega$.

The communication mechanism is simple: the DM sends information to the services and the services return an updated response that is embedded in the attributes of the dialogue state. This method can be carried out at three different stages:

- **Upon initialisation**: when the first user decoded utterance d_0 is reached. This stage is commonly used to load user-related information such as profiles.

- **At turn start**: every time a user decoded utterance d_i is reached. This stage is commonly used when the information to consume changes according to user input.

- **At turn end**: every time a system response a_j is returned to the user. This stage can be used when DM responses trigger external actions. For example, in multimodal DSs where the DM controls augmented reality animations.

The DS applications implemented and presented in Chapter 7 employ this mechanism in order to integrate external knowledge and adjust decision-making.

2.3.3 A-PFSBA Framework Differences with Other Approaches

In comparison with the previously presented theoretical frameworks, the A-PFSBA formulation has some differences:

User decoding d_t is explicitly encoded: the transition edges of the A-PFSBA employ the hypothesis decoded by the NLU module d_t in order to perform a transition to the next state. Every composition of the items of the user alphabet $d \in \Sigma$ are thus a valid item for the bi-language. As a result, the dialogue state encodes both decoded user actions and system responses jointly, which has both beneficial and negative effects. On the one hand, the user journey can be used to estimate its behavior observing the system responses for policy making. Also, there is no need to maintain a computationally expensive belief state, since STT uncertainty is encoded directly. On the other hand, small perturbations on user input lead the system to different states, augmenting the scarcity of the model. And scarce models lead to situations where the interaction reaches unknown states $q' \notin Q$, so robust model smoothing strategies are needed.

Unknown state handling: As the A-PFSBA states are implicitly defined once the Δ, Σ, Ω alphabets are set and the user decoding is explicitly encoded, it is common to face new states that are not explicitly modelled during the training phase. In (PO)MDP structures these unknown states are usually handled by employing a fallback action (Milhorat et al., 2019; Paek and Pieraccini, 2008). In DL/end-to-end approaches, every dialogue sequence is approximated by the internal weights of the network so the model will give a response to any situation, correct or not. Within the A-PFSBA framework, dialogue breakdown is easily detected when some unknown state $q' \notin Q$ is met. Then, the dialogue smoothing strategy used to select the next response can be tailored by employing spatial relations over the A-PFSBA structural model as explained in Chapter 4, without interfering with the regular dialogue policy Π.

Task-oriented policy learning: usually in (PO)MDP the transition probability matrix encodes the probability distribution of transitioning from one state to another observing the action. This leads to the theoretical options of: (a) all states can be connected with each other; and (b) for each state, every action can be a potential candidate. The associated complexity negatively impacts scalability, performance and optimal policy learning and reward strategies usually need to be employed leading to exploration/exploitation issues. To solve this problem, simulated UMs are usually employed in order to generate synthetic dialogue samples to learn the optimal policy by means RL / Dynamic Programming (Schatzmann, Weilhammer, et al., 2006; Eshghi et al., 2017; Gašić, Jurčıček, Keizer, et al., 2010; Shah et al., 2018; H. Chen et al., 2017). The Q-learning RL method has also been extrapolated to DL/end-to-end architectures for proper dialogue policy learning (Cuayáhuitl, S. Yu, et al., 2016; Cuayáhuitl, 2017; Z. Lipton et al., 2018; Inigo Casanueva et al., 2018; Budzianowski et al., 2017). The main issues with this approach are: (1) the need to build an environment to generate the dialogue samples for RL (i.e., a UM);

(2) that the policy is adjusted to the simulated UM and, therefore, might not be extrapolated to real users.

In the A-PFSBA formulation, both the transitions between states and the possible output actions are set during the structural learning-phase of the model. Then, model exploitation or policy Π definition is completely separated from structural learning. Consequently, applicable exploitation policies range from simple decision-making strategies (e.g., randomly sampling the next system action from the transitions) to more complex approaches (e.g., reinforcement-learning, response compositions). In addition, separating structural model learning from policy exploitation allows the A-PFSBA framework to employ hybrid policies that combine rule-based and data-driven methods as shown in Chapter 6. This can help reduce the complexity of the policies to be employed, allowing the A-PFSBA framework to be used in low-resource scenarios (e.g., industry, under-resourced languages, highly-technical domains, etc.).

Incremental learning: adjusting the initial DM model to new information without having to fully retrain the model (i.e., using all the training samples again) is a desirable property. Rule-based DMs require manual adjustments each time a new response/management rule needs to be integrated. On the other hand, (PO)MDP-based DMs that employ RL to define their policies require large amounts of data derived from UMs (Gasic et al., 2008; Jurčıček et al., 2011). This approach makes it intractable to incrementally learn the dialogue policy as several samples are needed to adjust it to new dialogue states and actions (Gašić, Breslin, et al., 2013). To overcome this problem, the combination of a Gaussian Processes-based RL and a dynamic Bayesian network update method was proposed to mitigate the requirements of previous policy-adaptation approaches (Gašić, Breslin, et al., 2013; Gašić, Jurčıček, Thomson, et al., 2011). Nevertheless, this online policy adaptation method still requires a few hundred dialogue samples (400-600) to achieve the consistency of a policy trained using a UM and RL and conditions the policy to be employed by the DM.

In DL/end-to-end architectures, either when the DM model is trained as a sequence-to-sequence/discriminative network or using Deep RL, incremental learning poses several challenges related to updating the initial model while maintaining representation spaces, learning the new representations and avoiding catastrophic forgetting (Goodfellow et al., 2014; Kirkpatrick et al., 2017; Lee et al., 2017; W. Wang et al., 2019; Greco et al., 2019).

Within the A-PFSBA framework, dialogue states are defined as a composition of user, system and attribute alphabets Σ, Δ, Ω and, thus, new states can be added incrementally to the initial structure without the loss of the previous states. In

addition, these new states do not necessarily have an impact on the A-PFSBA exploitation policy due to the disassociation between structural model and policy learning. This disassociation allows to incrementally learn an A-PFSBA model for DM following a simple and effective method that does not condition the exploitation policy as described in Chapter 6.

Low-resource settings: due to the high demand of DSs, it is common to face low-resource scenarios, specially at industrial level and for under-resourced languages. In these situations, rule-based DSs have thrived due to the considerable effort that generating dialogue training data requires (Bohus and Rudnicky, 2005b; Paek and Pieraccini, 2008; Rieser, 2008). To overcome this limitation, UMs have been widely used to bootstrap initial data-driven DSs (Eckert et al., 1997; W. Wang et al., 2019; Schatzmann, Weilhammer, et al., 2006). The latest trends on end-to-end DS consist in transferring knowledge from DL models trained on wide datasets involving several tasks to a target domain containing just a few available training samples (Shalyminov et al., 2019b; Shalyminov et al., 2019a). Despite their promising results, these approaches are still in their initial stages.

All these data-driven methods still require either a UM (that needs to be handcrafted per task and/or trained over annotated dialogue samples) or an existing dataset from which to transfer knowledge to the target domain, which can be difficult to obtain in industrial settings and for under-resourced languages.

Thanks to the disassociation between the structural and policy learning of the A-PFSBA, its model structure can be trained over just a few dialogue samples and then simple policies can be applied to exploit it. In addition, if necessary, the A-PFSBA structure can also be combined with rule-based policies to overcome limitations related to the lack of initial training data. These properties make the A-PFSBA framework applicable in industrial and low-resource scenarios.

2.4 Simulated User Models

Developing task-oriented DSs using data-driven approaches usually requires high amounts of dialogue samples from which DM can learn optimal strategies. To avoid manual compilation and labelling of dialogue samples, a common approach is to develop a simulated UM that mimics the behavior of real users from a small amount of annotated dialogue corpora. UMs are also used to evaluate DM performance and/or for policy optimisation using RL (Schatzmann, Weilhammer, et al., 2006; Eshghi et al., 2017; Gašić, Jurčıček, Keizer, et al., 2010; Serras, Marıa Inés Torres, et al., 2017; Shah et al., 2018; H. Chen et al., 2017). UMs are expected to maintain coherence throughout the dialogue and to imitate the behavior of real users. In

addition, they must also have some degree of variability in order to generate unseen or unlikely interactions.

Multiple methodologies to build UMs have been proposed in the literature. Initial approaches (Eckert et al., 1997; Levin et al., 2000; Pietquin, 2005) used N-grams, but the resulting models were not capable of capturing the dialogue history and, thus, lacked coherence. With the aim of generating more coherent dialogues, several stochastic approaches such as Bayesian Networks (Pietquin and Dutoit, 2006) and networks of Hidden Markov Models (Cuayáhuitl, Renals, et al., 2005) have been explored. Similar to these graph-based models, the A-PFSBA formulation is also suitable for building UMs, as it encodes both system and user transitions over the dialogue interaction (Orozko and M Inés Torres, 2015; Ghigi and M Inés Torres, 2015; Serras, Marıa Inés Torres, et al., 2017; Serras, Marıa Inés Torres, et al., 2020).

Another popular statistical UM is the Hidden Agenda model (Schatzmann, Thomson, Weilhammer, et al., 2007), in which the user goal is predefined as an agenda of constraints and pieces of information to be requested to the system and updated at each dialogue turn. Other approaches have exploited the analogies between user simulation and imitation learning using inverse reinforcement learning (Chandramohan et al., 2011). Recently, neural network approaches have been proposed for user simulation (Layla et al., 2016; Crook and Marin, 2017; Serras, Marıa Inés Torres, et al., 2019b), which have shown the ability to account for both dialogue history and user goal.

The A-PFSBA formulation has also been used to build UMs. The User Models built previous to this thesis were mirroring systems of a DM (Orozko and M Inés Torres, 2015; Ghigi and M Inés Torres, 2015), which employ user dialogue states and transitions to emulate the user. Throughout this thesis the capability of the A-PFSBA to build UMs is further tested and evaluated.

2.5 Dialogue Systems for Industrial Applications

When it comes to deploying DS into production environments, there are several known commercial solutions such as Siri, Cortana, Google Now, Alexa and others already exist [2]. Although the inner functionality of most commercial systems is unknown due to industrial secrecy, they are usually based on expert rules, ontologies and custom backends in order to manage dialogue interaction (Wessel et al., 2019; Janarthanam, 2017; Paek and Pieraccini, 2008).

[2] one may refer to `https://botlist.co/` to find a wide variety of chatbots for different use cases

In recent years, several Alexa Challenges have been organised (Ram et al., 2018; Khatri et al., 2018) in order to bridge the gap between the technological development of DS and industrial-scale applications. The systems proposed in the challenges employ a mixture of rule-based dialogue management with an ensemble of task-specific modules or skills (Curry et al., 2018; Papaioannou et al., 2017; Khatri et al., 2018). In addition, some systems are enhanced with failsafe approaches such as response filtering mechanisms, given their public exposure (Curry et al., 2018).

Nevertheless, the lack of data and the difficulties posed by the specific business rules, back-end integrations and dynamics of each application domain make of data-driven pipelines a distant milestone. As a result, the most popular commercial frameworks still ask developers to build rules, define dialogue flows, generate annotated data and/or handle dialogue management as an external module (Cañas and Griol, 2020; Janarthanam, 2017; Bocklisch et al., 2017).

Other desirable properties of industrial and production DS setups include the ability to handle unknown situations and bootstrap initial systems for maintenance and improvement. The next sections overview how existing frameworks address these problems, are both common and relevant in industrial applications.

2.5.1 Handling unknown situations

Once DSs are deployed and face new interactions, it is common to come across circumstances where dialogues lead to unexpected scenarios. In these situations, the main objective of every DM is to provide an appropriate response that does not induce a dialogue breakdown (Higashinaka et al., 2016; Bohus and Rudnicky, 2005a).

The usual strategy to deal with dialogue breakdowns is to employ error handling techniques such as predefined fallback actions (e.g. "*Sorry, could you rephrase that?*") (Milhorat et al., 2019; Paek and Pieraccini, 2008) or error recovery strategies that narrow down user responses to expected options (Bohus and Rudnicky, 2005a). Rule-based systems and MDPs usually employ this technique. On the other hand, POMDPs require the use of external means to identify unknown situations or the definition of specific fallback dialogue-states, since their belief state will always have a candidate state with higher probability. In DL approaches, every dialogue sequence is approximated by the internal weights of the network so the model is able to give response to any situation, correct or not. In order to avoid dialogue interruptions due to inappropriate system responses, different methods such as Bi-LSTM (Xie and Ling, 2017) and SVM (Lopes, 2017) have been explored.

Within the A-PFSBA framework, a dialogue smoothing strategy (Orozko and M Inés Torres, 2015; Ghigi and M Inés Torres, 2015) similar to the back-off smoothing strategy used in machine translation and speech recognition (Pérez et al., 2008; I. Torres and Varona, 2001) has been employed. Such dialogue smoothing strategy is triggered when an unknown dialogue state $q' \notin Q$ is reached. Then, a similar known state is used to continue the interaction. This is done in two steps:

1. The state most similar to the unknown dialogue state is sampled.

2. The actions that depart from the sampled state are used to continue the interaction.

Using this technique, the DM tries to recover from breakdowns in a subtle way.

2.5.2 Hybrid approaches and incremental learning

When facing real-world scenarios, the task-oriented DSs need to model a set of business-rules (e.g. buying a cinema ticket) according to some knowledge (fares, available movies, offers) and so on. These business rules tend to change over time, thus, the DS should change accordingly, i.e. the maintenance and improvement over an initial DS should be continuous. This is usually done manually, with its associated cost. In this scenario, exploiting data-driven approaches to leverage an initial rule-based DS is a desired property which could help to reduce the manual effort required to maintain and update DSs.

Efforts have been made to hybridise rule-based and data-driven paradigms. The work presented in (Williams, 2008) combines a POMDP-based DM with a rule-based DM running in parallel. In this setup, the rule-based system samples a set of possible actions, limiting the decision space of the POMDP, which selects the corresponding system response from this set of actions. Other works (Lison, 2015; Lison and Kennington, 2016) present hybridization techniques using probabilistic rules which encode dialogue states by using a Bayesian Network. In order to encode the dialogue state transitions of the network, a set of probabilistic rules is used to encode expert knowledge in the dialogue graph structure. A similar graph-based approach is presented in (Yoshino et al., 2013), where a graph-based model of a rule-based DM is extracted and combined with a POMDP-based optimization by using a simulated UM.

Learning to incrementally adjust the DM online to new interactions is also a desirable property of DSs, although it may be unfeasible depending on the framework

employed. While rule-based systems require manual adjustments, POMDP based architectures can adjust their policy online by using Gaussian Processes and Bayesian policies (Gašić, Breslin, et al., 2013; Gašić, Jurčıček, Thomson, et al., 2011). DL approaches have explored incremental DS learning by using human-in-the-loop architectures and uncertainty-detection techniques in order to filter-out responses which may cause a dialogue breakdown (W. Wang et al., 2019). In these situations, human operators would intervene and select the correct system answer, that is used to update the model. However, incrementally updating DL model weights while maintaining representation spaces and catastrophic forgetting (Goodfellow et al., 2014) pose serious challenges that still don't have clear solutions (Kirkpatrick et al., 2017; Lee et al., 2017; W. Wang et al., 2019; Greco et al., 2019). Finally, initial work on A-PFSBA demonstrated that the model is capable of performing structural updates in a turn-by-turn basis (Orozko and M Inés Torres, 2015). This approach is further explored in this thesis in Chapter 6.

3 Tasks and Model Building

In this chapter, the corpora and models used throughout this thesis are presented.

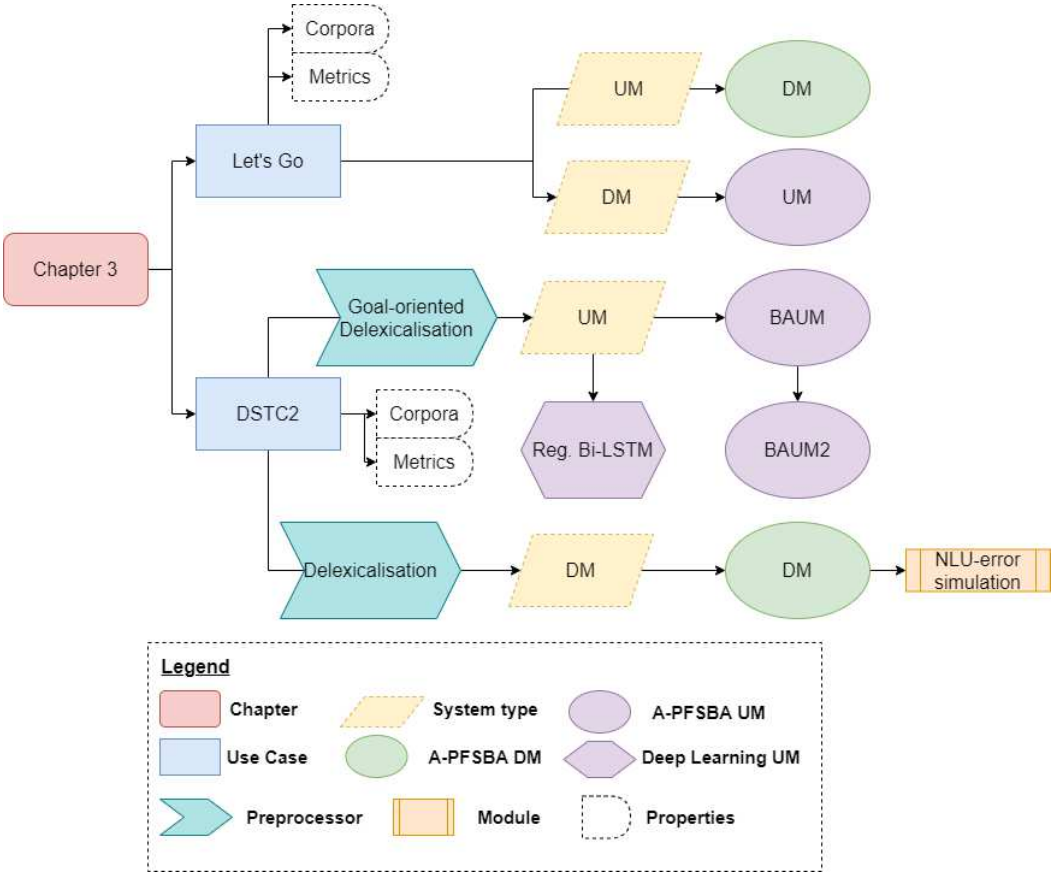


Fig. 3.1.: Chapter 3 structural map

As depicted in the map of Figure 3.1, the current chapter presents two different use case scenarios: Let's Go and Dialogue State Tracking Challenge 2 (DSTC2). For each use case, the main characteristics of the corpus and the evaluation metrics are described. In terms of the models built, for the Let's Go use case an Attributed Probabilistic Finite State Bi-Automata (A-PFSBA) based User Model (UM) and Dialogue Manager (DM) are presented. For the DSTC2 use case, a delexicalisation method is presented adjusted to each system type (UM and DM respectively). In terms of UMs over the DSTC2, two are presented: one based on Deep Learning (DL) and another one based on A-PFSBA. The Bi-Automata User Model (BAUM), is enhanced into BAUM2 to allow for more complex interactions. Then, an A-PFSBA

based DM is introduced, with a custom Natural Language Understanding (NLU) error simulation module.

These models will be used at the experimental sections of Chapters 4, 5 and 6.

3.1 Use Case: Let's Go!

The Let's Go corpus was used in previous works to build un-attributed PFSBA DM, develop preliminary path-based exploitation policies and explore initial versions of structural incremental learning methodologies (Orozko and M Inés Torres, 2015; Ghigi and M Inés Torres, 2015). In this dissertation, it has also been used to build and evaluate an attributed version of the PFSBA DM, implement multiple A-PFSBA exploitation policies and build an incremental learning mechanism to improve the A-PFSBA structure.

3.1.1 Corpus Description

The Let's Go Spoken Dialogue System (SDS) developed by Carnegie Mellon University (CMU) exploits the Olympus architecture using RavenClaw (Bohus and Rudnicky, 2003; Bohus and Rudnicky, 2005a; Bohus and Rudnicky, 2009) as DM to provide schedule and route information about the city of Pittsburgh bus service to the general public. The corpus linked to this SDS was collected from real user interactions in 2005, so events like unexpected dialogue closing, spontaneous talking, sudden noise etc. are observed. Some of the corpus statistics are shown in Table 3.1.

In the corpus, the decoding of user utterance transcriptions is carried out using the CMU Phoenix Parser (Ward, 1990), a context-free grammar parser similar to other parsing methods used for NLU (Kaiser, 1999; Y.-Y. Wang, 1999). Using the Phoenix representation, each user action $d \in \Sigma$ is represented by a string. System actions $a \in \Delta$ are also represented as strings. Attributes are discrete values related to bus schedule information, which need to be maintained throughout the dialogue. Table 3.2 shows some dialogue formatting examples.

Tab. 3.1.: Main features of the Let's Go Corpus

Let's Go Corpus Statistics					
Dialogues	1840	System Turns	28141	System Dialogue Acts $\|\Delta\|$	49
Attributes	14	User Turns	28071	User Dialogue Acts $\|\Sigma\|$	138

Tab. 3.2.: Let's Go Dialogue Formatting Example

$q = [(\tilde{d}_i : \tilde{a}_i), \tilde{\omega}_i]$	System Actions and User Decodings
$q_0 = [(\epsilon : \epsilon), \epsilon] \in Q_S$	S: Welcome to the CMU Let's Go bus information system. To get help... $\tilde{a_1}$ =inform_welcome,inform_get_help,request_query_departure_place
$q_1 = [(\tilde{a_1} : \epsilon), \epsilon] \in Q_U$	U: I'm leaving from CMU. $\tilde{d_1}$ =inform_departure_place, PlaceInformation_registered_stop $\omega_0 = \{\}$
$q_2 = [(\tilde{a_1} : \tilde{d_1}), \omega_0] \in Q_S$	S: Departing from <query.departureplace CMU>. Did I get that right? $\tilde{a_2}$ =Explicit_confirm, request_query_departure_place $\omega_0 = \{\}$
$q_3 = [(\tilde{a_2} : \tilde{d_1}), \omega_0] \in Q_U$	U: Yes. $\tilde{d_2}$ = Generic_yes $\omega_1 = \{departure.place : known\}$

3.1.2 A-PFSBA Dialogue Manager and User Model

The main objective of the interaction carried out between the user and the system in the Let's Go use case scenario is to obtain information about bus schedules. So, the main goal of the DM in this corpus is to make a coherent query to the database, in order to retrieve the bus schedule information requested by the users. Aligned with the DM goal and in order to maintain the memory of the dialogue, the attributes shared by the DM and the UM are defined in Table 3.3:

Tab. 3.3.: Attributes for the Let's Go A-PFSBA structural model

Attribute Name	Attribute Value
Departure place	1 if the departure place name is known, else 0
Arrival place	1 if the arrival place name is known, else 0
Leaving travel	1 if the travel time for leaving is known, else 0
Route number	1 if the route number is known, else 0
Departure stop name	1 if the departure stop name is known, else 0
Departure time	1 if the departure time is known, else 0
Arrival stop name	1 if the arrival stop name is known, else 0
Travel departure by	1 if the approximated departure time is known, else 0 (e.g. user says "I want to travel by 4 p.m.").
Travel arrival by	1 if the approximated arrival time is known, else 0. (e.g. user says "I want to arrive at downtown by 4 p.m.")
Neighbourhood	1 if the leaving neighbourhood is known, else 0
Neighbourhood covered	1 if the neighbourhood is covered by the system, else 0
Route covered	1 if the route is covered by the system, else 0

Note that the defined attributes have binary 1/0 values, because the NLU module is grammar-based so the users' decoding $\tilde{d}$ does not have a probability score.

Attribute iteration rules have been crafted manually: when the user communicates any of the slot values associated with an attribute (e.g. the name of a departure place), the corresponding attribute is activated. In order to make dialogue interaction faster, explicit confirmation is not required from the user.

Since every defined attribute has a related slot, attributes can be inferred directly from ongoing dialogue. As a result, the A-PFSBA structure trained on this corpus can and has been used both as DM and as UM.

3.1.3 Evaluation Metrics

When evaluating the performance of task-oriented Dialogue Systems (DSs) one of the most common metric is the Task Completion (TC) metric (Walker et al., 1997; Raux, Langner, et al., 2005; Iñigo Casanueva et al., 2017). Informally, the TC is an automated proxy that checks that the DS is correctly fulfilling the task it is designed for. These metrics can be Boolean (True if satisfied False if not) or continuous ones, which may return a score or reward. Usually the TC metrics are evaluated over a whole dialogue session.

Previous works on the Let's Go use case scenario employed Task Completion (TC) and Average Dialogue Length (ADL) in order to evaluate dialogue success (Orozko and M Inés Torres, 2015; Ghigi and M Inés Torres, 2015). The constraints of the used TC metric were:

1. The dialogue needed to last 3 turns at least.

2. A query to the database had to be done.

Unlike in (Raux, Bohus, et al., 2006), for the current experimentation there was no access available to the back-end system. Therefore, it was not possible to explicitly check if the query performed to the back end was complete. Instead, in the experiments performed over the Let's Go in this dissertation a TC metric that satisfies not only that there is a back-end lookup, but also that there is enough information to actually carry out such lookup is implemented. The implementation of such TC metric is described in Algorithm 6.

Algorithm 1 Task Completion

```
dialogue ← Dialogue to evaluate
if dialogue.length < 3 then
    return False
end if
Arrival_info = check_arrival(dialogue)
When_info = check_when_time(dialogue)
Request_Next_Bus = check_next(dialogue)
Is_query_to_db = check_query(dialogue)
Is_Info = Departure_info and Arrival_info and When_info
if Is_query_to_db is False then
    return False
end if
if (Is_Info or Request_Next_Bus) is True then
    return True
end if
return False
```

As it can be inferred from Algorithm 6, the TC evaluation metric employed in our Let's Go experiments checks that: (1) the dialogue lasts at least 3 turns; (2) a query is performed to the back-end database; and (3) the departure, destination and time information necessary to assert that the query to the back end is valid.

3.2 Use Case: Dialogue State Tracking Challenge 2

The DSTC2 corpus has not been used previously to build A-PFSBA based DMs. The main benefit of this corpus for experimental purposes is that the used NLU module is statistical and returns N-hypotheses, so the DM have to deal with the associated uncertainty. Also, user goals are explicitly annotated because paid volunteers were used to generate the corpus instead of real users, which allows the development of a more coherent UM and a more exhaustive evaluations in terms of TC. In this dissertation, the DSTC2 corpus has been used to build goal-oriented UMs and to test the behavior of the A-PFSBA based DM under uncertainty, simulating the impact of the Speech To Text (STT) module at NLU level.

3.2.1 Corpus Description

The second edition of the Dialogue State Tracking Challenge series (Henderson, Thomson, and Williams, 2014) focused on tracking the dialogue state of a SDS in

the Cambridge restaurant domain. For such purpose, a corpus[1] with a total of 3235 dialogues was released by the Cambridge Dialogue Systems group.

3.2.1.1 Dialogue Goal Representation

The DSTC2 corpus was gathered using paid Mechanical Turk volunteers in a semi-controlled environment, where interactions with the system were conditioned according to predefined scenarios.

At the beginning of each dialogue, a goal was given to each volunteer in order to guide their interaction with the system. This goal defined the user's preferences regarding the restaurant of interest (e.g. food type and price range) and the information to retrieve from the system (e.g. phone number and address), once a valid restaurant is found. The given goal was annotated in two ways: 1) in human readable text generated by using templates as instructions for the Mechanical Turkers; and 2) in a machine-readable JSON representing the goal. This goal representation used the corpus-dependant Agenda format (Schatzmann, Thomson, Weilhammer, et al., 2007). Goals were annotated using constraints to find a suitable venue and items to be requested from the system once a suitable venue was found, as shown in Figure 3.2.

```
"task-information": {
        "goal": {
            "text": "Task 05700: You are looking for an expensive restaurant
                     and it should serve european food. Make sure you get the phone number.",
            "request-slots": [
                "phone"
            ],
            "constraints": [
                [
                    "food",
                    "european"
                ],
                [
                    "pricerange",
                    "expensive"
                ]
            ]
        }
}
```

Fig. 3.2.: Simple goal given to a participant, with its textual description and JSON annotation using the Agenda schema

Different characteristics were combined by the developers of the DSTC2 corpus when defining the goals to test different scenarios of their DS:

[1] http://camdial.org/∼ mh521/dstc/

1. **Goal with achievable constraints:** where the constraints given are viable, i.e. there exists at least a venue in the knowledge base that satisfies the constraints. This is intended to test the most basic aspects of the dialogue system.

2. **Different information to request:** where the user is also given some explicit information to request to the system about the offered venues. The goal is to test if the system can respond to the information requested by the user, such as addresses, phone numbers and so on.

3. **Goal with impossible constraints:** where there is no venue that satisfies the given constraints. If this happens, the user is also given an additional set of achievable constraints. This is intended to test the system's capability of informing the user that those combinations of constraints cannot be satisfied.

4. **Request alternatives for the given venue:** sometimes the user is told that they must ask for alternative venues instead of accepting just the first venue that is offered by the system. This goal feature is included so the system can handle situations where multiple restaurants are available.

When using this corpus in this dissertation, the main issue has been that only conditions (1) and (2) were explicitly annotated by the corpus developers in machine readable JSON format, as it can be seen in Figures 3.2 and 3.3. On the contrary, the information about the goal of points (3) and (4) needs to be inferred from the given text. Two examples are presented below that depict this issue:


```
"task-information": {
        "goal": {
            "text": "Task 00282: You want to find a restaurant in the centre and it should serve fusion food.
                     If there is no such venue how about european type of food. You want to know the phone number and postcode.",
            "request-slots": [
                "phone",
                "postcode"
            ],
            "constraints": [
                [
                    "food",
                    "european"
                ],
                [
                    "area",
                    "centre"
                ]
            ]
        },
```


Fig. 3.3.: Double goal given to a participant, both in textual description and JSON annotation of the Agenda schema.

As it can be seen in Figure 3.3, the first constraint (a restaurant serving fusion food) is impossible to satisfy in the city centre, so an additional food constraint (European food) is given to the volunteer. This will condition they behaviour when interacting with the system, but there is no explicit annotation as machine-readable JSON.

```
"task-information": {
        "goal": {
            "text": "Task 12306: You are looking for a moderately priced restaurant and it should be in the west part of town.
                     Don't go for the first venue the system offers you, ask if there is anything else.
                     You want to know the address, phone number, and type of food.",
            "request-slots": [
                "addr",
                "phone",
                "food"
            ],
            "constraints": [
                [
                    "pricerange",
                    "moderate"
                ],
                [
                    "area",
                    "west"
                ]
            ]
        },
```


Fig. 3.4.: Single goal with request alternatives given to a participant, both in textual description and JSON annotation of the Agenda schema.

As shown in the Figure 3.4, the user is asked to look for a moderately priced restaurant in the west part of town, but is also required to ask for alternative venues to the one offered by the system, and again, this is not represented in the annotated JSON schema. In addition, this requirement is never met jointly with the double goal requirement.

As it will be described in more detail in Section 3.2.2.2, the discrepancies between the JSON annotated goal and the text given to the participants will have an impact when building UMs in our experiments, resulting in two different ones: one that only takes into account the JSON annotation; and a more complex one, which extracts information from the given texts by using regular expressions.

3.2.1.2 Taxonomy Description

The hierarchy associated with the Dialogue Act (DA) representation (e.g. the **Inform** intent has associated the *Food* slot that can have [*Chinese, Japanese,* ...] values) is often referred to as the taxonomy, which describes the semantic space of the user and the system in DSs.

Table 3.4 summarizes the user DAs of the DSTC2 corpus, together with their related slots. Note that many intents do not have related slots and that the slots of the *Request* intent have no value. Table 3.6 includes all the informable slots in the DSTC2 corpus and some examples of their possible values. In addition to those specific values, every slot has the special value *dontcare*. On the other hand, Table 3.5 summarizes the system intents and the related slots per intent. Finally, Table 3.7 includes a dialogue of the DSTC2 corpus represented using this annotation schema.

Tab. 3.4.: Dialogue Acts that the user can trigger in the DSTC2 corpus

User Intents	Related Slots
Acknolwedge	*Null*
Affirm	*Null*
Bye	*Null*
Confirm	*Area, Food, Price Range, Restaurant*
Deny	*Area, Food, Price Range, Restaurant*
Hello	*Null*
Help	*Null*
Inform	*Area, Food, Price Range, Restaurant*
Negate	*Null*
Repeat	*Null*
Request Alternatives	*Null*
Request More	*Null*
Request	*Area, Food, Price Range, Restaurant, Phone, Address, Signature, Postcode*
Restart	*Null*
Silence	*Null*
Thankyou	*Null*

Tab. 3.5.: Intents of the system in the DSTC2 corpus

System Intents	Related Slots
Affirm	*Null*
Bye	*Null*
Confirm-domain	*Null*
Negate	*Area, Food, Price Range, Restaurant*
Repeat	*Area, Food, Price Range, Restaurant*
Request-more	*Null*
Welcomemsg	*Null*
Canthelp	*Area, Food, Price Range*
Canthelp-exception	*Restaurant*
Explicit-confirm	*Area, Food, Price Range*
Implicit-confirm	*Area, Food, Price Range*
Inform	*Area, Food, Price Range, Restaurant, Phone, Address, Signature, Postcode*
Request	*Area, Food, Price Range, Restaurant*
Offer	*Restaurant*
Select	*Area, Food, Price Range*

Tab. 3.6.: Possible slot values for the *Inform, Confirm* and *Deny* intents

Informable Slots	Possible Values	Examples
Restaurant Name	113	Nandos, Pizza Hut, ...
Food Type	91	Basque, Italian, European, ...
Price Range	3	Cheap, Moderate, Expensive
Area	5	North, West, South, East, Centre

This corpus is already split into Train, Development and Test sets of 1612/506/1117 dialogues respectively and these partitions are used throughout the experimentation phases.

Tab. 3.7.: Dialogue example tagged with the DSTC2 Taxonomy

Who Is	Text Representation	Dialogue Act Representation
System	Hello , welcome to the Cambridge restaurant system... How may I help you?	welcomemsg
User	moderately priced north part of town	inform(pricerange=moderate), inform(area=north)
System	golden wok is a nice restaurant in the north of town in the moderate price range	offer(venue=Golden Wok), inform(area=north), inform(pricerange=moderate)
User	what is the address and phone number	request(address), request(phone)
System	The phone number of golden wok is 01223 350688 and it is on 191 Histon Road Chesterton	offer(venue=Golden Wok), inform(address=191 Histon ...), inform(phone=01223 350688)
User	thank you good bye	thankyou, bye

3.2.2 User Models

Two methodologies have been developed to build UMs in the DSTC2 domain, based on A-PFSBA and DL technologies respectively. Both models employ the same user and system taxonomy. In order to reduce the dimension of the DA space in the original taxonomy without losing the information required for dialogue management, a delexicalisation approach has been employed across UMs. The next sections describe the proposed delexicalisation method and the developed A-PFSBA and Deep Learning UM.

3.2.2.1 Goal-oriented delexicalisation for DSTC2

As described previously in Section 3.2.1.2, the DA representation that is used to represent the input and output space of the DM is structured in a hierarchical taxonomy. A common issue when some slots have several values is that such DA representation is highly sparse. As a result, it is difficult to capture useful patterns for dialogue management and the computational effort required by the DM models increases, which results in a negative impact on system usability.

As explained in Section 3.2.1.1, in the DSTC2 corpus each goal is represented as a set of constraints C and values to request R that the user needs to fulfill through the interaction, following the Agenda schema (Schatzmann, Thomson, Weilhammer, et al., 2007). Figure 3.2 shows a clear example of this representation where $Goal = (C, R)$ follows this tuple schema: $\{food : european,\ pricerange : expensive\}$ are the constraints to find the venue and $[phone]$ is the information to be obtained regarding the venue. The constraints and requests of the goal have a direct

correlation with the user taxonomy, conditioning the dialogue interaction logic of the user.

The proposed *goal-conditioned delexicalisation* of the DSTC2 corpus assumes that the user will be collaborative and will try to satisfy the given goal . Under this constraint, it is assumed that the specific slot values are relevant to the interaction logic if they correlate with the constraints and requests set in the initial goal. Then, every slot value is delexicalised with a *goal* or *other* tokens, depending on whether they match the given constraint values or not. Following this assumption, the possible values of the slots that are part of the set goal constraint C are highly reduced. Table 3.8 shows how the slot values of an interaction represented at dialogue act level are compressed using the proposed technique, according to the given goal.

Tab. 3.8.: DSTC2 interaction example, delexicalised according to the user goal

	Text Representation of the Goal	**Goal Agenda Representation**
Goal	*Task 01875: You are looking for a moderately priced restaurant and it should be in the north part of town. Make sure you get the address and phone number of the venue.*	*Constraints: area=north pricerange=moderate To request phone, addr*
Who Is	**Original Text**	**Goal-delexicalised Dialogue Acts**
System	Hello , welcome to the Cambridge restaurant system. How may I help you?	welcomemsg
User	moderately priced north part of town	inform(pricerange=<price-goal-1>), inform(area=<area-goal-1>)
System	golden wok is a nice restaurant in the north of town in the moderate price range	offer(venue=<venue>), inform(area=<area-goal-1>), inform(pricerange=<price-goal-1>)
User	what is the address and phone number	request(address), request(phone)
System	The phone number of golden wok is 01223 350688 and it is on 191 Histon Road Chesterton	offer(venue=<venue>), inform(address=<venue-address>), inform(phone=<venue-phone>)
User	thank you good bye	thankyou, bye

As it can be seen, goal-oriented delexicalisation has a direct impact on the DA representation, narrowing down each possible slot value. As a result, slot value level information can be included in the DA representations for user modelling purposes, avoiding excessive sparsity and with small information loss. This delexicalisation technique is used in both, the A-PFSBA based UM and Deep Learning based UM presented in the next two sections.

3.2.2.2 Bi-Automata User Model

This section describes the Bi-Automata User Model built using the DSTC2 corpus in detail. In addition to applying *goal-oriented delexicalisation* to reduce the sparsity of

the DA space, the A-PFSBA model is also explicitly conditioned to the user goal in order to guide the interaction according to a predetermined set of constraints.

Goal-conditioning the A-PFSBA

In the DSTC2 corpus, goal constraints C specify requirements of the user to achieve (e.g. restaurant type) while requests R specify desired pieces of information to retrieve (e.g. address and phone number), following the Hidden Agenda notation of (Schatzmann, Thomson, Weilhammer, et al., 2007). Both C and R are represented as slot-value pairs. In order to condition the behavior of the BAUM to a given dialogue goal, C and R are encoded as A-PFSBA attributes Ω before starting the conversation as depicted in Figure 3.5

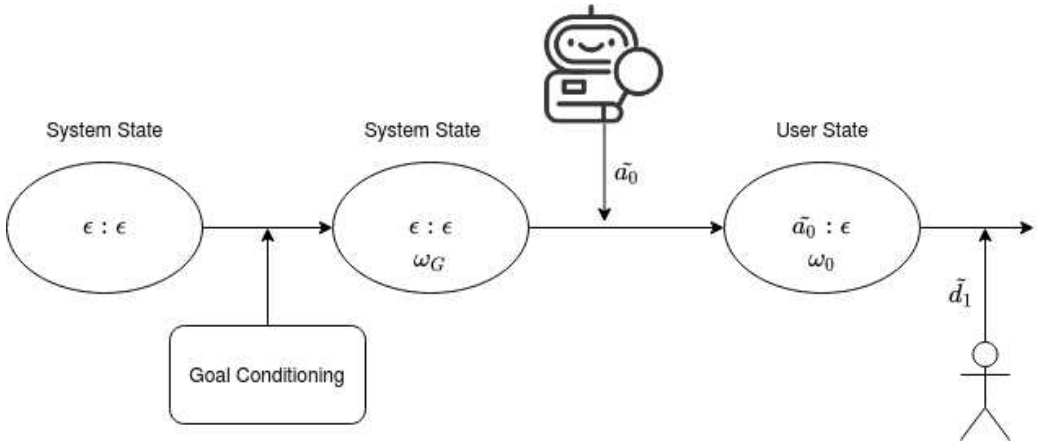


Fig. 3.5.: Goal-conditioning the interaction at dialogue initialisation

As a result, $\Omega_G \subset \Omega$ can be defined as the union of two sets of attributes $\Omega_C \cup \Omega_R = \Omega_G$, where Ω_C corresponds to the slots given as constraints and Ω_R corresponds to the slots the user has to request about. A dialogue interaction including this conditioning can be found in Table 3.9

As explained in the corpus Section 3.2.1.1, two goal representations can be found in the DSTC2 corpus: (1) an explicit one annotated in JSON format; and (2) the text given to the paid volunteers of Mechanical Turk. Based on these two different representations, two different UMs have been trained using the A-PFSBA framework. While the first UM denoted BAUM uses the JSON representation of the goal, the second one referred to as BAUM2 extracts the goal by using regular expressions from the text given to the volunteers. This BAUM2 is more complex and can handle the multi-goal scenarios 3 and 4 described in Section 3.2.1.1, such as requesting alternative venues once an appropriate one is found or changing the goal if there isn't any restaurant that matches the initial constraints of the given goal.

BAUM Attributes The dialogue attributes defined to model the BAUM are split into three categories; (1) goal related attributes, which are responsible for encoding the users' goal as shown in Table 3.10; (2) constraint communication attributes, which keep track of which user constraints are successfully communicated to the system as shown in Table 3.11; and (3) venue information retrieval attributes, which control the information the system has given regarding the offered venues as shown in Table 3.12.

Tab. 3.10.: Goal related attributes of BAUM and their values

Goal related Attributes	Values	Description
Constraint food	1,0	Food slot is set as constraint
Contraint area	1,0	Area slot is set as constraint
Constraint pricerange	1,0	Pricerange slot is set as constraint
Constraint name	1,0	Venue name is set as constraint
Request food	1,0	Food value needs to be requested
Request area	1,0	Area value needs to be requested
Request pricerange	1,0	Price value needs to be requested
Request address	1,0	Address value needs to be requested
Request phone	1,0	Phone value needs to be requested
Request signature	1,0	Signature value needs to be requested
Request postcode	1,0	Postcode value needs to be requested

Tab. 3.11.: Constraint communication related attributes of BAUM and their values

Constraint Communication Attributes	Values	Description
User informed food goal	1,0	The user informed about the given food constraint in the goal
User informed food other	1,0	The user informed about a food value different from the one given as goal
User informed area goal	1,0	The user informed about the given area constraint in the goal
User informed area other	1,0	The user informed about an area value different from the one given as goal
User informed price goal	1,0	The user informed about the given pricerange constraint in the goal
User informed price other	1,0	The user informed about a price value different from the one given as goal
User informed name goal	1,0	The user informed about the given venue name constraint in the goal
User informed name other	1,0	The user informed about a venue name value different from the one given as goal
System understood food goal	1,0	The system has understood the user food goal value
System understood food other	1,0	The system has understood the user food other value
System understood area goal	1,0	The system has understood the user area goal value
System understood area other	1,0	The system has understood the user area other value
System understood price goal	1,0	The system has understood the user price goal value
System understood price other	1,0	The system has understood the user price other value
System understood name goal	1,0	The system has understood the user venue name goal value
System understood name other	1,0	The system has understood the user venue name other value

Tab. 3.12.: Venue information retrieval related attributes of BAUM and their values

Information Retrieval Attributes	Values	Description
Received venue address	1,0	The system has informed about the current venue's address
Received venue phone	1,0	The system has informed about the current venue's addres
Received venue postcode	1,0	The system has informed about the current venue's addres
Received venue name	1,0	The system has informed about the current venue's name
Received venue food	1,0	The system has informed about the current venue's food type
Received venue pricerange	1,0	The system has informed about the current venue's price value
Received venue signature	1,0	The system has informed about the current venue's signature
Offered new venue	1,0	The system offered a new, different venue

The values of these attributes are inferred from the dialogue interaction with simple handcrafted rules that follow an *if-this-then-that* logic. These rules are described in Appendix A.1.

BAUM2 Attributes BAUM2 extends the capabilities of BAUM to explicitly capture scenarios 3 and 4 of Section 3.2.1.1, employs different goals and performs more realistic and complex interactions such as requesting for alternative venues and changing the goal if there isn't any restaurant that matches the initial one.

In order to achieve this, additional attributes are added to the BAUM attribute set. To this end, goal constraints and requests are inferred from the text templates that were received by the Mechanical Turkers by using regular expressions.

The new BAUM2 attributes, which are used to extend the BAUM ones, are shown in Tables 3.13 and 3.14.

Tab. 3.13.: Goal attributes of BAUM2 and their values

Enhanced Goal Attributes	Values	Description
Secondary food goal	1,0	There is a secondary constraint for the food goal
Secondary area goal	1,0	There is a secondary constraint for the area goal
Secondary price goal	1,0	There is a secondary constraint for the price goal
Secondary name goal	1,0	There is a secondary constraint for the name goal
Request Venue Alternative	1,0	The user has to ask for an alternative venue

Tab. 3.14.: Constraint communication related attributes of BAUM2 and their values

Enhanced Constraint Communication Attributes	Values	Description
User informed secondary goal food	1,0	The user has informed about the secondary goal of the slot food
User informed secondary goal area	1,0	The user has informed about the secondary goal of the slot area
User informed secondary goal price	1,0	The user has informed about the secondary goal of the slot pricerange
User informed secondary goal name	1,0	The user has informed about the secondary goal of the slot name
System understood food secondary goal	1,0	The system has understood the secondary goal value of the area slot
System understood area secondary goal	1,0	The system has understood the secondary goal value of the food slot
System understood price secondary goal	1,0	The system has understood the secondary goal value of the pricerange slot
System understood name secondary goal	1,0	The system has understood the secondary goal value of the name slot

Note that now, the "other" slot value refers to any slot value that is different from the first or second constraint set for that slot. The specific attribute-iteration rules are described at Appendix A.

3.2.2.3 Deep Learning based User Model

A generative concatenative DL model has been proposed to build a User Model (Serras, Maria Inés Torres, et al., 2019b) over the DSTC2 corpus in order to set a hard baseline for the initial BAUM described in the previous section.

In order to be able to compare this DL-based UM against another sequence-to-sequence DL model published in (Layla et al., 2016) was used. This UM also employs the DSTC2 corpus and the goal representation given in the JSON annotations. To perform a proper comparison, the DL-based UM described in this section was only trained over the JSON encoded goal, leaving the more complex scenarios encoded in the Mechanical Turk descriptions out.

The proposed neural network architecture consists of a generative concatenative model (Z. C. Lipton et al., 2015) with a regularization layer. It encodes the dialogue history in a sequence both forward and backward and exploits a regularization mechanism to improve generalization.

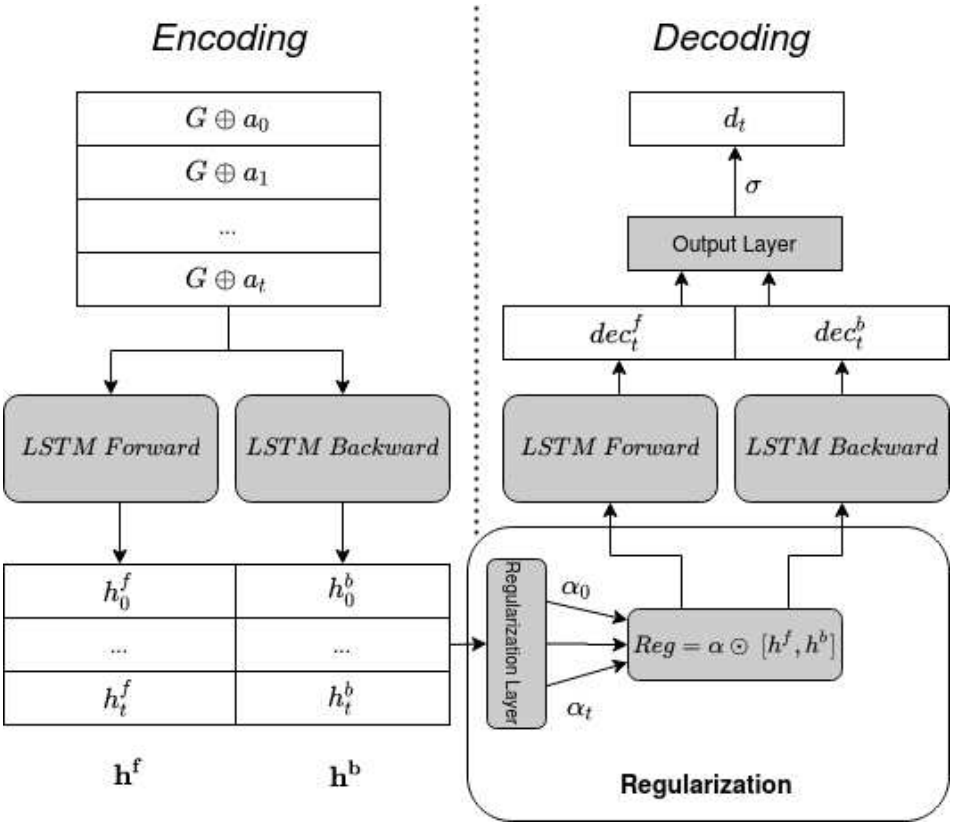


Fig. 3.6.: Neural network architecture proposed for user modeling

As shown in Fig 3.6, the input to the network is a concatenation of the user goal representation G as used in the BAUM attributes and the sequence of system dialogue acts until the current turn t:

$$A_0^t = (a_0, a_1, ..., a_t)$$

Where a_t is the system action of turn t. The user-given goal G is represented as a 1-hot encoding of the slots given as constraints C and requests R in the dialogue scenario. The output of the network is a prediction of the user dialogue act at the next turn d_t by using the sigmoid activation, so it is equivalent to a multi-regressor system. Note that under this notation, turns are made up of both system and user, so turn t is represented as (a_t, d_t). Also note that while system dialogue acts change turn by turn, the initial goal representation remains the same throughout the dialogue.

The encoding layer is made up of a bidirectional Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), whose output is the dialogue history encoded

as $\mathbf{h^f}$ forward and as $\mathbf{h^b}$ backward. The applied regularization mechanism requires to learn the weight vectors α for each row of the encoding matrix: $(\alpha_0, \cdots, \alpha_t)$

$$H = [\mathbf{h^f}, \mathbf{h^b}]$$

.

Being $H_i \; \forall \; 1, \cdots, t$ the i-th row of the encoding matrix, the vector α_i is calculated as $\alpha_i = \sigma(W_a H_i)$

Where W_a are the parameters of the Regularization Layer and σ the sigmoid function. Once H and α are known, the encoded sequence is regularized by the element-wise product as follows:

$$Reg = \alpha \odot H$$

The motivation of this operation is to override the non-relevant values of the encoded sequence.

Decoding is then applied to Reg through another bidirectional LSTM, which outputs forward and backward decoding vectors dec_t^g and dec_t^b at turn t. These vectors are finally concatenated and processed by the output layer with a sigmoid activation function, from which the user dialogue act at the current turn d_t is predicted.

The proposed model uses an expert network for every possible user intent, its slots and values as shown in Table 3.4, so the architecture in Fig 3.6 is replicated and each network learns to predict just one intent, with its slots and values. As a result, the final UM is an ensemble of models, each of which predicts the slots of a specific user intent as shown in Fig 3.7. As the sigmoid function is used in the output layer, a threshold needs to be set to decide whether a user intent needs to be applied to the UM response. To this end, an individual threshold is set for each network (e.g. θ_{inform} for the network that predicts the *inform* intent, slots and its values). These thresholds are set by performing a grid-search in the validation set to maximise the F1 score.

The final DA output d_t is the combination of all user acts given by each specialised network of the ensemble.

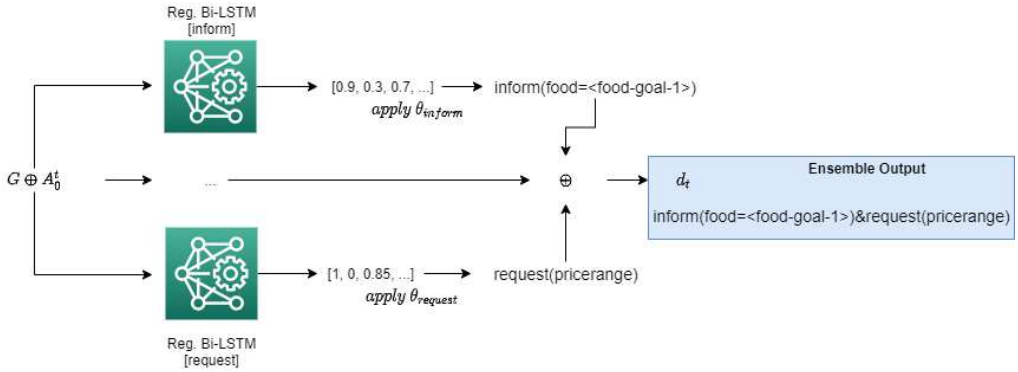


Fig. 3.7.: Ensemble of Dialogue Act networks from which d_t is predicted

3.2.3 A-PFSBA Dialogue Manager

This section describes how a DM has been built over the DSTC2 corpus using the A-PFSBA framework. The goal of this DM is to find the users' preferences regarding a venue type (i.e. food, area, price range), to offer a venue that satisfies those needs and to return venue-related information (e.g. address, phone, postcode, etc.). To put it simply, the goals the DM needs to satisfy through dialogue are:

1. Find the user preferences for a venue.

2. If the user's preferences can be met, offer a suitable venue.

3. If there is no venue that matches the user's preferences, inform about the mismatch.

4. Once a venue is identified, respond adequately to user questions regarding that venue's properties.

To that end, in addition to managing the dialogue, the DM must communicate with a Search Engine (SE) that exploits a Data Base (DB) which stores the information of available venues and their characteristics. The motivation for these modules is the dynamic nature of the domain, where the available venues can change on a daily basis, so the search logic needs to be handled by an external service. This enables to modify the information stored in the DB without rebuilding or updating the DM. As a result, the DM can employ features extracted from this external service to handle the interaction (e.g. if no venue satisfies the user's constraint, inform about that to the user).

3.2.3.1 Index Delexicalisation

As the BAUMs, the A-PFSBA DM also uses the DA representation for user decodings $d \in \Sigma$ and system actions $a \in \Delta$. Given the large amount of possible slot values in the DSTC2 corpus, data sparsity is also an obstacle that must be overcome to achieve a good modelling. In the case of the DM, sparsity is even higher than for UM, since STT-induced errors corrupt user input $d \in \Sigma$, which increases the possible slot-values that need to be handled for proper dialogue management. In order to reduce the amount of items that the A-PFSBA model needs to handle, a delexicalisation method is defined to reduce the amount of items of the alphabets Δ, Σ, which, in practice, is a transformation of the user and system dialogue acts.

Unfortunately, the goal-oriented delexicalisation method employed for the development of the DSTC2 User Models in Section 3.2.2.1 cannot be applied for DM development because the dialogue goal G is hidden from the system. Instead, delexicalisation is carried out by replacing slot values with an indexed generic token and by storing the specific slot value in a Mapping Blackboard (MB) as shown in Figure 3.8.

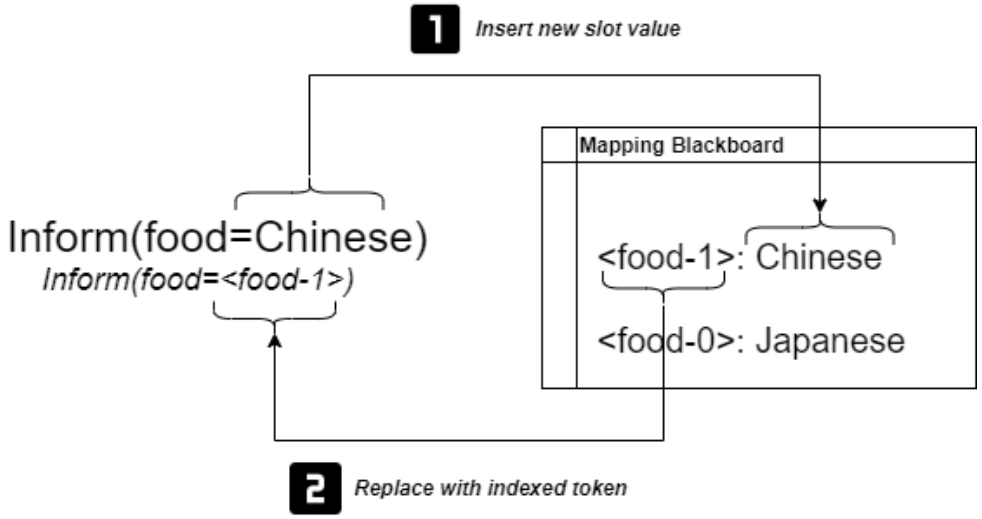


Fig. 3.8.: Example of the delexicalisation of a user-act with an indexed token by using a Mapping Blackboard

Similar to the method used at (S. Young et al., 2010; S. Young, 2006), the assumption behind this delexicalisation approach is that *the specific slot-value is irrelevant for the dialogue management logic, as decision-making is carried out depending on the results of the queries to the Search Engine in the DB*. In other words, the most relevant information for the interaction to carry on is not that the user has asked specifically about a *Chinese* restaurant, but whether the user has asked about a venue

characteristic that exists in the DB or not. This conditioning of the dialogue logic is depicted in Figure 3.9.

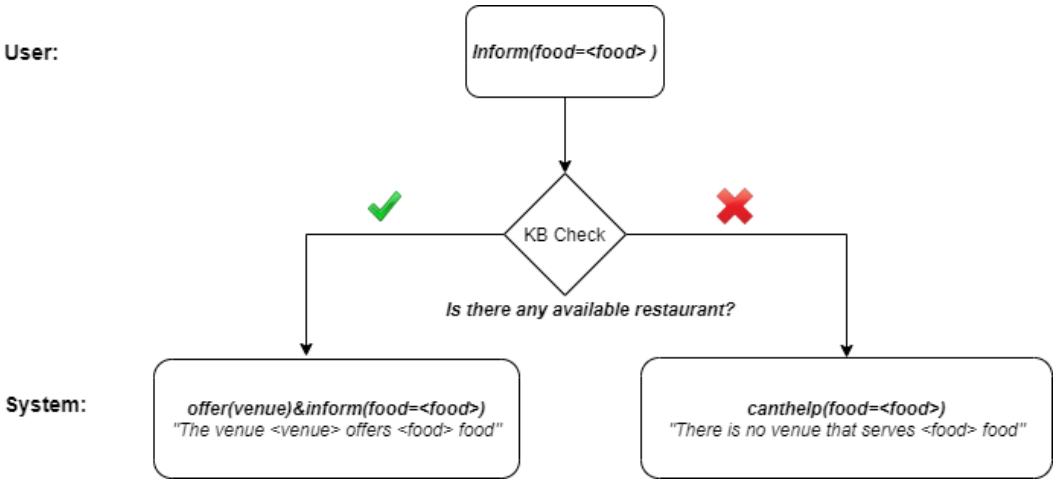


Fig. 3.9.: Example interaction where the exact slot-value is irrelevant for the dialogue logic

Note that this assumption is valid for the DSTC2 use case, because the restaurant finding domain is an information retrieval task where the interaction logic is subject to the SE and the DB status and properties.

Simple Delexicalisation has been used during the dialogue generation experiments carried out in Chapter 4 and in Section 6.5.1.

3.2.3.2 Value Ranking Delexicalisation

The Simple Delexicalisation approach employs the **appearance-order** of the slot values in order to turn them into generic tokens. This results in a static delexicalisation (e.g. the token *food-1* in Figure 3.8 will always be set to *Chinese*).

In order to better capture the interaction logic, a more dynamic delexicalisation technique referred to as Value Ranking Delexicalisation has also been implemented. Within this approach, the 5 best values of each slot that are stored in the blackboard are used for delexicalisation. These values are **ranked by their score**, so, the tokens *<food-rank-0>*, *<food-rank-1>*, ..., *<area-rank-0>* etc. are used to delexicalise specific values of the slots. Note that these tokens are static (i.e. *<food-rank-0>* will always refer to the food slot value with highest score or confidence) but the slot value that these tokens are replacing may change during the interaction, as the user can negate the current foot slot value and suggest another one.

Value Ranking Delexicalisation has been used when using handcrafted and probabilistic rules for DM policy in Chapter 6, Section 6.5.3.

3.2.3.3 Using External Services to Infer DM Attributes

As mentioned before, the dialogue interaction logic of the DSTC2 corpus is subject to the SE status. This SE is responsible for finding appropriate venues and making queries to a DB. Then, according to the information retrieved by this SE, the DM needs to adjust its strategy.

The SE performs read/write/update operations over the the MB throughout the interaction, according to a set of rules. In addition, each time the MB status changes, venue-finding queries are performed. Finally, both the information of the MB and the results received from the SE queries are used to update the attributes of the DM.

The following simple handcrafted rules are employed to modify the MB information:

- **Inform Rule:** each time the user informs about a slot value, this value is written in the memory together with its associated probability. If it is already in the MB, the probability is added up to a maximum of 1.

- **Canthelp Rule:** if the system triggers a *canthelp* action over a concrete slot (i.e. it is not possible to combine a specific slot and value such as food=Basque), this value is removed from the MB.

- **Deny Rule:** if the user denies a slot value, the probability of the deny act is subtracted from the score of the MB. If the resulting score is 0 or less, the slot value is removed.

- **Explicit confirmations:** if the user acknowledges or affirms a concrete slot and value after an explicit request or confirmation of the system, the score of the user act is added up to the one in the MB. If, conversely, the user negates, the score is subtracted.

- **Log Offered Venues:** the memory logs the different offered venues and their characteristics, so it keeps track of the available venues when the user asks about alternatives.

- **Request Alternative Venues:** when the user asks for an alternative, the current venue is flagged as offered and loses priority over the venues that have the same score.

When searching for venues, the combinatorial over the available food, area and pricerange slot values is used because multiple values may be active in the MB for the same slot. At least, information about two different slots is required to perform a search query by the SE.

Each search query returns a matching venue-cluster (i.e. the group of venues that satisfy the search constraints). Being $s(food_i), s(area_j), s(price_k)$ the scores of the MB of certain food, area and price slot-values to make some query $Query = \{food_i, area_j, price_k\}$. Then, the score given to the venue-cluster that is returned by the SE over that query is:

$$\frac{s(food_i) + s(area_j) + s(price_k)}{|Query|}$$

Note that if any score is 0 and/or the slot value is "dontcare" the item is not used to perform the query. The score of each venue is normalized by the amount of possible constraints to search for.

3.2.3.4 Inferring the Attributes

The DM attributes are updated at each turn by using the Memory Blackboard and the scored venue-clusters received from the queries performed to the Search Engine. The list of inferred attributes is the following:

- **Food maximum score**: the maximum score for the food slot-value in the MB.
- **Area maximum score**: the maximum score for the area slot-value in the MB.
- **Price range maximum score**: the maximum score for the price range slot-value in the MB.
- **Top venue score**: the maximum score of all the venue clusters.
- **Next venue-cluster score**: the second maximum score of all venue clusters.
- **Top 2 clusters score difference**: the score difference between the first and the second venue clusters.

- **Amount of clusters**: number of available venue-clusters, up to 2.

To avoid unnecessary sparsity, all attribute values are rounded to the first decimal.

3.2.3.5 NLU Error Simulation

As mentioned in Chapter 2, channel noise n_t induces errors in STT transcriptions (Schatzmann, Thomson, and S. Young, 2007a) which cause uncertainty in NLU when decoding user transcriptions $\tilde{u}_t$ into the DA $d \in \Sigma$ that compose $\tilde{d}_t \in \Sigma^{\leq m}$. As a result, the DM needs to handle these perturbations.

To simulate this type of error for the DSTC2 scenario, a statistical approach has been followed. The proposed method models the error distributions observed in the corpus for each possible user DA $d \in \Sigma$. Note that although noise is induced at STT level, the DM uses the user-language decoded by the NLU as input and, thus, the error is modelled at NLU level. In particular, the NLU error corrupts the correct user decoding $\tilde{d}_t \in \Sigma^{\leq m}$ at two levels :

1. **Intent-slot level**: at this level, the error is modelled at a coarser granularity, by storing the probability of misunderstanding the user's intent and its associated slot in a confusion matrix. For example, the confusion matrix captures the probability of misunderstanding *Inform(food=*)* with *Inform(area=*)*, the exact values of the slots *food* and *area* are ignored. A special <*empty*> value is used to depict when the correct hypothesis or dialogue act is not detected due to an STT error.

2. **Slot-value level**: here the error is modelled at a finer granularity, pinpointing cases when a specific slot value is misunderstood with another. Since there is not enough data to derive a complete confusion matrix for all the possible slot and value combinations, this confusion matrix is built using a delexicalised form of the slot-value pairs by capturing the frequency of corrupting the correct slot-value with an incorrect one (e.g. "food|<correct-value>" with "food|<incorrect-value>").

In addition, the NLU score-distribution calculated as the output NLU probability assigned to each error type is also sampled for each of the above confusion-matrices. Then, using the confusion matrices and the sampled NLU-score distributions, Algorithm 2 describes the process of perturbing the user decoding elements $d_i \in \tilde{d}_t$ given a correct user action.

Algorithm 2 NLU error simulation to corrupt the DA user decoding $\tilde{d}_t$

$IS_{cm} \leftarrow$ Intent + Slot name confusion matrix
$SV_{cm} \leftarrow$ Slot + Value delexicalized confusion matrix
$\alpha_{corrupt} \leftarrow$ Corruption parameter
$n\_rounds \leftarrow$ Number of corruption rounds
$\tilde{d} \leftarrow$ Uncorrupted user decoding
$\tilde{d}^{'} \leftarrow \tilde{d}$ Corrupted user decoding
for i in range(n_rounds) **do** ▷ Corrupt for N rounds
  for $d \in \tilde{d}_t$ **do**
    $d_{is}, d_{sv} \leftarrow split(d_i)$ ▷ Split into intent-slot and slot-value.
    $\tilde{d}_{is}, p_{is} \leftarrow corrupt\_IS(d_{is}, IS_{cm}, \alpha_{corrupt})$

    if d_{sv} has slot value **then**
      $d^{'}_{sv}, p_{sv} \leftarrow corrupt\_slot\_value(d_{sv}, SV_{cm})$
    end if
    $d^{'} \leftarrow concatenate(d^{'}_{is}, d^{'}_{sv})$
    $p = (p_{is} + p_{sv})/2$
    $\tilde{d}^{'} += d^{'}$ with probability p ▷ Add the alphabet item d to $\tilde{d}^{'}$
  end for
end for
Return $normalize\_probabilities(\tilde{d}^{'})$ ▷ Normalise $\tilde{d}$ probabilities by using the maximum p score.

Where $\alpha_{corrupt}$ is a normalisation coefficient of a discrete probability distribution $X = [x_1, x_2, \cdots, x_N]$ such that it re-converts each probability into:

$$x_i^{\alpha_{corrupt}} = \frac{x_i^{\alpha_{corrupt}}}{\sum_{j=1}^{N} x_j^{\alpha_{corrupt}}}$$

Thus, the lower the alpha parameter, the more likely the system will corrupt the original user action. In addition, as the number of perturbation rounds n_rounds is higher, there are more chances of corrupting the initial user action.

An example of the $\tilde{d}_t = d_1, d_2$ where the DAs are $d_1 = request(address)$ and $d_2 = request(phone)$ can be found below. Then, the perturbed version of the user input $\tilde{d}_t^{'}$ is displayed.

```
Original user decoding:
-----------------------
    {'request(addr)': 1, 'request(phone)': 1}

Perturbed user decoding:
------------------------
```

```
{'request(addr)': 0.96, 'request(phone)': 0.48}
```

A valid method to measure the perturbation at NLU level is to compare the original $\tilde{d}_t$ and its perturbed version $\tilde{d}'_t$ by means of the L1 distance. The example above has $L1(\tilde{d}_t, \tilde{d}'_t) = 0.56$. This measure can give us some hints on the NLU error distribution of the corpora.

Table 3.15 shows the mean L1 differences between the correct NLU input and the perturbed version for the different corpus partitions. In addition, figure 3.10 depicts the distribution of the normalised frequencies according to the L1 difference of the correct NLU input and its corrupted version.

Tab. 3.15.: Mean L1 difference between correct user inputs and NLU perturbed outputs

Partition	L1 difference
DSTC2 - Train	0.6899
DSTC2-Dev	0.7893
DSTC2-Test	0.7198
All	0.7168

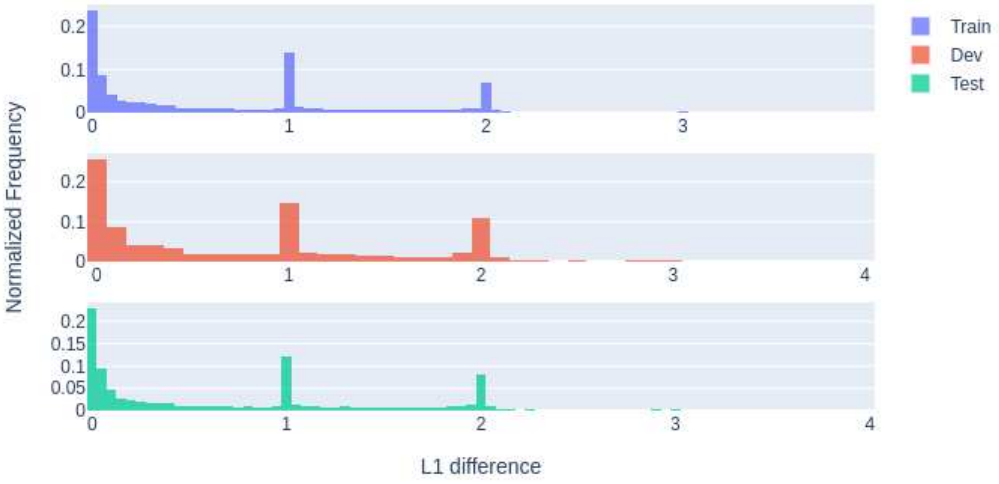


Fig. 3.10.: NLU perturbation L1 differences for the train, development and test corpus partitions of the DSTC2

As it can be seen, most of the differences happen on the [0-1] span, which means that the difference between the original NLU input and the corrupted one is usually

small. In addition, all histograms have significant overlapping, thus, the NLU error distribution is quite similar across data partitions.

In order to analyse how the proposed NLU corruption algorithm works, the effect of parameters $\alpha_{corrupt}$ and n_rounds over the initial user input has been measured by performing a grid search for $\alpha_{corrupt} \in [0, 4]$ and $n\_rounds \in [0, 20]$. Figure 3.11 shows the three dimensional graph of the L1 difference between the received user action and the corrupted one with respect to these two variables for the DSTC2 corpus. Each point was computed 5 times and the mean is used to build the graph surface.

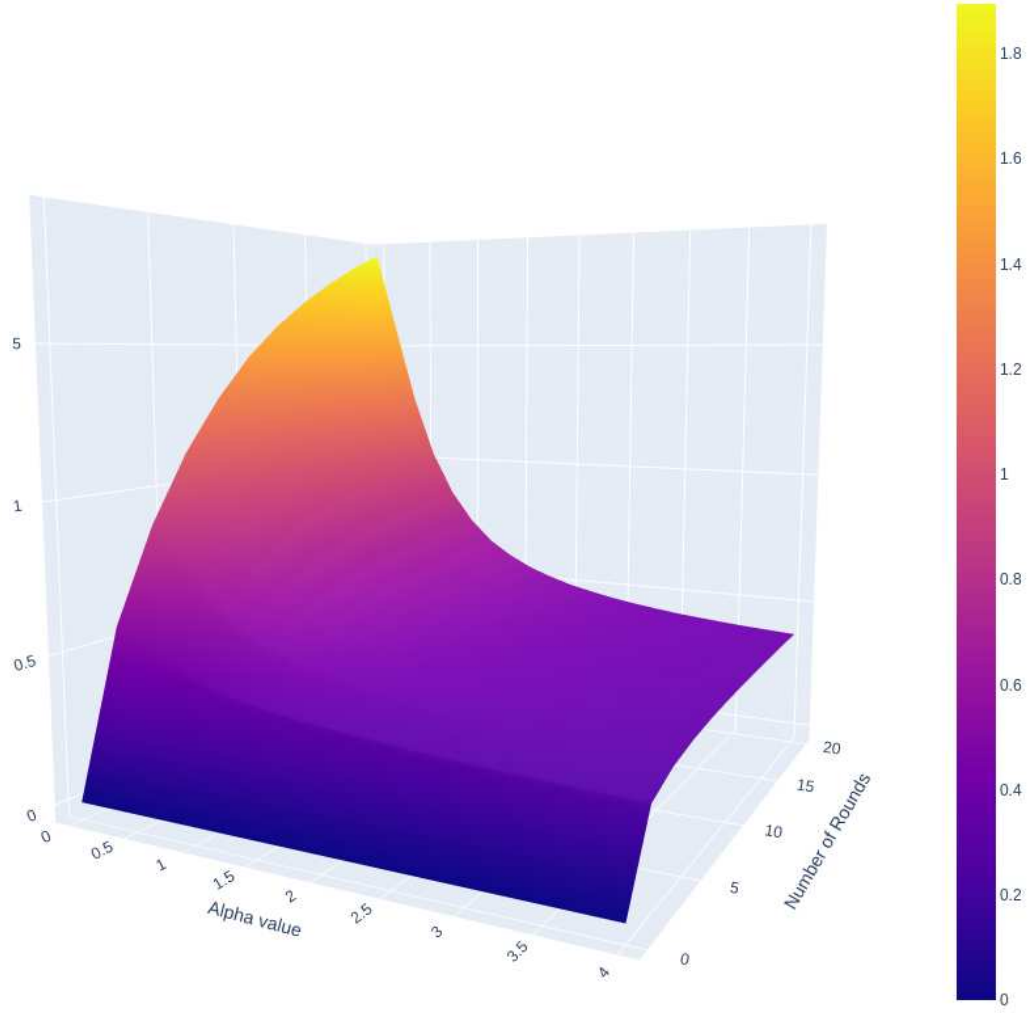


Fig. 3.11.: Surface of the L1 difference between the correct user action and the corrupted one by the NLU error model

As it can be seen, the lower $\alpha_{corrupt}$ is and the higher n_rounds is, the higher is the L1 difference between the original NLU decoding and the corrupted one. Once $\alpha_{corrupt}$ is set higher than 1 and the number of perturbation rounds is increased, the L1 difference reaches a plateau.

In order to evaluate the performance of the A-PFSBA DM over the DSTC2 corpus and the proposed NLU error model, $\alpha_{corrupt}$ was set to 1. Then, the n_rounds parameter was increased simulating different levels of channel noise. The set of histograms shown in Figure 3.12 depict the error distribution obtained over the complete DSTC2 corpus by using $\alpha_{corrupt} = 1$ and n_rounds 3, 5 and 10, respectively.

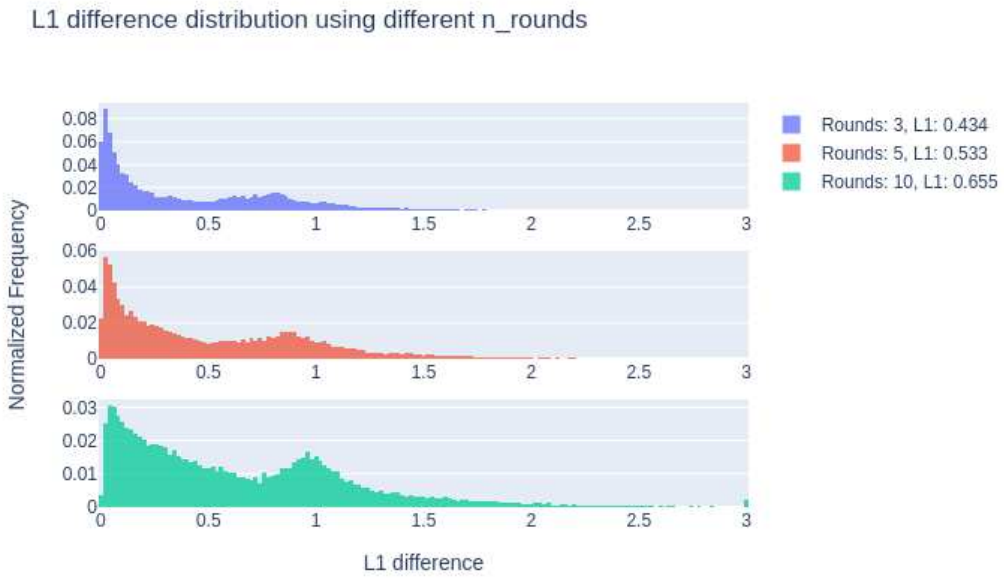


Fig. 3.12.: Histograms of L1 differences with $\alpha_{corrupt} = 1$ and different number of rounds

As it can be seen, an increase in the number of rounds directly increases uncertainty, rendering a smoothed distribution over the NLU error. The induced error is capable of achieving lower to higher L1 differences according to the experiment needs. Note that the error distribution of the proposed model is less spiked than the one observed in the DSTC2 corpus.

This NLU perturbation model is used in the DSTC2 dialogue generation experiments of Chapter 4 and in the incremental learning experiments of Chapter 6 in order to determine the robustness of the A-PFSBA framework against channel noise.

3.2.4 Evaluation Metrics

The following tests are applied to the system and are used to evaluate **dialogue generation** and *incremental learning* experiments between an UM and a DM in Chapters 4 and 6.

3.2.4.1 User Model Evaluation Metrics

In order to evaluate the DSTC2 UM, two different evaluation scenarios are used: direct comparison against real-user responses and Task Completion (TC) over generated dialogues.

Direct comparison contrasts the responses of the UM and the responses of real users in terms of Precision, Recall and F1-score as in (Layla et al., 2016; Schatzmann, Weilhammer, et al., 2006; Cuayáhuitl, Renals, et al., 2005; Quarteroni et al., 2010).

$$\textbf{Precision (P):} \frac{\text{Num. of correctly predicted dialogue acts}}{\text{Num. of predicted dialogue acts}}$$

$$\textbf{Recall (R):} \frac{\text{Num. of correctly predicted dialogue acts}}{\text{Num. of dialogue acts in the corpus}}$$

$$\textbf{F1-score:} \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

These metrics allow comparing the dialogue acts of real and simulated users, measuring the behavior and consistency of the model. They have been used to compare the proposed Deep Learning UM and the BAUM models.

Also the BAUM models have been evaluated in terms of TC: the dialogues generated between the UM and the DM have been evaluated using different tests. These tests check if the UM satisfies the following criteria during the conversation:

1. **Has the user given the constraints of the goal?** This test evaluates the percentage of constraints given in the initial goal that have been informed to the DM.

2. **Has the user requested the information of the goal?** This test evaluates the percentage of request slots that have been asked to the DM once a venue has been offered.

3. **Has the user asked about alternatives?** When the dialogue goal asks the user to request alternative restaurants, this test checks if the user has done so.

4. **Has the user said goodbye?** Checks if the user has said goodbye to close the interaction.

These TC sub-goal tests are used to measure and check the consistency of the proposed BAUMs.

3.2.4.2 Dialogue Manager Evaluation Metrics

When implementing the TC metric over the DSTC2 domain, a specific metric has been implemented to test each functionality of the DM. Each TC metric is correlated with a sub-task that the DM has to satisfy (K. Lu et al., 2019), which is tested individually. A sub-task is any partition that can be made to the original goal of the dialogue. For example, when the user requests the address of a venue, the system completes a sub-task or sub-goal of the dialogue by correctly answering with the current venue address, even if it may fail in other sub-goals (e.g. the system may misunderstand the given food type). In order to check all the functionalities, the TC rate is split into the following smaller and more granular tests.

1. **Has the system offered a restaurant that satisfies the user-given constraints?:** This test checks if the system has offered a restaurant that satisfies the constraints given by the user.

2. **Has the system given adequate responses to the user requests?** This test checks if the system has given an adequate response to the requests of the user in the immediate next turn. For example, when the user asks about the telephone number of an offered restaurant, this test checks if the system provides the corresponding phone number in the following turn.

3. **Has the system informed correctly about impossible combinations?** This test checks if the system lets the user know when the requested combination of user constraints cannot be found in the Database.

All the models presented in this Chapter will be used to evaluate the contributions made in this dissertation in terms of A-PFSBA model smoothing, policy-making and incremental learning.

Tab. 3.9.: Attribute update example for the User Model

	Goal Agenda Representation		User Model Attributes
Goal	*Constraints:* *area=north* *pricerange=moderate* *To request* *phone, addr*	ω_g	constraint-food: 0 constraint-area: 1 constraint-price:1 request-address: 1 request-phone: 1
Who Is	**Goal-delexicalised Dialogue Acts**		
System	welcomemsg	ω_0	constraint-food: 0 constraint-area: 1 constraint-price:1 request-address: 1 request-phone: 1
User	inform(pricerange=<price-goal-1>), inform(area=<area-goal-1>)	ω_1	constraint-food: 0 constraint-area: 1 constraint-price:1 request-address: 1 request-phone: 1 informed-price-goal: 1 informed-area-goal: 1
System	offer(venue=<venue>), inform(area=<area-goal-1>), inform(pricerange=<price-goal-1>)	ω_2	constraint-food: 0 constraint-area: 1 constraint-price:1 request-address: 1 request-phone: 1 informed-price-goal: 1 informed-area-goal: 1 system-unders-price-goal: 1 system-unders-area-goal: 1
User	request(address), request(phone)	ω_3	constraint-food: 0 constraint-area: 1 constraint-price:1 request-address: 1 request-phone: 1 informed-price-goal: 1 informed-area-goal: 1 system-unders-price-goal: 1 system-unders-area-goal: 1
System	offer(venue=<venue>), inform(address=<venue-address>), inform(phone=<venue-phone>)	ω_4	constraint-food: 0 constraint-area: 1 constraint-price:1 request-address: 1 request-phone: 1 informed-price-goal: 1 informed-area-goal: 1 system-unders-price-goal: 1 system-unders-area-goal: 1 offered-new-venue: 1 received-venue-address:1 received-venue-phone:1
User	thankyou, bye	ω_5	constraint-food: 0 constraint-area: 1 constraint-price:1 request-address: 1 request-phone: 1 informed-price-goal: 1 informed-area-goal: 1 system-unders-price-goal: 1 system-unders-area-goal: 1 offered-new-venue: 1 received-venue-address:1 received-venue-phone:1

4

Model Smoothing

Model smoothing is a key feature to avoid dialogue-breakdowns and improving such mechanism is one of the main contributions of this thesis.

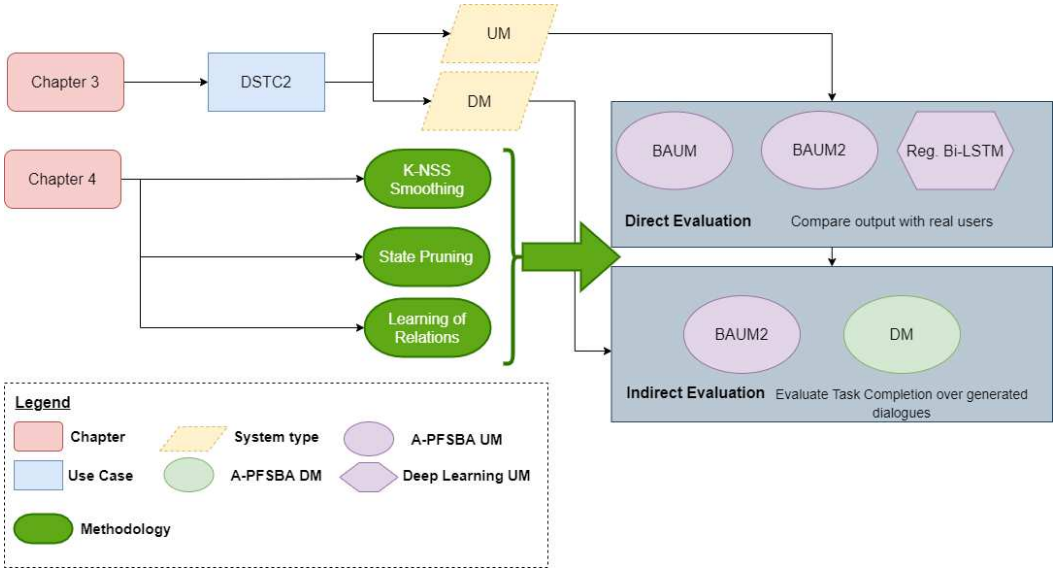


Fig. 4.1.: Model Smoothing chapter map.

In this chapter, the model smoothing approach employed in previous works (Orozko and M Inés Torres, 2015; Ghigi and M Inés Torres, 2015) is described first. Next, three new methods are proposed to improve the traditional Attributed Probabilistic Finite State Bi-Automata (A-PFSBA) model smoothing strategy, based on custom smoothing policies and spatial semantization methods. Two experimental setups are tested. First, the User Models (UMs) presented in Chapter 3 are directly evaluated against the output given by real users, evaluating their behaviour according to the presented contributions. Then, the Bi Automata User Model 2 (BAUM2) of Section 3.2.2.2 and the A-PFSBA Dialogue Manager (DM) of Section 3.2.3 are indirectly evaluated by measuring the Task Completion (TC) rate over the dialogues that they generate when talking to each other. The impact of the different smoothing strategies over the TC rate is measured using these generated dialogues.

4.1 Model Smoothing Strategy

As introduced in Chapter 2, A-PFSBA model learning also requires to define a model smoothing strategy. This strategy is used to generalise to unseen situations, routing unknown dialogue states to some known state. In practice, this means that the A-PFSBA model should give a response to any situation, known or not.

When dealing with stochastic structural models, smoothed models are usually learned at the training phase, rendering models capable of handling any state. Unfortunately, as the DM has to model a decision-making problem, there is no guarantee to ensure a proper decision outcome for every smoothed dialogue state. Due to this, the smoothing strategies described in this chapter are used at decoding time, ensuring a response for every dialogue state but without learning a smoothed model. Ways to improve the initial A-PFSBA model by using the smoothed dialogue states in the decision-making context are further detailed in Chapter 6.

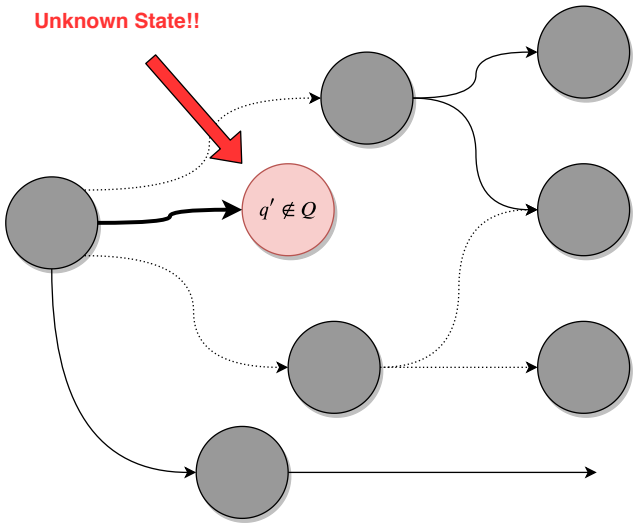


Fig. 4.2.: Dialogue interaction reaching an unknown state q'.

When the dialogue leads to an unseen dialogue state $q \notin Q$ in the A-PFSBA framework, the A-PFSBA model $\hat{M}$ is unable to execute its exploitation policy $\Pi_{A-PFSBA}(q')$ and the dialogue is broken. In this situation, a strategy to rectify the interaction must be defined. Under the assumption that "*Similar dialogue states will trigger similar responses*", model smoothing can be performed as a two-step task: first, the most similar known dialogue states with respect to $q' \notin Q$ are found and then, the next action is selected using the transitions from these states. In its simplest form, this is equivalent to finding the nearest neighbour states of the unknown state q' and to use their candidate actions to select the next system response (Orozko and M Inés Torres, 2015; Ghigi and M Inés Torres, 2015; Serras, Maria Inés Torres, et al., 2017; Serras, Maria Inés Torres, et al., 2019c).

In order to employ this smoothing strategy, a state similarity function Sim must be defined first. This spatial relation will determine the similarity between two dialogue states $(q_i, q_j) \ : \ q_i, q_j \in Q$.

$$\begin{aligned} Sim : Q \times Q \rightarrow \mathbb{R} \\ Sim(q_i, q_j) \rightarrow \mathbb{R} \end{aligned} \tag{4.1}$$

This similarity function needs to hold the following properties:

- **Identity of indescernibles:** $Sim(q_i, q_j) = max_{\forall q', q'' \in Q} Sim(q', q'') \longleftrightarrow q_i = q_j$
- **Symmetry:** $Sim(q_i, q_j) = Sim(q_j, q_i)$

Other properties, such as a bounded image $Sim(Q, Q) \in [0, 1]$ or a positive image $Sim(Q, Q) \in \mathbb{R}^+$, are desirable but optional. Note that Sim can also be a dissimilarity or distance function, by inverting the selection criteria (e.g. using the closest dialogue state instead of the most similar).

4.1.1 Nearest State Smoothing

Early DM works on A-PFSBA employed the Nearest State Smoothing (NSS) mechanism, where the unknown state q' is redirected to the closest state $q_j \in Q$ (Orozko and M Inés Torres, 2015; Ghigi and M Inés Torres, 2015; Serras, Maria Inés Torres, et al., 2017; Serras, Maria Inés Torres, et al., 2019c). Then, this state is used to sample the next action according to the defined policy $\tilde{a}_{t+1} = \Pi_{A-PFSBA}(q_j)$. The first step of this approach is represented in Figure 4.3.

4.1.2 Ambiguity of the State Representation

The way in which states $q \in Q$ are represented is crucial to define or select the spatial relation function Sim. Preliminary work represented dialogue states as strings (Orozko and M Inés Torres, 2015; Ghigi and M Inés Torres, 2015) and calculated their similarity by using edit distances. The limitation of this approach was shown in (Serras, Maria Inés Torres, et al., 2017), where introducing attributes and representing states as a binary vector make it possible to employ a combination of edit distances and the L1 norm.

The vectorial form of a dialogue state $\vec{q}$ can be defined as the concatenation of the vectorial form of the user action, system action and attributes of the state:

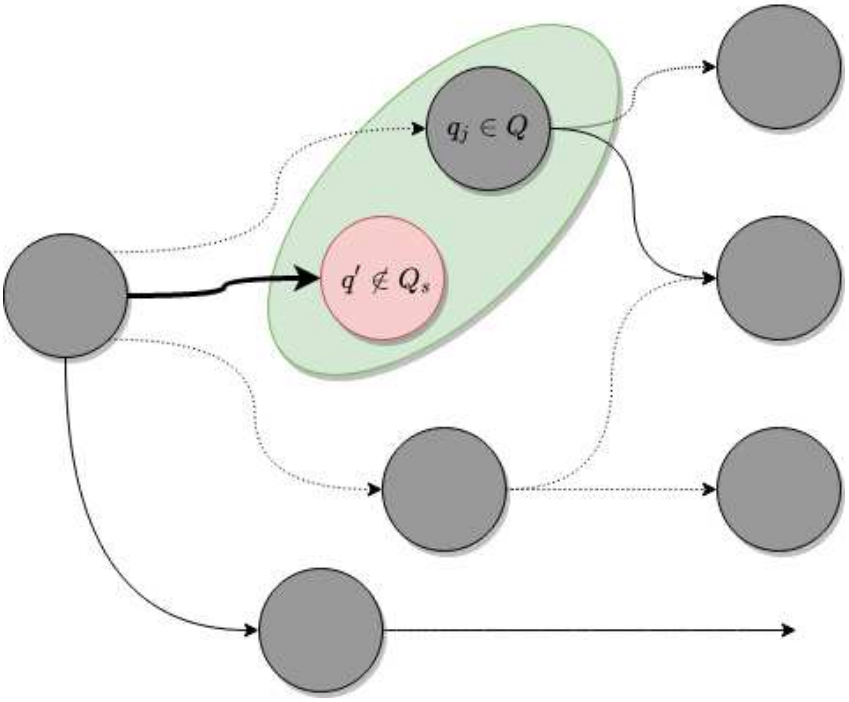


Fig. 4.3.: Smoothing of q' to its nearest and known dialogue state q_j.

$\vec{q} = [\vec{\tilde{d}}_q,\ \vec{\tilde{a}}_q,\ \vec{\omega}_q]]$, where $|\vec{q}| = |\Sigma| + |\Delta| + |\Omega| = L$. The notation $q[i]$ is used to determine the $i-th$ element of the vector. Along this chapter all the dialogue states are used in their vectorial form and $q = \vec{q}$ applies unless the contrary is said.

When the A-PFSBA dialogue states are represented in vectorial they can fully operate in the $\mathbf{R}^L$ space. This enables the use of spatial relations different from the edit distance in order to calculate state (dis)similarity Also, it makes possible to craft particular representations or training Machine Learning (ML) algorithms to be used as similarity or distance function.

Nevertheless, representing the A-PFSBA dialogue states as vectors has some drawbacks. The main one is that the semantic meaning of the user, system and/or attribute alphabets encoded in different dimensions of the vector can be ignored when classical spatial relations such as the euclidean distance or the cosine similarity are used. For example, given the following three user-decoded actions $\tilde{d}_i$ and their text representations:

1. **Inform(food=chinese)** "I would like Chinese food"

2. **Deny(restaurant=Nandos)** "I just don't feel like eating at Nandos"'

3. **Inform(area=city center)** "I would like a restaurant in the city center"

A binarized vector representation of these user actions could be [1, 0, 0], [0, 1, 0] and [0, 0, 1]. In this scenario, the euclidean distance between representations 1 and 2 would be the same as the one between 1 and 3, although their semantic meaning is completely different. This drawback must be taken into account during smoothing, because depending on how the similarity function is determined semantically different states may be encountered under the same similarity radius. The smoothing strategies proposed in the next three sections aim to tackle this issue.

4.2 K-Nearest State Smoothing

In order to augment the context of the smoothed state q', a voting mechanism can be defined using the K nearest states instead of just the nearest one. Formally, being $q' \notin Q$ the unknown state, let $Q_K = \{q_1, ..., q_j, ..., q_K \ : \ q_i \in Q \ \forall i = 1, \cdots, K\}$ be the set of closest known K dialogue states according to Sim.

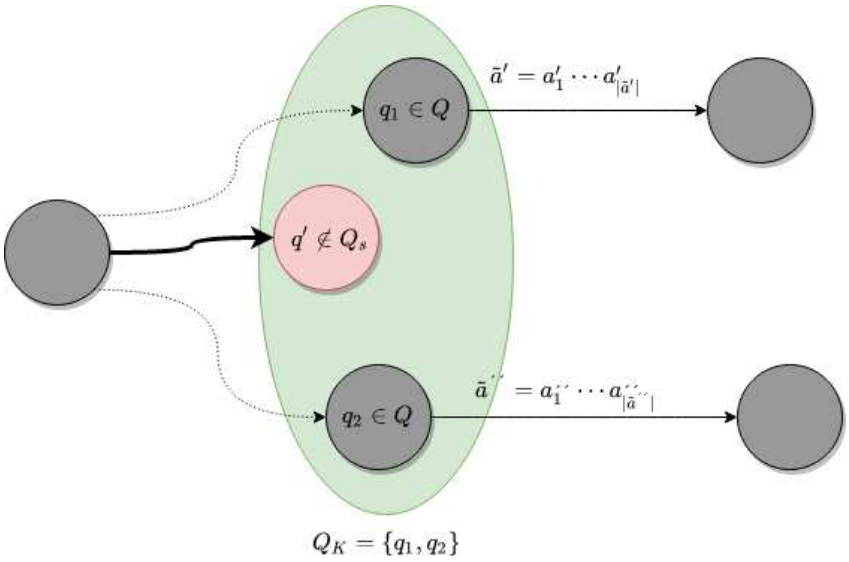


Fig. 4.4.: System actions $a^{'}$ and $a^{''}$ departing from the closest 2 dialogue states q_1, q_2

Using Q_K as departing states, the policy $\Pi_{A-PFSBA}$ needs to be able to exploit all the system actions that depart from Q_K:

$$\delta(Q_K) = \{\tilde{a}_k \in \Delta^{\leq n} \ : \exists(q_i, \tilde{a}_k, *) \in \delta_S \ \forall q_i \in Q_K\}$$

Where $*$ is used to denote any dialogue state $q \in Q$. Figure 4.4 depicts the departing actions of the set Q_K, where $\delta(Q_K) = \{a^{'}, a^{''}\}$.

To this end, two policies are defined: **Maximum Scoring Action (MSA)** and **Thresholded Scoring Action (TSA)**, which are a variant of the same voting mechanism.

Let $\Delta_{Q_K} = \cup_{\tilde{a}' \in \delta(Q_K)} \{a_i \ \forall a_i \in \tilde{a}'\} \subseteq \Delta$ be the candidates from which the next system action $\tilde{a}_{t+1} \in \Delta^{\leq n}$ will be composed. In order to perform this composition, a weighted-voting method is defined that assigns a score to each item $a_i \in \Delta_{Q_K}$:

$$S_{smoothing}(a_i) = \frac{1}{K} \sum_{q_j \in Q_K} Sim(q',\ q_j) P(a_i|\ q_j) \tag{4.2}$$

Where $Sim(q',\ q_j)$ is the similarity between the unknown state q' and the known dialogue state q_j and $P(a_i|\ q_j)$ is the probability of using a_i as an element to make a transition action from q_j.

Then, the **MSA** policy selects the next user action $\tilde{a}_{t+1}$ as the composition of those system actions $a_i \in \Delta_{Q_K}$ with maximum score:

$$\tilde{a}_{t+1} = \{a_k : a_k = \arg\max_{a_i \in \Delta_{Q_K}} S_{smoothing}(a_i)\} \tag{4.3}$$

On the other hand, in the **TSA** policy the maximum constraint is relaxed by using a threshold θ_{TSA}. If the score of an action $a \in \Delta_{Q_K}$ is higher than or equal to θ_{TSA}, it is used in the next action $\tilde{a}_{t+1}$:

$$\tilde{a}_{t+1} = \{a_i : S_{smoothing}(a_i) \geq \theta_{TSA}\ \forall a_i \in \Delta_{Q_K}\} \tag{4.4}$$

These policies are specific to K-Nearest State Smoothing (KNSS) model smoothing and are only used when this smoothing strategy is triggered. Note that these mechanisms could also be implemented for UM, just by swapping the Δ alphabet with the user-alphabet Σ.

4.3 State Pruning

The larger the state-space $|Q|$, the higher the computation time required to calculate the similarity between the K dialogue states closest to unknown state q' with every other dialogue state in the system $Sim(q', q_j)\ \forall q_j \in Q_S$. Unfortunately, this affects response time, which needs to be quick enough to handle real-time interactions.

Maintaining the assumption that similar states will trigger similar actions and knowing that $|\Sigma| << |Q|$ and $|\Delta| << Q$ will be satisfied for DMs and UMs trained with enough data to model the interactions, a dialogue state pruning methodology has been defined.

The dialogue state pruning methodology aims to establish a semantic relation between the representation of dialogue states and their output action. Then, this relation is used to exclude and filter dialogue states that are not related to the unknown state $q' \notin Q$.

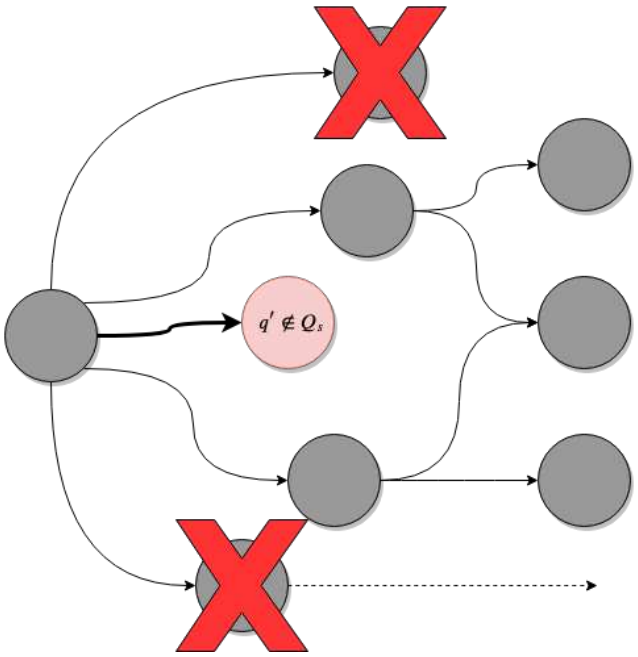


Fig. 4.5.: Illustration of state pruning previous to searching for the most similar states to the unknown state q'

To that end, a Pruning Model (PM) that learns the relation between the vectorial representations of the dialogue state $q \in Q$ and the actions that are triggered from those states $\delta(q)$ is determined. Then, the PM is a function that provides a score for each item of the system action alphabet $a \in \Delta$ for a DM. The objective is to prune those states that trigger low scoring actions as shown in Figure 4.5.

$$PM(q) \rightarrow \{(a_i, score_i) | \, \forall a_i \in \Delta\}$$

This function can be easily achieved using conventional ML algorithms, such as Multinomial Naive Bayes (MNB) or Support Vector Machines (SVM). The only requirement is that the chosen algorithm needs to perform decoding fast enough to ensure the real-time performance of the DM or the UM. The selection of the best model also needs to take into account other relevant factors for each application scenario such as training time, performance or memory consumption. Algorithm 3 details how to train a ML model to be used as a PM and Algorithm 4 describes how to use a PM during decoding.

Algorithm 3 How to train a ML model to be used as a PM

```
Mod ← Model to train as PM
M̂ ← A-PFSBA structure that models the interactions
X, y = [], []
HT = {}                          ▷ Hash table that relates dialogue actions with state indexes.
for q_i ∈ Q_S do
    for ã_j ∈ δ(q_j) do
        for a_j^k ∈ ã_j do
            X.append(q_i)
            y.append(a_j^k)
            HT[a_j^k].append(q_i.index)
        end for
    end for
end for
PM ← Mod.fit(X, y)               ▷ Train the model with the state and action data
Return Mod, HT
```

Note that in order to train a PM for UM purposes, it is enough to change dialogue states for user states Q_U and system actions for user actions $d \in \Sigma$.

Algorithm 4 How to prune dialogue states by using a PM

```
PM ← PM trained model
q' ← Unknown dialogue state
θ_prune Pruning threshold
HT ← Hash table that relates actions to state indexes.
Q_invalid = {}Set of invalid states.
action_score_dict ← PM.predict_action_score(q')
for a_j ∈ Δ do
    if action_score_dict[a_j] < θ_prune then
        for state_index ∈ HT[a_j] do
            q_invalid ← Q.get_state_by_index(state_index)
            Q_invalid.add(state_index)
        end for
    end if
end for
Q_valid ← Q_S / Q_invalid        ▷ i.e. the states of Q_S that are valid
Return Q_valid
```

As the proposed state pruning mechanism is triggered before nearest dialogue-state sampling for model smoothing, these two advantages are obtained:

1. The amount of dialogue states needed to compute $Sim(q', q_j)\ \forall q_j \in Q_{valid}$ is reduced, thus shortening the time required for finding the most similar dialogue states when smoothing .

2. Semantically unrelated dialogue states are removed so that similar-yet-unrelated dialogue states according to $Sim(q', q_j)$ are not taken into account for model smoothing.

The proposed state pruning method is used to define a semantic model in the dialogue state space defined over $\mathbf{R}^L$ where $L = |q|$, thus, it can be used with the previously defined KNSS policies. Also, this algorithm can also be used for UM purposes, changing only dialogue states by user states Q_U and system actions by user actions $d \in \Sigma$.

4.3.0.1 Predictive Smoothing

Instead of using a ML model just to prune states, a predictive algorithm trained on the A-PFSBA model $\hat{M}$ as done with the PM in Algorithm 3 can be used to predict the next action. So the next action from an unknown state q' can be sampled as:

$$\tilde{a}_{t+1} = \{a_k : a_k = \arg\max_{a_i \in \Delta} PM(q')\}$$

Note that similar to K-NSS smoothing, this method does not guarantee that an existing transition of the A-PFSBA model $\hat{M}$ is used.

4.4 Spatial Relation Learning

Another approach to improve model smoothing is to learn a similarity function that takes into account the semantics of the states over the vectorial space defined by dialogue state vectors $q \in Q_S$ of the A-PFSBA model $\hat{M}$.

To build a similarity or distance function that takes into account the inter-dimensional relation, and being W a similarity matrix of dimension $L \times L$ where $L = |q|$, the soft cosine similarity between two states $q',\ q''$ can be defined as:

$$soft\_cos(q',\ q'') = \frac{\sum_{i,j}^{L} w_{ij} q'[i] q''[j]}{\sqrt{\sum_{i,j}^{L} w_{ij} q'[i] q'[j]} \sqrt{\sum_{i,j}^{L} w_{ij} q''[i] q''[j]}}$$

Where $w_{ij} \in W$. Note that if $W = I_L$, the resulting formula is the conventional cosine similarity. Additionally, and using the same matrix W a weighted version of the Euclidean distance can be defined:

$$weighted\_euclidean(q',\ q'') \ = \ \sqrt{\sum_{i,j}^{L} w_{ij} \cdot (q'[i] - q''[j])^2} \tag{4.5}$$

The weights w_{ij} of the similarity matrix establish a relation between the $i-th$ element of q' and the $j-th$ element of q'', so learning these weights is equivalent to learning the first order relation between the dialogue state dimensions.

The symmetry of the weight matrix W is set as a constraint, so the symmetry property is met. In addition, if the range of the weights of W is positively defined, the image of the soft cosine or weighted euclidean will also be positively defined.

4.4.1 Chi Square Test

The method proposed to learn the weight matrix W is to compute the χ^2 test over the vectorized dialogue state q and its transition actions $\delta(q)$. The χ^2 test is widely used for feature selection to discard uninformative features, i.e. those features that are independent of the class or element to be predicted. When a χ^2 test is performed over a set of features X and a possible set of outcomes Y, a high score over a feature $x \in X$ means that the null-hypothesis H_0 (i.e. the statistical independence between x and Y) can be discarded and, therefore, that feature $x \in X$ correlates with the possible outcomes of Y.

When it comes to the A-PFSBA structure, this is equivalent to testing if the joint-occurrence of all the combinations of the different alphabets Δ, Σ, Ω have statistical dependence when predicting the next action $\tilde{a}_{t+1}$. As an illustrative example, let us suppose that the χ^2 is performed over certain elements of the user and system alphabet $(d_i, a_j) \in \Sigma \times \Delta$. Then, the co-occurrence of both items can be written as $cc(d_i, a_j)$. In order to measure if the co-occurrence of these two elements is relevant, the $\chi^2(cc(d_i, a_j))$ test poses the following null hypothesis:

$$H_0 \ : P(\tilde{a}_{t+1}, cc(d_i, a_j)) = P(\tilde{a}_{t+1}) \cdot P(cc(d_i, a_j))$$

Informally, the null hypothesis is that the joint probability of a next action $\tilde{a}_{t+1}$ (any next action) and the co-occurrence of d_i and a_j is equal to the multiplication of both probabilities, this is, that they are independent.

Then, if the result of the $\chi^2(cc(d_i, a_j))$ test is high, the null hypothesis can be discarded. This means that the co-occurrence of the alphabet elements d_i and a_j is relevant to determine the next system action $\tilde{a}_{t+1}$.

As a result, the χ^2 test can be used to set the weights w_{ij} of the spatial relation matrix W for the weighted Euclidean distance of Equation 4.5. Each weight w_{ij} will determine the importance of the co-occurrence of the features of the i-th and j-th dimension of the dialogue state vector, where higher scores (co-occurrences that are non-independent to determine the next action) will have a higher impact on the spatial relation.

Denoting $\chi^2(cc(q[i], q[j]))$ as the chi square test result over the joint-feature that captures the co-occurrence of features $q[i], q[j]$ with the next system action $\tilde{a}_{t+1}$, the spatial relation matrix W can be determined as:

$$W = \begin{pmatrix} \chi^2(cc(q[1], q[1])) & \cdots & \cdots & \chi^2(cc(q[1], q[L])) \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ \chi^2(cc(q[L], q[1])) & \cdots & \cdots & \chi^2(cc(q[1], q[L])) \end{pmatrix}$$

Note that for the DM only system states $Q_s \subset Q$ are taken into account and for UM only user states $Q_u \subset Q$.

4.5 Direct Evaluation over User Models

In the first experimental section, the impact of the different model smoothing strategies is evaluated using the UMs. Following the evaluation procedure used by (Layla et al., 2016; Serras, Maria Inés Torres, et al., 2019b; Serras, Maria Inés Torres, et al., 2019a; Cuayáhuitl, Renals, et al., 2005), UMs (BAUM, BAUM2 and Reg. Bi-LSTM) are evaluated in direct comparison by contransting the output of the UM with the real users' output in terms of precision, recall and F1 score.

4.5.1 Baseline Evaluation

In order to set a starting point, both for the single-goal BAUM and for the double-goal BAUM2 introduced in Chapter 3, a baseline is built which has the following configuration:

Tab. 4.1.: Parameter configuration for both baselines

Parameter	**Description**
Regular A-PFSBA Policy	Maximum Probability Transition
Spatial Relation	Cosine Similarity
Dialogue Smoothing - K States	Smooth to the closest one (K=1)
Dialogue Smoothing - Policy	Maximum Probability Transition
Dialogues used for training	Training set

As it can be seen, this is the simplest configuration, which is later enhanced by using the model smoothing techniques proposed in this chapter.

As a hard-baseline for the first BAUM, the Regularized Bi-LSTM User Model (**Reg. Bi-LSTM**) presented in Chapter 3 was employed to test the performance of the first BAUM. The **Reg. Bi-LSTM** UM outperformed previous DL UM approaches and set the stat-of-the-art results of that time (Layla et al., 2016). The results of these baselines in direct comparison are shown in Table 4.2, where **Seq-to-one** was the previous SotA using DL techniques (Layla et al., 2016). Note that the BAUM and Reg. Bi-LSTM baselines are trained in a single-goal setup, so they can be compared to each other. Also, as BAUM2 is trained in a double-goal scenario, it cannot be directly compared to either BAUM or Reg. Bi-LSTM.

The proposed models are evaluated using different granularity, at intent (inform, request, ...) level and delexicalised slot value level (e.g. *inform(food=goal)*). All the models use the maximum likelihood to select the next action and for the A-PFSBA models (BAUM, BAUM2) the nearest state is used for smoothing. The Seq-to-one model is used for comparative purposes against the Reg. Bi-LSTM and BAUM models.

Tab. 4.2.: Baseline performance over the DSTC2 Corpus

	Development set			**Test set**		
Baseline	Precision	Recall	F1 Score	Precision	Recall	F1 Score
BAUM*	0.690/0.566	0.731/0.591	0.710/0.578	0.699/0.556	0.728/0.577	0.712/0.566
Seq-to-one (Layla et al., 2016)	–	–	0.37/–	–	–	0.29/–
Reg. Bi-LSTM	0.70/0.60	0.72/0.63	0.71/0.62	0.71/0.60	0.73/0.64	0.72/0.62
BAUM2	0.699/0.563	0.752/0.619	0.725/0.590	0.713/0.568	0.752/0.609	0.732/0.587

As we can see, for the first scenario, BAUM and Reg. Bi-LSTM both outperform the Seq-to-one, and, the Reg. Bi-LSTM improves the F1 score, where the difference regarding BAUM increases when evaluating at slot-value level. The BAUM2, despite not being comparable to the previous models, slightly improves the F1 scores in both development and test dataset although it BAUM2 has a larger feature-space. This can be explained as the goal now encodes information about more complex scenarios as explained in Section 3.2.1.1.

4.5.2 K-Nearest State Smoothing Evaluation

In this section the KNSS policies from Section 4.2 are tested. These policies employ the K-closest dialogue states to the unknown one q' and employ an specific policy $\Pi_{smoothing}$ to decide which action to take next. This policy is only used when model smoothing is activated. In the rest of the situations the regular A-PFSBA policy is used $\Pi_{A-PFSBA}$.

These policies are tested over the A-PFSBA based UMs BAUM and BAUM2. These UMs are explained in Sections 3.2.2.2 and 3.2.2.2, where the BAUM2 is an enhancement of the BAUM model, which employs more attributes to encode more complex dialogue goals.

4.5.2.1 Maximum Scoring Action Policy

For the MSA smoothing policy, the following parameters are evaluated. The achieved results for the BAUM and BAUM2 are shown below:

Parameter	**Description**
Regular Policy	Maximum Probability Transition
Spatial Relation	Cosine Similarity
Dialogue Smoothing - K States	K in [1, ... , 20]
Dialogue Smoothing - Policy	MSA

The achieved results, both for BAUM and BAUM2 for the test sets are shown in Figure 4.6.

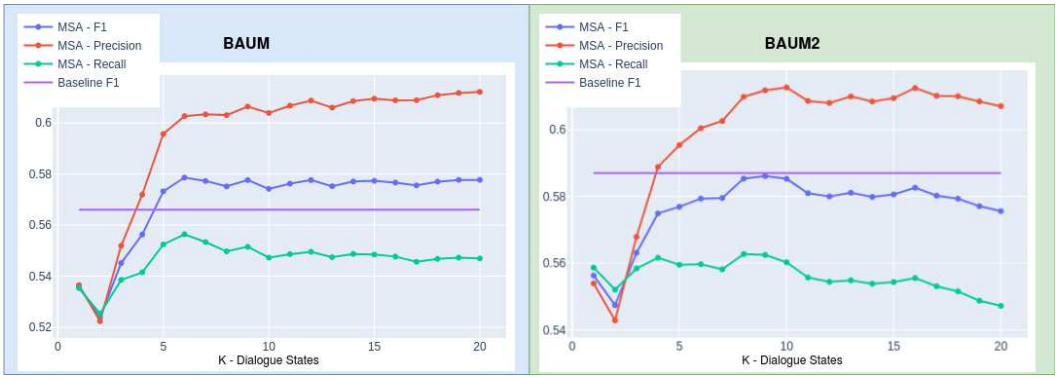


Fig. 4.6.: BAUM and BAUM2 MSA Precision/Recall/F1 score evolution according to the K states in the Test set

On the whole, for both BAUM and BAUM2 cases the MSA achieves a high precision when comparing their output with the responses given by real users. Yet, as the policy samples only those user actions $d_i \in \Sigma$ which obtain the maximum score in Equation 4.3, the recall worsens, indicating that the real users tend to be more verbose when giving answers in the DSTC2 scenario.

4.5.2.2 Thresholded Scoring Action Policy

For the thresholded action policy, the following configurations are tested:

Parameter	**Description**
Regular Policy	Maximum Probability Transition
Spatial Relation	Cosine Similarity
Dialogue Smoothing - K States	K in [1, ..., 20]
Dialogue Smoothing - Policy	TSA
Threshold Relax μ_{rel}	[0.7,0.8, 0.9]

For the threshold used in Equation 4.4, a relaxed-maximum score is used as a threshold, according to a relaxing coefficient μ_{rel}, so the final threshold θ_{tsa} can be calculated:

$$\theta_{TSA} = \mu_{rel} \cdot [\max_{d_i \in \Delta_{Q_K}} S_{smoothing}(d_i)]$$

This dynamic threshold-setting mechanism has some advantages over the ad-hoc threshold fine-tuning presented at (Serras, Maria Inés Torres, et al., 2019a), such as avoiding empty responses due to an overly constraining threshold, and a more generic implementation of the TSA smoothing policy.

Figure 4.7 describes the achieved slot-level F1 metric with different μ_{rel} parameters over the DSTC2 test set. The x-axis describes the amount of neighbour dialogue-states K used in the TSA.

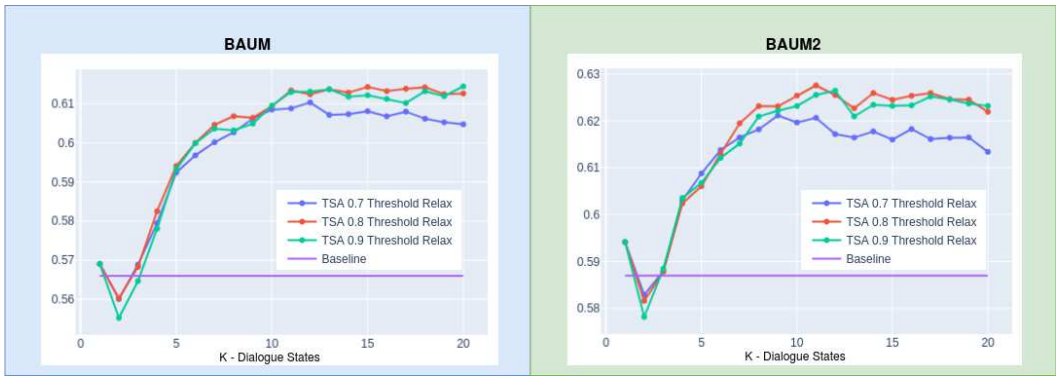


Fig. 4.7.: BAUM an BAUM2 TSA slot-level F1 score evolution according to the K states in the test set and different μ_{rel} coefficients

For both UMs, the best performing setup is the TSA with a relaxing coefficient of $\mu_{rel} = 0.8$, where $10 < K < 15$ obtains the best results in terms of slot-f1. It is interesting to note that these smoothing policies, as they weigh the most similar states to draw the next action, require a few states to ensure its robustness, and using $K < 5$ may even worsen the baseline results. This can be explained due to the use of spatial relations such as the cosine similarity over the the dialogue state vectors $\vec{q}$, which do not capture the semantics of the dialogue-state, so the spatial similarity does not guarantee the semantic similarity (i.e. that both states will yield a similar result).

In the Figure 4.8, all the scores of the TSA smoothing policy with a relaxing parameter of 0.8 are depicted for the DSTC2 development and test set.

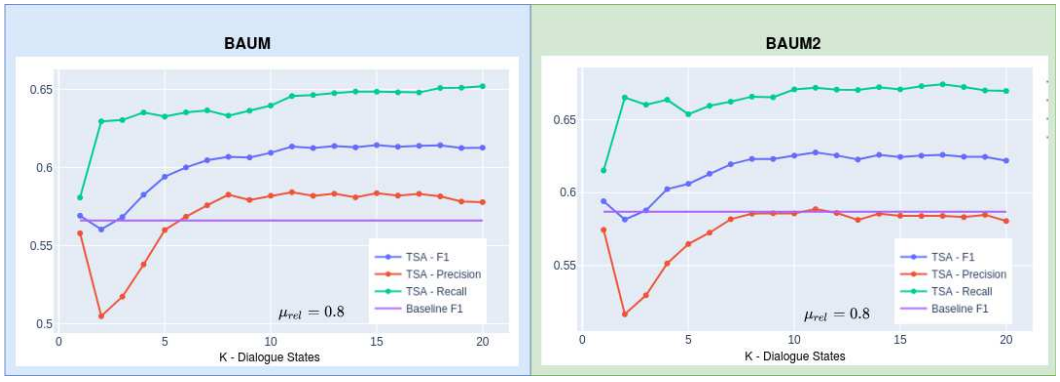


Fig. 4.8.: BAUM and BAUM2 TSA slot-level metric evolution according to the K states in the test set with $\mu_{rel} = 0.8$

As it can be seen, both for BAUM and BAUM2 the TSA dialogue smoothing policy clearly favours the Recall of the system, rendering a more verbose UM. For the DSTC2 case, this strategy considerably improves the achieved results. Yet, if we want

to modify the behaviour of our UM to a more precise and less verbose user profile, the MSA smoothing policy could be used.

Regarding the number of neighbour states to take into account, both for MSA and TSA setups, $5 < K < 15$ is advised (at least when simple spatial relations are used, such as the cosine similarity). Augmenting the number of states to draw the next action provides an easy way to improve the robustness of the dialogue smoothing strategy. Note, however, the bigger K is, the bigger the amount of calculations to be performed will be. And as a diminishing effect can be perceived when $K > 15$, to keep increasing the amount of neighbours will have a negative impact on the smoothing policy.

4.5.3 State Pruning Evaluation

The BAUM/BAUM2 used as a baseline selects just the nearest dialogue state when performing the dialogue smoothing strategy by using the cosine similarity. Once this state is fixed, the next action is chosen using the transition edge with the highest probability. This initial baseline is enhanced with the dialogue state pruning method presented at 4.3 with five well-known off-the-shelf ML algorithms. The PM is trained using the A-PFSBA model $\hat{M}$ inferred from the DSTC2 training set. The θ_{prune} threshold is selected using the development set and performing a grid search with step size of 0.1 to optimize the F1 score.

The ML algorithms employed in this section are the Multinomial Naive Bayes (**MNB**) (Schütze et al., 2008), Support Vector Machines with linear kernel function (**SVM**) (Platt, 1999), Passive Aggressive Classifier (**PA**) (Crammer et al., 2006), Multi-Layer Perceptron (Hinton, 1990) (**MLP**) and Random Forest Classifier (**RF**) (Breiman, 2001). All these algorithms were implemented using scikit-learn (Pedregosa et al., 2011) and for the sake of simplicity, given that the goal is to prove the ease of use of the state pruning, no hiperparameter tuning was performed. As a hard baseline for the BAUM method, one of the latest DL user-modelling approach based on an ensemble of regularized Bi-LSTM (Serras, Maria Inés Torres, et al., 2019b) is also included (**Reg. Bi-LSTM**).

Tab. 4.3.: Act/Slot level global evaluation metrics over the DSTC2 corpus for different pruning methods. The BAUM model with no dialogue-state pruning is used as a baseline and the DL Bi-LSTM ensemble as a hard baseline.

		Development set			**Test set**		
Pruning Model	Best $\theta_{override}$	Precision	Recall	F1 Score	Precision	Recall	F1 Score
baseline: BAUM	–	0.690/0.566	0.731/0.591	0.710/0.578	0.699/0.556	0.728/0.577	0.712/0.566
BAUM - MNB	0.1	0.717/0.590	0.746/0.605	0.731/0.598	0.747/0.594	0.744/0.586	0.746/0.590
BAUM - SVM	0.2	**0.766/0.654**	0.750/**0.630**	**0.758/0.642**	**0.790/0.662**	0.756/0.625	**0.772/0.643**
BAUM - PA	-1.6	0.733/0.610	0.739/0.605	0.736/0.607	0.749/0.607	0.742/0.595	0.745/0.600
BAUM - MLP	0.2	0.752/0.638	**0.753**/0.626	0.752/0.631	0.774/0.638	**0.761**/0.618	0.768/0.628
BAUM - RF	0.3	0.757/0.630	0.705/0.579	0.730/0.603	0.770/0.630	0.703/0.570	0.735/0.599
Reg. Bi-LSTM	–	0.70/0.60	0.72/**0.63**	0.71/0.62	0.71/0.60	0.73/**0.64**	0.72/0.62

Tab. 4.4.: Act/Slot level global evaluation metrics over the DSTC2 corpus for different pruning methods. The BAUM2 model with no dialogue-state pruning is used as a baseline.

		Development set			**Test set**		
Pruning Model	Best $\theta_{override}$	Precision	Recall	F1 Score	Precision	Recall	F1 Score
baseline: BAUM2	–	0.699/0.563	0.699/0.563	0.752/0.619	0.713/0.568	0.752/0.609	0.732/0.587
BAUM2 - MNB	0.1	0.724/0.592	0.757/0.628	0.740/0.609	0.761/0.607	0.755/0.609	0.758/0.608
BAUM2 - SVM	0.2	**0.768/0.649**	0.768/**0.656**	**0.768/0.652**	**0.801/0.678**	0.776/0.662	**0.789/0.670**
BAUM2 - PA	-1.1	0.738/0.614	0.753/0.635	0.746/0.624	0.749/0.620	0.742/0.621	0.746/0.620
BAUM2 - MLP	0.2	0.766/0.641	**0.773**/0.656	0.769/0.648	0.791/0.660	0.784/0.659	0.787/0.659
BAUM2 - RF	0.3	0.783/0.655	0.724/0.618	0.753/0.636	0.805/0.678	0.728/0.623	0.765/0.650

As it is clearly seen in Table 4.4, the inclusion of a dialogue state pruning mechanism improves the BAUM and BAUM2 baselines regardless of the ML algorithm used to build the PM. The best suited ML algorithms for the DSTC2 case happened to be the SVM and the MLP, both geometric classifiers. It is interesting to note that the results obtained with the BAUM - SVM/MLP achieved higher scores than the Deep Learning Bi-LSTM ensemble in most of the metrics.

4.5.3.1 Using the Dialogue State to Predict the Next Action

One of the question that arises when using ML classification algorithms to prune the non-equivalent dialogue states is "*Why not use the ML classifier to select the next action?*". This experiment is carried out for both presented A-PFSBA UMs in direct comparison.

Tab. 4.5.: Results achieved by selecting the next action with the ML algorithm instead of using the pruned back-off smoothing method for the BAUM model

Predicting the Next Action	**Development set**			**Test set**		
Prediction Algorithm	Precision	Recall	F1 Score	Precision	Recall	F1 Score
MNB	0.649/0.515	0.706/0.556	0.676/0.535	0.700/0.489	0.730/0.500	0.715/0.490
SVM	**0.732**/0.609	0.711/0.582	0.721/0.595	0.767/**0.631**	0.721/0.586	0.743/**0.607**
PA	0.698/0.546	0.689/0.527	0.694/0.537	0.725/0.531	0.701/0.502	0.713/0.516
MLP	0.730/**0.611**	**0.732/0.604**	**0.731/0.607**	**0.770**/0.613	**0.752/0.590**	**0.760**/0.60
RF	0.703/0.570	0.701/0.562	0.702/0.566	0.704/0.551	0.693/0.542	0.699/0.547

Tab. 4.6.: Results achieved by selecting the next action with the ML algorithm instead of using the pruned back-off smoothing method for the BAUM2 model

Predicting the Next Action	**Development set**			**Test set**		
Prediction Algorithm	Precision	Recall	F1 Score	Precision	Recall	F1 Score
MNB	0.655/0.502	0.744/0.594	0.697/0.544	0.691/0.493	0.739/0.549	0.715/0.520
SVM	0.738/0.605	0.766/0.638	0.751/0.621	0.774/**0.634**	0.774/0.642	0.774/0.638
PA	0.694/0.550	0.682/0.547	0.688/0.549	0.697/0.535	0.672/0.517	0.684/0.526
MLP	**0.741/0.617**	**0.804/0.678**	**0.771/0.646**	**0.777**/0.632	**0.826/0.681**	**0.801/0.655**
RF	0.708/0.568	0.748/0.619	0.727/0.592	0.718/0.561	0.753/0.611	0.735/0.585

Both for BAUM and BAUM2, Tables 4.5 and 4.6 show that this method can achieve very good results, sometimes even better than the dialogue state pruning method as for the **BAUM2 - MLP** scenario, although it lacks robustness. The obtained results vary greatly depending on the used ML algorithm. For this particular case, geometric and linear models (SVM, PA, MLP) achieve the best results, even though the difference between one approach and the other is significant. Also, simpler statistical predictors such as the MNB achieve worse results than the BAUM/BAUM2 baseline, whereas in the state pruning setup all mechanisms improved the initial baseline.

4.5.4 Chi-square Distance Evaluation

This section introduces and explains the experiments performed with the proposed χ^2 similarity W matrix and the weighted Euclidean distance presented at Equation 4.5 for UM direct comparison. The two BAUMs are enhanced with the χ^2-based W spatial relation matrix, and its impact is evaluated against the BAUM and BAUM2 baselines.

Tab. 4.7.: Direct evaluation impact of using Chi square score in BAUM

	Development Set			**Test Set**		
BAUM Model	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Baseline	0.690/0.566	0.731/0.560	0.710/0.578	0.699/0.556	0.728/0.577	0.712/0.566
Baseline + χ^2	**0.703/0.576**	**0.740/0.591**	**0.721/0.587**	**0.730/0.589**	**0.754/0.602**	**0.741/0.595**

Tab. 4.8.: Direct evaluation impact of using Chi square score in BAUM2

	Development Set			**Test Set**		
BAUM2 Model	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Baseline	0.700/0.566	0.753/0.618	0.726/0.590	0.703/0.563	0.743/0.605	0.73/0.583
Baseline + χ^2	**0.72/0.583**	**0.756/0.626**	**0.737/0.604**	**0.739/0.592**	**0.765/0.621**	**0.752/0.606**

When evaluating under the direct comparison, the use of the χ^2 scores of the extended features as the weights $w_{i,j}$ of the spatial relation W, and the use of the weighted Euclidean distance, improve the achieved results in direct comparison. The improvement keeps being consistent for both versions of BAUM as shown in Tables

4.7, 4.8. The score-distribution for both BAUMs, i.e. the value of the χ^2 test for each feature of the expanded vector dialogue-state, is depicted in Figure 4.9. The behaviour of both distributions is similar, where most of the features are packed near zero.

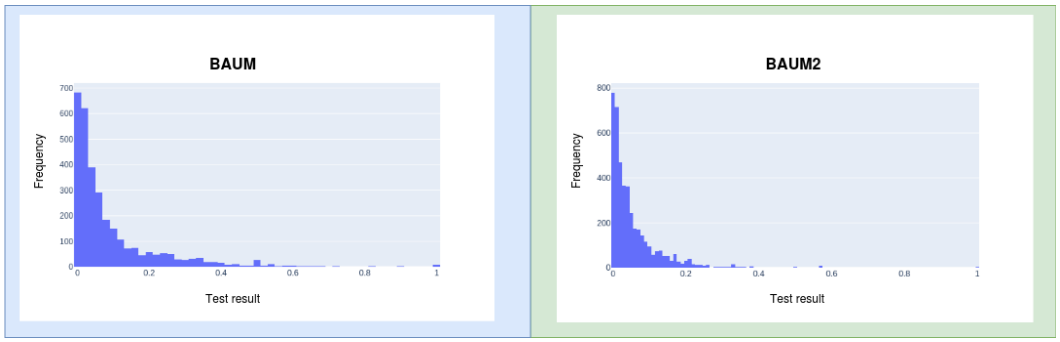


Fig. 4.9.: Chi square score distribution

According to the achieved results, the χ^2-importance scoring proved to be a easy-to-use statistical method in order to improve the spatial relation by taking into account the independence of the dialogue-state features with respect to the output action. This allows us to define W for the spatial-relations in a matter of minutes, without having to use costly genetic algorithms or methods which may be more constrained for industrial and/or rapid-prototyping environments.

On the whole, all the presented model smoothing methods improved the initial BAUM and BAUM2 (nearest state smoothing with Euclidean distance) at direct evaluation. The highest improvements were achieved when including the dialogue state pruning with SVMs, where hard baselines such as the Reg. Bi-LSTM were improved in terms of F1 score. One of the main advantage of the presented methods (K-Nearest State Smoothing, State Pruning and Spatial Learning with χ^2) is that they can be combined. These combinations are tested in a dialogue-generation setup in the next section.

4.6 DSTC2 - Indirect Evaluation

In addition to the direct evaluation methodology described in the previous section, where UM outputs were compared against the responses given by real users, an indirect evaluation has also been performed in order to measure the impact of the different smoothing strategies. Indirect evaluation aims to evaluate the behaviour of a UM and a DM by using dialogue-level metrics such as the Task Completion rate. To this end, an A-PFSBA UM and DM are set up talking to each other to generate dialogue samples. In this scenario, the smoothing strategy has been

changed from one experimental setup to another and its impact has been evaluated on the generated conversations.

The generated dialogues have been evaluated using the TC metrics introduced in Section 3.2.4.2. The TC metrics defined for the DM are:

- **Requested Informed:** The DM has informed about the items requested by the user, such as the venue address, phone and so on.
- **Canthelp Informed:** The DM has correctly informed about those constrains that are impossible to meet with the venues available in the database.
- **Valid Venue Offered:** The DM has offered a venue according to the constraints given by the user.

The TC metrics for the UM are:

- **Constraints Informed:** The UM has informed about the constraints that were given in the goal.
- **Request Information:** The UM has requested all the information slots (phone, address, etc.) set in the initial goal.
- **Request Alternatives (Reqalts):** The UM has asked for alternatives when a suitable venue has been offered by the system.
- **Bye**: The UM says goodbye when their goal is satisfied.

Note that some of these TC metrics may not be binary, i.e. if two constraints are given to the UM and only one of them is informed, the **Constraints informed** score will be 0.5. During dialogue generation, BAUM2 is used as the UM as it provides more complex interactions. The A-PFSBA DM introduced in Section 3.2.3 is used. Using these models, the impact of the different smoothing strategies over the generated dialogues are explored in the next section.

4.6.1 Corpus Reference

The DSTC2 corpus consists of a Human-Machine Interaction corpus where Mechanical Turkers had conversations with a system in order to find a suitable venue in Cambridge.

Table 4.9 and Table 4.10 show the TC rates achieved by the DM employed for the compilation of the DSTC2 corpus and the Mechanical Turkers in the different train/development/test partitions of the corpus.These numbers are taken as a baseline for the rest of the indirect evaluation experiments.

Tab. 4.9.: Task Completions scores achieved by the DSTC2 DM on the different partitions of the corpus

DSTC2 Dialogue Manager Task Completion over the DSTC2 corpus			
Task Completion	*Requested Informed*	*Canthelp Inform*	*Valid Venue Offer*
Train	0.92 ± 0.012	0.89±0.025	0.97±0.006
Dev	0.92 ± 0.022	0.78±0.06	0.96±0.014
Test	0.91 ± 0.016	0.89±0.029	0.97±0.008

Tab. 4.10.: Task Completion achieved by Mechanical Turkers on the DSTC2 corpus

Mechanical Turkers Task Completion				
Task Completion	*Constraint Inform*	*Request Information*	*Request Alternatives*	*Goodbye Score*
Train	0.92±0.19	0.92 ± 0.24	0.88±0.06	0.99±0.1
Dev	0.9±0.21	0.93 ± 0.24	0.93±0.25	0.98±0.15
Test	0.91 ± 0.26	0.91 ± 0.26	0.89±0.31	0.99±0.09

As it can be seen, scores are consistent on the training and the test sets both for the DSTC2 DM and the real users. On the other hand, the DSTC2 DM performs slightly worse on the development set overall, whereas real users perform slightly better in terms of task completion.

4.6.2 BAUM2 - Indirect Evaluation

With the corpus TC metrics set as a baseline for comparative purposes, an A-PFSBA UM and DM are evaluated using the same TC metrics for different model smoothing configurations. BAUM2 is evaluated first and fine-tuned according to the different smoothing strategies. The resulting UM is then further used to test the A-PFSBA DM.

The parameters of the baseline configuration for the dialogue generation task are shown in Table 4.11 both for the BAUM2 and the A-PFSBA DM. The A-PFSBA DM is trained using the *Training* dialogue set and the BAUM2 is trained on the *Test* to ensure that no dialogue sample is shared by the models. Dialogue filtering is not performed, i.e. all the dialogues of those sets are used to train the A-PFBSA models instead of using only those dialogues that complete all the TC metrics of the UM. In order to test the BAUM2 at functionality-level, no Speech To Text (STT) error is induced in the Natural Language Understanding (NLU) module during the

initial experiments. Random Sampling policy is used both as A-PFSBA policy and for smoothing. In practice, these parameters set a baseline configuration where none of the presented model smoothing contributions are used.

In order to generate the dialogue goals G of the BAUM2 during these experiments, the probabilities of the goal constraints and requests of the original corpus are sampled. Then, the new goals of the BAUM2 for each dialogue are set by sampling these probabilities.

Tab. 4.11.: Baseline Configuration BAUM2 Dialogue Generation Parameter Setting

Parameter	User Model - BAUM2	Dialogue Manager
Regular Policy	Random Sampling	Random Sampling
Spatial Relation	Cosine Similarity	Cosine Similarity
Dialogue Smoothing - Policy	Random Sampling	Random Sampling
Dialogue Smoothing - States	1	1
Chi-square spatial	No	No
Pruning Method	None	None
Train Partition	Test	Train
Filter Dialogues	No	No
NLU error - n_rounds	–	–
NLU error - $\alpha_{corrupt}$	–	–

Dialogue Generation - BAUM2 Baseline Smoothing Configuration

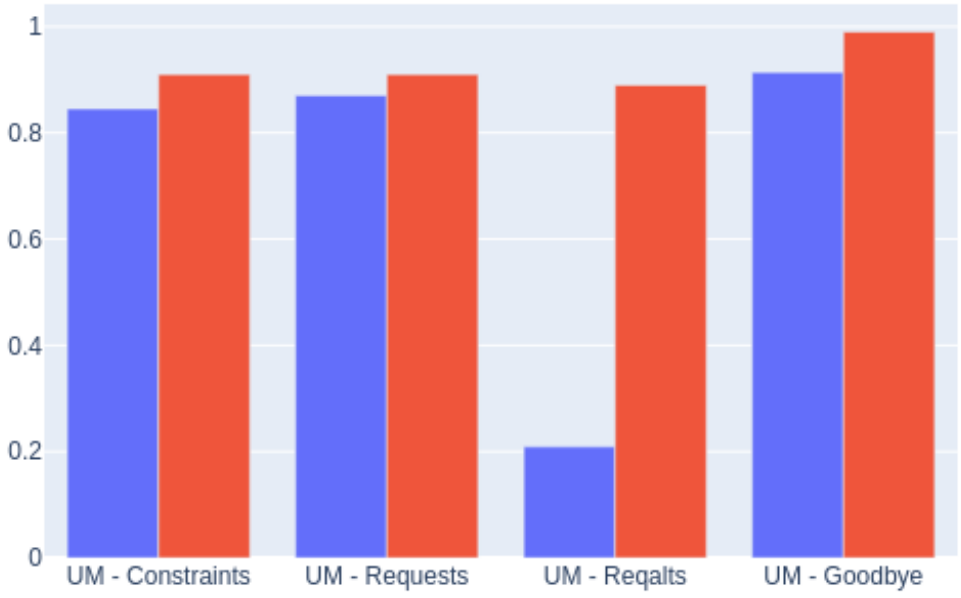


Fig. 4.10.: User TC metrics over the dialogues generated with the BAUM2 and A-PFSBA DM using the Baseline Configuration (blue) compared to the results of the TC achieved by the Mechanical Turkers in the DSTC2 test set (red)

Figure 4.10 shows the results achieved by the BAUM2 in a dialogue generation setup with the A-PFSBA DM without using the proposed model smoothing techniques. To obtain these metrics, a total of 1000 dialogues were simulated and evaluated according to the defined TC tests. On the whole, the BAUM2 without model smoothing performs slightly worse than Mechanical Turkers informing constraints, requesting info and saying goodbye to the system. The worst performing metric is the *Request Alternatives* metric, i.e. the BAUM2 does not ask for alternative venues and tends to go with the first offered venue.

Tab. 4.12.: BAUM2/ A-PFSBA DM Smoothing parameters for grid-search optimization

Parameter	Values
Regular Policy	Random Sampling
Spatial Relation	Cosine Similarity
Dialogue Smoothing - Policy	Nearest State / TSA - 0.8
Dialogue Smoothing - States	[1, 5, 10, 15, 20]
Chi-square spatial	Yes, No
Pruning Method	None, SVM
Pruning Thresholds	0.05, 0.075, 0.1, 0.15, 0.2, 0.25
Train Partition	Test
Filter Dialogues	Yes, No
NLU error - n_rounds	–
NLU error - $\alpha_{corrupt}$	–

Tab. 4.13.: BAUM2 Optimized Smoothing Configuration parameters

Parameter	Best Value
Regular Policy	Random Sampling
Spatial Relation	Cosine Similarity
Dialogue Smoothing - Policy	TSA - $\mu_{prune} = 0.8$
Dialogue Smoothing - States	10
Chi-square spatial	No
Pruning Method	SVM
Override Thresholds	0.2
Train Partition	Test
Filter Dialogues	Yes
NLU error - n_rounds	–
NLU error - $\alpha_{corrupt}$	–

Table 4.13 shows the BAUM2 smoothing strategy parameter configuration optimized to achieve the best TC score on the grid-search. Best-performing parameters are coherent with the results achieved in previous model smoothing experiments in Section 4.5. The combination of the TSA smoothing policy and SVM-based state pruning rendered best results and dialogue filtering was also useful. Figure 4.11 shows the best results achieved. Although every score increases when the model smoothing strategies are improved, the **UM-Reqalts** TC rate shows the highest improvement. This can be explained by the inclusion of both the SVM and the TSA mechanisms in the model smoothing strategy. The first one takes the explicit goal

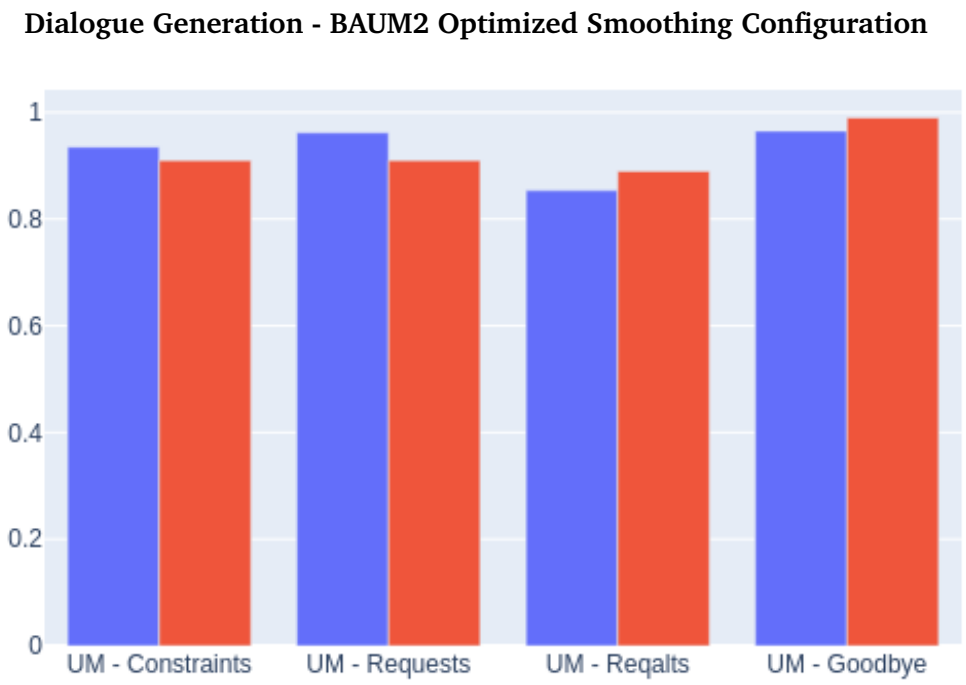


Fig. 4.11.: User TC metrics over the dialogues generated with the BAUM2 and A-PFSBA DM by using the Optimized Smoothing Configuration for BAUM2 (blue) compared to the results of the TC achieved by the Mechanical Turkers in the DSTC2 test set (red)

representation into account in order to sample semantically-related dialogue states. The second one makes it more likely to trigger a request alternative action once a valid venue is offered to the user, because it increases the verbosity of the UM.

This configuration is used for further model smoothing experiments with the DM, as it provides a suitable UM for the dialogue generation setup of the indirect evaluation.

4.6.3 A-PFSBA DM - Indirect Evaluation

Once the best smoothing configuration is found for the BAUM2, the same process is carried out for the A-PFSBA DM.

With zero STT error induced over the NLU, and using the Optimized Smoothing Configuration (OSC) BAUM2 of Table 4.13 and the A-PFSBA DM with the Baseline Smoothing Configuration (BSC) of Table 4.11, the following results are achieved:

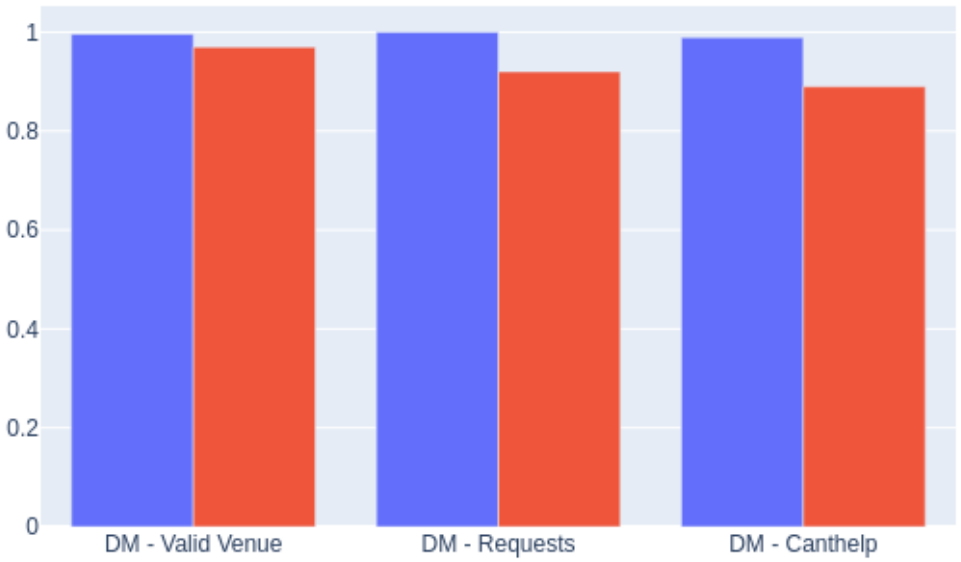


Fig. 4.12.: System TC metrics over the dialogues generated with the BAUM2 by using the Optimized Smoothing Configuration and the A-PFSBA DM using the Baseline Smoothing Configuration (blue) compared to the results of the TC achieved by the DSTC2 DM in the train set (red)

The DM with baseline smoothing parameters achieved almost perfect results when there is no STT-induced error.

4.6.3.1 DM - STT Noise Robustness

STT-induced NLU error is a phenomenon that almost every SDS needs to handle. To test if the presented smoothing techniques help the DSTC2 DM to overcome uncertainty, the smoothing configuration has been optimized for mild-level STT error scenario. To do so, $n_{rounds} = 10$ and $\alpha_{corrupt} = 1$ has been set in the NLU error module described in Section 3.2.3.5. Then, a grid-search has been performed on the different model smoothing configurations for the A-PFSBA DM, using the parameters of Table 4.12 rendering the optimal combination of Table 4.14.

Similar patterns arise when comparing the best configuration of the BAUM2 and the DM, where the TSA smoothing policy and the SVM-based state-pruning are present in both cases. This pattern can be justified as the SVM-based state pruning removes unrelated dialogue-states and the TSA policy contextualizes the selection of the next action by using several nearest dialogue states.

Then, to visualize the impact of the smoothing techniques over the A-PFSBA DM when different levels of STT-induced NLU error, the **baseline A-PFSBA DM** (the

Tab. 4.14.: Optimized Smoothing Configuration for the A-PFSBA DM for Dialogue Generation

Parameter	Best Value
Regular Policy	Random Sampling
Spatial Relation	Cosine Similarity
Dialogue Smoothing - Policy	TSA - 0.7
Dialogue Smoothing - States	10
Chi-square spatial	No
Pruning Method	SVM
Override Threshold	0.075
Train Partition	Train
Filtered Dialogues	Yes
NLU error - n_rounds	10
NLU error - $\alpha_{corrupt}$	1

DM with Baseline Smoothing Configuration of Table 4.11) and the **fine-tuned A-PFSBA DM** (the DM with the Optimized Smoothing Configuration of Table 4.14) are compared. This comparison can be viewed in Figures 4.13, 4.14 and 4.15, where the three established TC metrics are presented (y axis) depending on the $\alpha_{corrupt}$ parameter (x axis). Remember that the lower $\alpha_{corrupt}$ is, the higher the induced NLU error will be. The amount of times that the original NLU is corrupted is the same for every point: $n\_rounds = 10$.

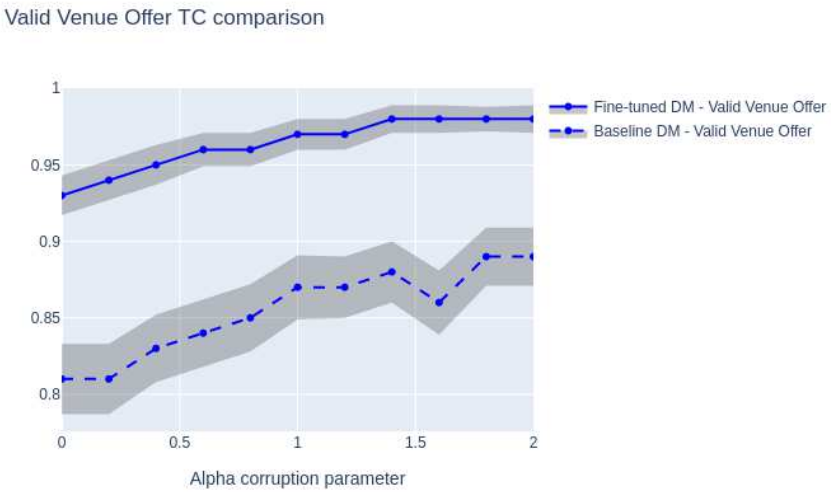


Fig. 4.13.: TC rate of the Valid Venue Offered metric for both the baseline and the fine-tuned A-PFSBA DMs

In order to ensure statistical significance, the 95% error interval is shaded. As it can be seen, for the Request and the Invalid Constraints task completions there is no statistical difference at all, but the fine-tuned smoothing configuration renders better TC when it comes to offer a valid venue according to the user's informed

constraints. In addition, the error intervals of the fine-tuned smoothing configuration are narrower than the baseline ones for the Valid Venue Offer TC metric, which is one of the main goals of the SDS.

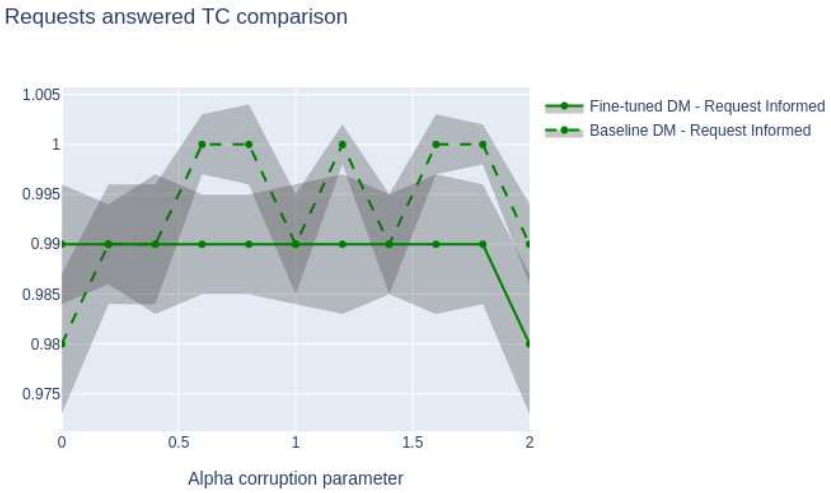


Fig. 4.14.: TC rate of the Request Informed metric for both the baseline and the fine-tuned A-PFSBA DMs

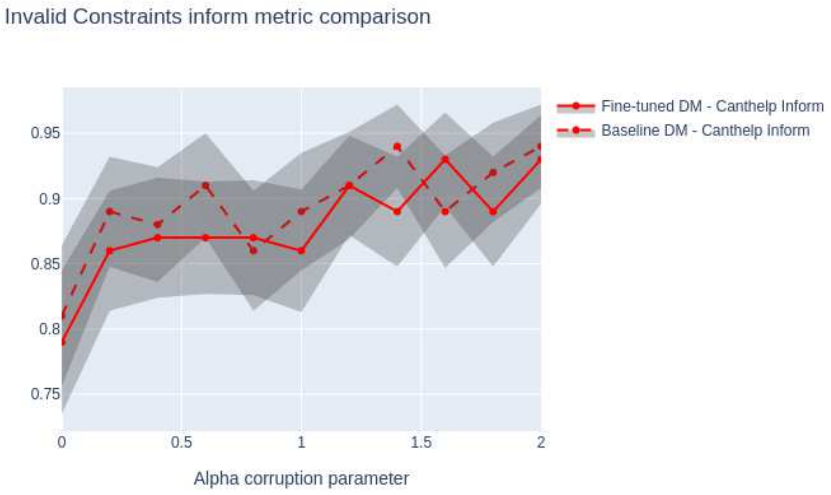


Fig. 4.15.: TC rate of the Canthelp Informed metric for both the baseline and the fine-tuned A-PFSBA DMs

As a concluding remark, the presented model smoothing methods improved the initial models (nearest state smoothing with Euclidean distance) at indirect evaluation, improving the behavior of the models when performing an holistic evaluation with TCs. The improvements were consistent even when STT induced error was present in the dialogues. Similar to the results achieved in the direct evaluation, both optimized smoothing configurations for the BAUM2 and A-PFSBA DM include

contextualising the smoothing strategy with K nearest states and pruning the states that were not relevant.

5 Exploitation Policies

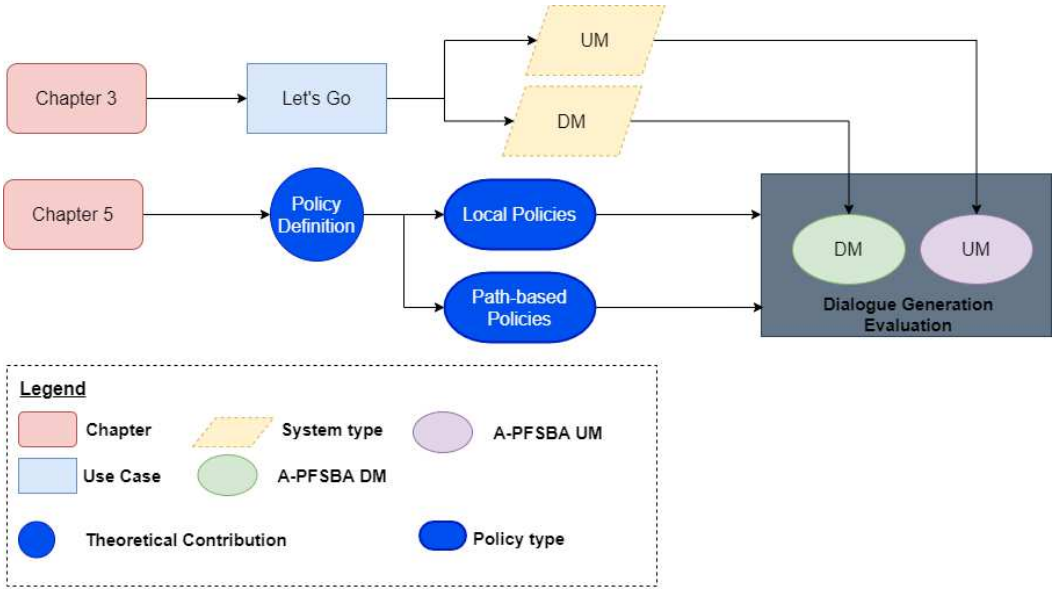


Fig. 5.1.: Exploitation Policies chapter map.

As depicted in Figure 5.1, this chapter introduces a theoretical framework based on the Attributed Probabilistic Finite State Bi-Automata (A-PFSBA) formulation for policy making. These policies define how the A-PFSBA structural model can be employed in order to build decision strategies that take into account both user and system. Then, different local and path-based policies are implemented both for User Model (UM) and Dialogue Manager (DM) on the Let's Go corpus using the proposed formulation. Finally, their behaviour is analysed and their results are compared against local policies.

5.1 Policy Definition

In a decision process model Sutton & Barto defined (Sutton and Barto, 1998) the policy as:

> A policy defines the learning agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of the environment to action to be taken when in those states.

When it comes to spoken dialogue interaction, the agent would be the DM, which is a decisional process. In the A-PFSBA framework the perceived states are the dialogue states $q_j \in Q$ since they encode the user decoded actions at the current turn $\tilde{d}_t \in \Sigma^{\leq m}$, the previous system actions $\tilde{a}_{t-1} \in \Delta^{\leq n}$ and the attributes $\omega_t \in \Omega$ that define the dialogue memory. The actions to be mapped are the system actions $\tilde{a} \in \Delta^{\leq n}$. Then, the policy Π corresponds to a mapping from each system dialogue state $q_j \in Q_S$ to the set of system actions Δ.
The policy Π can be represented in multiple forms, either deterministically from the current state (Bohus and Rudnicky, 2003; Bohus and Rudnicky, 2009) or stochastically over the set of the possible actions (Griol et al., 2008; S. Young, 2006; Thomson, K. Yu, et al., 2010). More generally, it can also be seen as a ranking problem, where the policy Π associates a score to each system action given the current dialogue state, in a way similar to reinforcement-learning methodologies (Schütze et al., 2008).
In the A-PFSBA framework, the policy corresponds to a function $\Pi_{A-PFSBA}$ that maps the current system dialogue state $q_j \in Q_S$ and the set of possible actions that label each transitions of $\delta(q_j)$

$$\delta(q_j) = \{\exists\ (q_j,\ (\epsilon\ :\ \tilde{a}_k),\ q')\} \subseteq Q_S \times \Gamma \times Q_U$$

Figure 5.2 shows a situation where four possible actions $\{\tilde{a}_1, \tilde{a}_2, \tilde{a}_3, \tilde{a}_4\}$ that label the transitions of $\delta(q_j)$ can be chosen to continue with the interaction from state q_j. The objective of a policy Π is to choose the best action to complete the task of the DM.

As the A-PFSBA formulation captures the transitions of both system and user actions, user information can be easily exploited in order to determine path-based policies when defining decision strategies. Note that this formulation is symmetrical for UMs, by swapping the associated alphabets.

5.1.1 Path-Based Policies

Path-based policies can be defined as a scoring function over an A-PFSBA path of states $\xi = (q_{start}, q_1, \cdots, q_{end})$ with depth $D = |\xi| - 1$ where $q_i \in Q$. In this path ξ, q_{start} denotes the departing state and q_{end} is the last state of the path.

The score associated with a given path or path-value $V(\xi)$ needs to take into account every taken step, the differences between the departure and the final states (q_{start} and q_{end}), and the length of the path (since more distant actions should have lesser

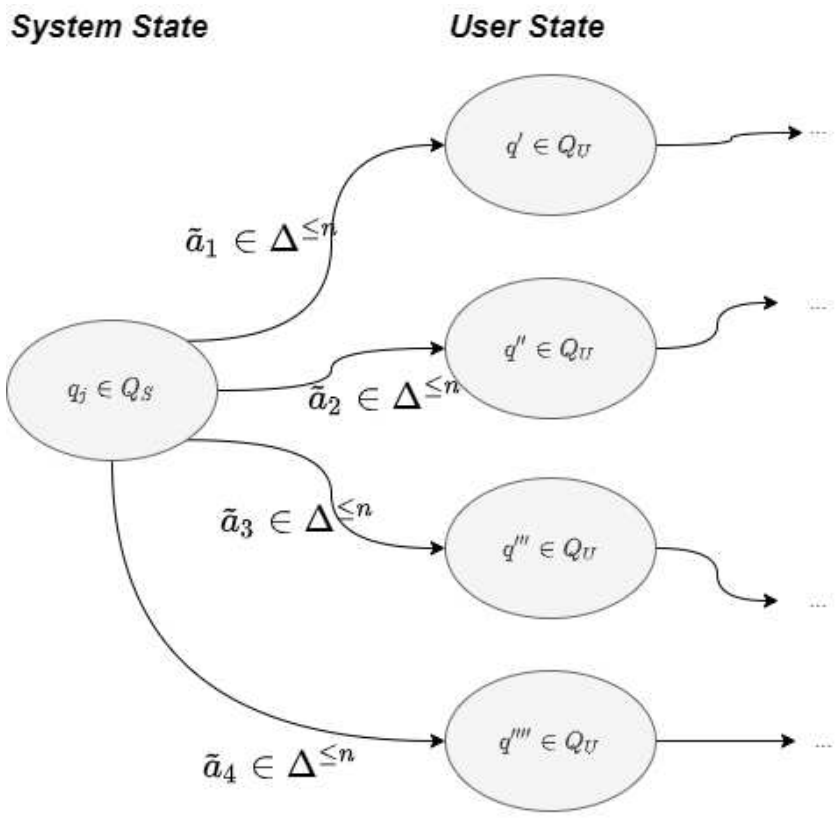


Fig. 5.2.: Dialogue representation of a state $q_j \in Q_S$ where four possible actions may be chosen to keep on with the interaction.

impact). These properties can be summarized in the following path-value function:

$$V(\xi) \;=\; \psi(q_{start}, q_{end}) \cdot \frac{\lambda}{|\xi|} \prod_{i=0}^{D} \gamma^{i} \;\cdot \phi(q_i, q_{i+1}) \tag{5.1}$$

where the function $\psi(q_{start}, q_{end})$ is the endpoint-value function that evaluates the differences between the departure state q_{start} and the final state q_{end} of path ξ:

$$\psi : Q \times Q \rightarrow \mathbb{R} \tag{5.2}$$

In practice, the ψ function can be any scorer that weighs the difference between the departing and ending dialogue states according to the task the DM needs to fulfill. Then, λ is the length normalization factor that determines the penalisation of the dialogue length and $\gamma < 1$ is the discount factor that controls the temporal decay. ϕ is the step-value function that associates the score of transitioning from state q_i to q_{i+1}. The step-value function can be defined separately for user-taken steps ϕ_U and system-taken steps ϕ_S:

$$\phi(q_i, q_{i+1}) = \begin{cases} \phi_U(q_i, q_{i+1}), & if\ q_i \in Q_U\ and\ q_{i+1} \in Q_S \\ \phi_S(q_i, q_{i+1}), & if\ q_i \in Q_S\ and\ q_{i+1} \in Q_U \end{cases}$$

(5.3)

Then, the system action $\tilde{a}_{t+1}$ to select as next response from a departure system state q_s is the one that maximizes the expected path-value of all the possible paths ξ that depart from q_s by means of system action $\tilde{a}_{t+1}$.

$$\tilde{a}_{t+1} = \texttt{argmax}_{\tilde{a}_k \in \delta(q_s)} \frac{1}{|\Xi_{q_s,\tilde{a}_k}|} \sum_{\xi \in \Xi q_s,\tilde{a}_k} V(\xi)$$

(5.4)

where $\Xi_{q_{start},\tilde{a}_k}$ is the set of paths ξ that start in state q_{start} and perform system action $\tilde{a}_k$ as the first action.

Under this formulation, local policies can be represented as a subset of path based policies, i.e. those paths ξ that contain only the departure and final states $|D| = 1$. This type of policies are suitable for simple domains and/or when faster and computationally less expensive policies are sought after.

5.2 Let's Go Experimentation Framework

Results on the Let's Go corpus have been evaluated using the Task Completion (TC) metric described in Chapter 3. This metric measures if there is enough information to perform a query to the bus route searching backend and whether the query information is returned to the user. Additionally, the Average Dialogue Length (ADL) is used to measure the number of turns it takes to end a conversation[1].

5.2.1 A-PFSBA Policies

In this section, the DM and UM exploitation policies implemented on the A-PFSBA model trained on the Let's Go corpus are described in more detail.

[1]The ADL metric considers each user or system intervention as a turn.

5.2.1.1 Dialogue Management Policies

Four DM policies have been defined following the presented A-PFSBA policy notation. All of them use the step-value function described in Equation 5.5 next:

$$\phi(q_i, q_{i+1}) = \begin{cases} \phi_U(q_i, q_{i+1}) = \frac{P((q_i,(\epsilon{:}\ \tilde{a}_k),\ q_{i+1})\ )}{P((q_i,(\epsilon{:}\ \tilde{a}_k),\ q_{i+1})\ )^{1-\beta}} \\ \phi_S(q_i, q_{i+1}) = P((q_i, (\epsilon:\ \tilde{a}_k),\ q_{i+1})\ ) \end{cases} \tag{5.5}$$

This is, the transition probability is used to score system-steps and a β-normalized transition probability for user-steps.

The user awareness parameter $\beta \in [0, 1]$ weighs the user transition probability in the scoring function $\phi(q_i, q_{i+1})$. When $\beta = 0$, the a-priori probability is ignored: every transition probability has the same weight, so, $\phi_U(q_i, q_{i+1}) = 1$. On the other hand, when $\beta = 1$ the user transition probability is taken into account in the scoring function and more probable user-actions achieve a higher score.

Using this step-value function, the following policies have been defined, where the main change is the endpoint-value function of Equation 5.2:

- **Maximum Probability Path (MPP):** chooses the path of system actions with maximum probability. Its endpoint-value function is:

$$\psi(q_{start}, q_{end}) = 1$$

- **Maximum Probability (MP):** It is a local version of the **MPP** policy, i.e. $D = 1$.

- **Attributed Path (AP):** It chooses the path with highest probability and also searches to complete as many dialogue attributes as possible. The endpoint-value function of Equation 5.2 is changed to:

$$\psi(q_{start}, q_{end}) = \frac{1}{1 + (|\omega_{q_{end}}| - |\omega_{q_{start}}|)}$$

  where $\omega_{q_{start}}$ and $\omega_{q_{end}}$ are the attributes of the initial and the final states, respectively.

- **Task Completion Path (TCP):** It chooses the path with highest score according to the Task Completion rate, i.e. the path that satisfies most constraints

in order to consider a dialogue successful. The endpoint-value function is modified to:

$$\psi(q_{start}, q_{end}) = \frac{1}{1 + (TCS(q_{start}, q_{end}))}$$

where $TCS(q_{start}, q_{end})$ is a scoring version of the TC metric shown in Algorithm 6. For this scorer, a simple mechanism is used, per each condition satisfied of the TC, 0.25 is added to the score. As an exapmle, if from q_{start} to q_{end} three conditions are satisfied (departure place known, arrival place known and departing time known), then $TCS(q_{start}, q_{end}) = 0.75$.

In this policy, instead of guiding the dialogue to simply fulfill attributes, the dialogue manager selects those actions that guide the interaction to satisfy the constraints needed to complete the task.

In order to estimate the best action $\hat{a}$ for each system state, every possible path over the A-PFSBA structure that starts in state q_s and performs $\tilde{a}$ as the first action $\Xi q_s, \tilde{a}_k$ would need to be calculated. As this is computationally intractable, randomized sampling is used to generate multiple paths by using their transition probabilities.

To better measure the impact of user uncertainty in the structure of the A-PFSBA model, each path-based policy is evaluated performing a grid search over the user-awareness rate $\beta \in [0, 1]$ and the depth D parameters.

5.2.1.2 User Modelling Policy

For the UM, a Random Sampling (**RS**) policy has been implemented. The objective of this policy is to be non-deterministic in order to account for the variability of the user. The user action to perform $\tilde{d}_{t+1}$ is randomly sampled from the distribution of user actions seen in the current state. So as to control the score-distribution, an α_{um} parameter is used. Being $P((q_i, (\tilde{d} : \epsilon), q_j))$ the probability of transition $(q_i, (\tilde{d} : \epsilon), q_j) \in \delta_U$, this probability can be re-scored by α_{um} as follows:

$$\alpha_{um}(P((q_i, (\tilde{d} : \epsilon), q_j))) = \frac{P((q_i, (\tilde{d} : \epsilon), q_j))^{\alpha_{um}}}{\sum_k P((q_i, (\tilde{d} : \epsilon), q_j))^{\alpha_{um}}}$$

Thus, if α_{um} is 1, the next UM action is sampled using the original probability distribution. On the other hand, if $\alpha_{um} = 0$, all the transition probabilities will be the same and, consequently, the next action sampling will ignore the prior probabilities of the A-PFSBA model. This parameter can be adjusted to test the dialogue generation behavior depending on the degree of randomness of the user.

5.2.2 Results

Exploitation policy evaluation has been carried out on a dialogue-generation setup. A DM and a UM are deployed in order to generate dialogues by talking to each other. Then, such dialogues are evaluated in terms of TC and ADL. In order to train the A-PFSBA DM and UM, the corpus has been split into two, so there is no dialogue sample overlap when building the corresponding DM and UM A-PFSBA models. Table 5.1 shows the achieved results.

Tab. 5.1.: Results achieved by the different policies on the Let's Go Corpus. The same metrics on the corpus are also depicted with the previous baselines.

	DM Model	Policy DM	Policy UM	Task Completion (%)	Average Dialogue Length
Corpus	*CMU RavenClaw*	Rules	–	54	32.33 ± 1.2
Baseline(Orozko and M Inés Torres, 2015)	*PFSBA*	MP	Random Choice	20.08	29.23 ± 0.28
	A-PFSBA	MP	Random Choice	31.53 ± 1.54	31.39 ± 0.722
	A-PFSBA	MP	Random Sampling	60.02 ± 1.36	30.98 ± 0.94
	A-PFSBA	MPP	Random Sampling	59.3 ± 0.6	32.2 ± 0.3
	A-PFSBA	ADL	Random Sampling	59.5 ± 0.6	32.8 ± 0.3
	A-PFSBA	TCP	Random Sampling	61.2 ± 0.6	32.5 ± 0.3

For illustrative purposes, the first row shows results achieved by the RavenClaw DM, which is the rule-based DM used in the Let's Go Corpus. This row can be seen as the initial evaluation of the corpus. In the second row, the **PFSBA** DM model shows results achieved by previous **unattributed** bi-automata works (Orozko and M Inés Torres, 2015). Note that the policy used by the UM in this first setup is **Random Choice**, which is equivalent to the **Random Sampling** policy with $\alpha_{um} = 0$, i.e. all the possible transitions of the UM have the same probability of being chosen as the next action.

As it can be seen, the inclusion of attributes in the PFSBA formulation clearly improves the results of the DM, which improved more than 10 points compared to the un-attributed **PFSBA** model. In addition, swapping the UM policy from Random Choice to Random Sampling (i.e. changing $\alpha_{um} = 0 \rightarrow \alpha_{um} = 1$) improves the TC rate in almost 30 points. When it comes to path-based policies, in general the differences in TC rate are not significant, with **TCP** performing slightly better than **MPP** and **AP**, but without statistical significance regarding the local MP policy, as there is an overlap in their confidence intervals (95% confidence interval). The slight improvement over the TC rate of **TCP** can be put down to the inclusion of external information in the dialogue policy. Nevertheless, there is no statistical significance between the **TCP** and **MP** policies.

Regarding the ADL metric, the local **MP** policy tends to generate slightly shorter dialogues. This is usually better than long dialogues[2] in task-oriented dialogue systems as it is the case for the Let's Go scenario. Nevertheless, a difference of 1 turn can be considered negligible from the point of view of the end users.

For better evaluation of the path-based policies, a grid-search evaluation has also been performed on the two parameters that need to be fine-tuned in these policies: the path-depth D and the user-awareness rate $\beta \in [0, 1]$. As explained before, the path-depth D determines how much future steps the policies take into account and the user-awareness rate β represents the relevance given to the user transition probabilities in the scoring function of the policy. $\beta = 0$ means that this probability is not taken into account. Instead, if $\beta = 1$, the transition probabilities of the A-PFSBA model are used in the step-value function of Equation 5.5. These parameters have a direct impact on the performance of path-based policies.

The following figure shows the TC score achieved by the three path-based policies for different combinations of $D \in [1, 35]$ and $\beta \in [0, 1]$:

[2] In social dialogue systems the longer the dialogue the better, as their goal is to maximize the user engagement with the system.

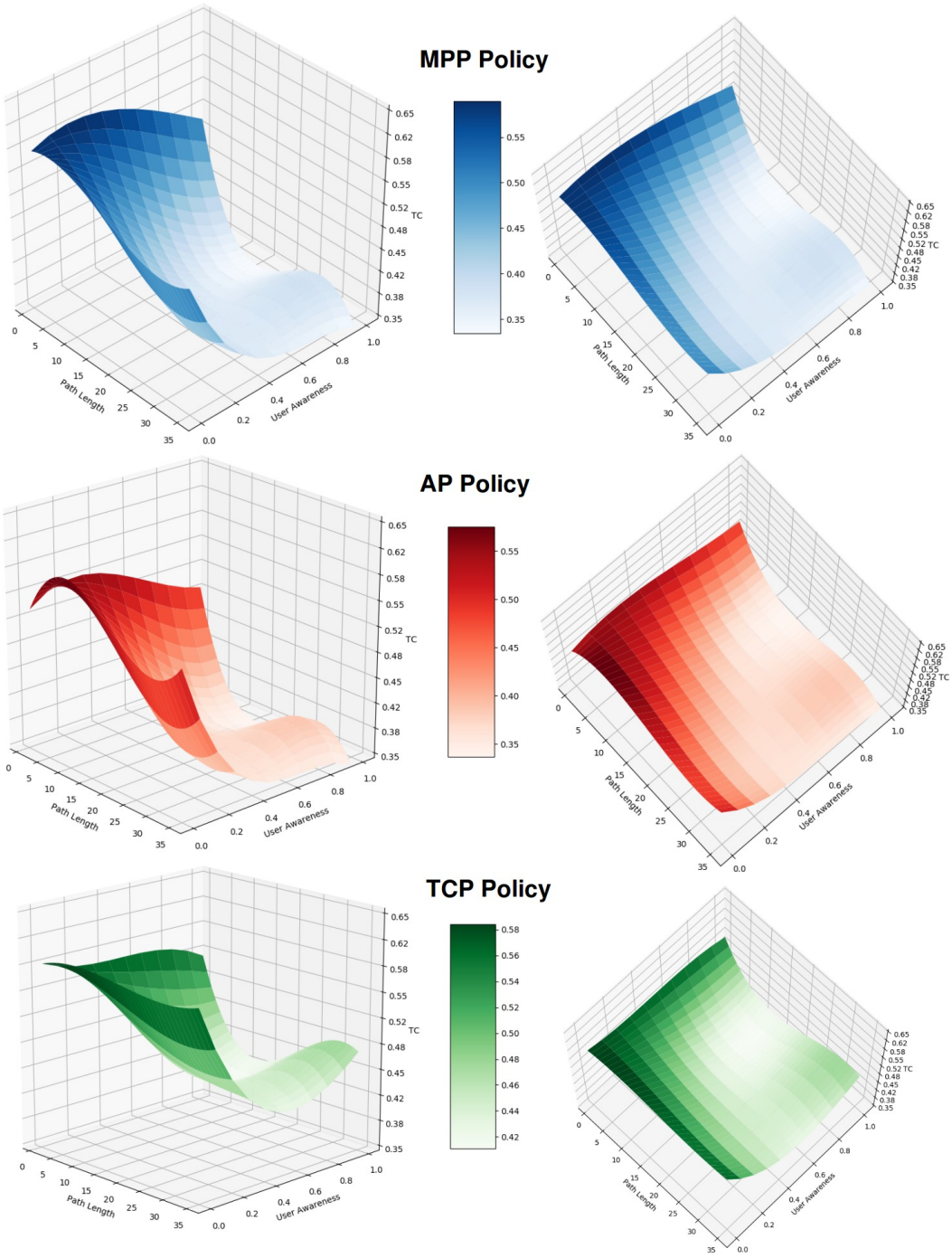


Fig. 5.3.: Spline-smoothed plots of the TC rate of the path-based policies with different perspectives of the same plot. The left-axis indicates the depth of the path D, the right axis indicates the value of the user awareness parameter β and the z-axis indicates the TC rate. Left: front view, right: top view

The left-axis indicates the depth of the path D, the right axis indicates the value of the user awareness parameter β and the z-axis indicates the TC rate. These graphs are spline-smoothed for a better visualisation of the trends. Each point of the graph is calculated 5 times, and the mean is used as value.

Policies that perform worse when β is set to 1 than when $\beta = 0$ indicate that the user transition probabilities of the A-PFSBA model are not correctly estimated. Also, the variability of the TC rate conditioned over the path-depth D indicates how well the A-PFSBA model is fitted to the user. Long paths performing worse than short paths signal that the model is not taking into account paths that the user commonly employs.

For all the implemented policies, the relation between path length and user-awareness rate is clear: long and user-aware paths perform worse. This conclusion validates the hypothesis of (Ghigi and M Inés Torres, 2015) that path-based policies perform worse than local policies in general due to user uncertainty. In addition, it is clear that the initial models are not fitted to the user, as the TC rate gets worse when the path length is increased. Also, it is interesting to note that the **TCP** policy has a higher low-boundary at the TC rate. This can be attributed to the inclusion of external information such as the Task Completion score, which reduces the amount of decay introduced by path length and user uncertainty.

6 Incremental Learning

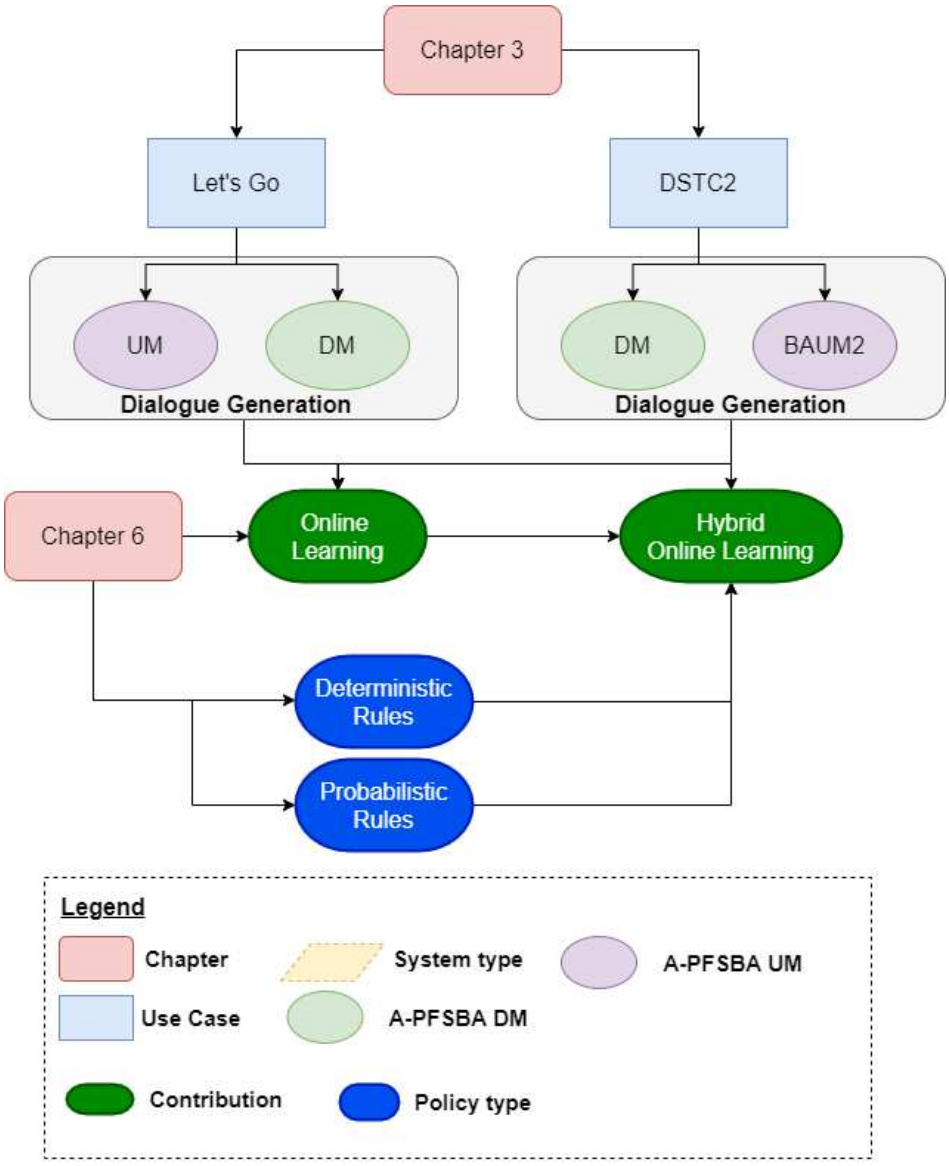


Fig. 6.1.: Incremental Learning Chapter Map

Incremental Learning (IL) or the ability to continuously improve initial models is a desired property of Machine Learning (ML) frameworks. Under the Attributed Probabilistic Finite State Bi-Automata (A-PFSBA) formulation, this can be done by exploiting the new dialogue states and transitions created during model smoothing.

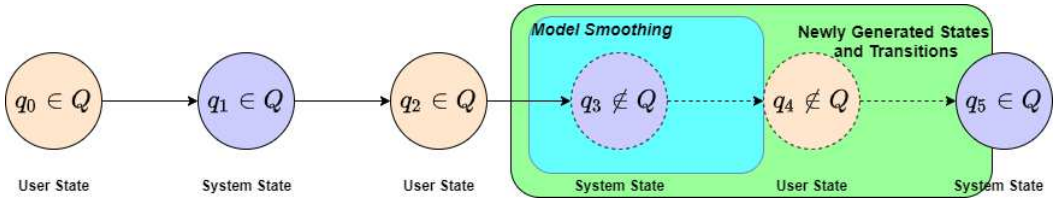


Fig. 6.2.: New dialogue states and transitions generated when performing the model smoothing over the unknown state $q_3 \notin Q$

In this Chapter, as depicted in Figure 6.1, the Online Learning (OL) algorithm is proposed to ensure proper decision-making when learning incrementally under the A-PFSBA formulation. In addition, a Hybrid Online Learning (HOL) method is also presented. This method hybridizes a rule-based Dialogue Manager (DM) and an A-PFSBA based DM, which is incrementally learned using the OL algorithm. This method makes it possible to initialise a rule-based DM in zero data scenarios and to improve the corresponding A-PFSBA model by exploiting the generated dialogues through OL.

Using the presented OL and HOL methods, three hypotheses are formulated and tested in different dialogue generation settings:

1. *The DM can be improved using OL on a dialogue-by-dialogue basis*.

2. *OL is suitable even when the amount of initial dialogue samples is relatively small*.

3. *Hybrid Online Learning makes it possible to build a data-driven DM by using a rule-based DM initialisation*.

The first hypothesis is tested over the Let's Go corpus. The second and the third ones are tested over the Dialogue State Tracking Challenge 2 (DSTC2) corpus.

6.1 Incremental Learning over the A-PFSBA

Under the A-PFSBA formulation, incremental learning of the DM can be done by exploiting the new dialogue states and transitions created during model smoothing. As an illustrative example, Figure 6.2 shows the new states and transitions when applying the smoothing strategy to the unknown state $q_3 \in Q$. Previous work to this dissertation showed that the A-PFSBA model was able to learn new dialogue states and transitions on the fly on a turn-by-turn basis (Orozko and M Inés Torres, 2015). Such learning was performed each time the model smoothing was carried out, leading to a smoothed version of the initial DM. Unfortunately, this approach was unable to ensure the learning of a proper decision-making strategy on a turn-by-turn

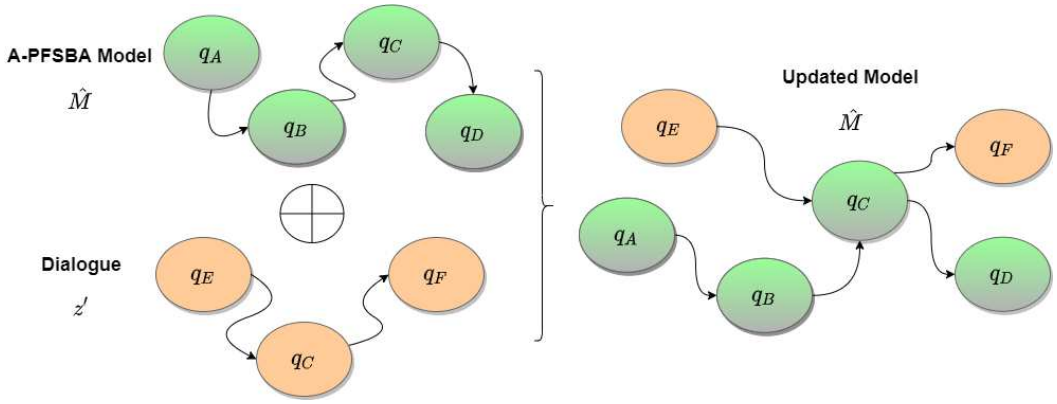


Fig. 6.3.: Online Learning from an initial A-PFSBA model $\hat{M}$ and a new dialogue $\mathbf{z}'$

basis. Thus, the resulting smoothed DM was unable to improve the Task Completion (TC) rate. In order to avoid learning non-beneficial patterns, and as described in Chapter 4, the A-PFSBA formulation employed in this dissertation does not perform this learning each time the model smoothing is triggered.

In this chapter, instead of performing the model learning on a turn-by-turn basis, a dialogue-level method is presented, called Online Learning algorithm. This algorithm ensures that a proper decision making has been done at dialogue-level before updating the initial A-PFSBA model.

6.2 Online Learning

Being a generative model, the A-PFSBA formulation enables to transform its structural modelling and properties as shown in (Orozko and M Inés Torres, 2015). In practice, this means that new dialogue states and transitions can be added to the initial A-PFSBA model $\hat{M}$.

When interacting with real users, unseen situations arise, which lead the interaction to states that are not in the initial A-PFSBA model $\hat{M}$. As described in Chapter 4, a smoothing strategy is defined over the initial model $\hat{M}$ in order to continue with the interaction. However, these new dialogue states and transitions are not learnt during smoothing, as proper decision making cannot be ensured (Orozko and M Inés Torres, 2015). To overcome the problem of learning dialogue states and transitions which render bad decision-making, the presented OL algorithm employs a Quality Metric (QM) to determine whether a dialogue is suitable for learning or not. Using this metric, the A-PFSBA model learns only from those dialogues rendered successful by the QM. In practice, this QM can be based on automatic metrics such as TC or assessed by soft human supervision.

A formal description of the OL algorithm; let $\hat{M}$ be the initial A-PFSBA model inferred from Z dialogue samples, let $\mathbf{z}' \notin Z$ be an unseen dialogue sample and $\hat{M}_{\mathbf{z}'}$ the A-PFSBA model inferred from the single sample $\mathbf{z}'$. If the QM renders $\mathbf{z}'$ valid for the learning process, $\hat{M}$ is expanded by merging it with $\hat{M}_{\mathbf{z}'}$. By doing so, the states q_j and the corresponding set of transitions $\delta(q_j) = \{(q_j, (\tilde{d}_i : \tilde{a}_i), q')\}$ of $\hat{M}_{\mathbf{z}'}$ are added to $\hat{M}$. The online learning pseudo-algorithm is defined as follows:

Algorithm 5 Online Learning

$\hat{M} \leftarrow$ *A-PFSBA from samples* Z
$\hat{M}_{\mathbf{z}'} \leftarrow$ *A-PFSBA from new sample* $\mathbf{z}'$
if $QM(\mathbf{z}')$ **is** True **then**
    for $q_j \in Q_{\mathbf{z}'}$ **do**:
        $\hat{M} \leftarrow merge(\hat{M}, q_j, \delta(q_j))$
        $\hat{M} \leftarrow update\_edge\_count(\hat{M})$
    end for
end if
return $\hat{M}$

Figure 6.3 shows an unseen dialogue $\mathbf{z}'$ rendered valid by a QM. As a result, the initial A-PFSBA DM model is augmented with the new dialogue states and transitions seen in the dialogue sample $z' \notin Z$.

In practice, this OL algorithm adds the unseen states and transitions generated during the model smoothing (See Figure 6.2) and updates the transition probabilities of the already existing transitions.

6.2.0.1 Online Learning Under Uncertainty

As described in Chapter 4, the model smoothing strategy is applied every time an unknown dialogue state $q' \notin Q$ is reached. In this situation, a distance (or similarity) function is used to find the closest known dialogue state $q_j \in Q$. In scenarios where channel noise n_t is present, any small perturbation of a known state q_j renders an unknown state: $\tilde{q}_j \notin Q$, even when $\tilde{q}_j \approx q_j$. This effect increases the sparsity of an updated A-PFSBA model when OL algorithm is used.

In order to overcome this effect, a radius r_ϵ is defined so that, before adding a new unseen dialogue state $q' \notin Q$ to the A-PFSBA model $\hat{M}$, the following decision is made:

$$\begin{cases} \text{swap } q' = q_j \in Q, & \text{if } dist(q', q_j) \leq r_\epsilon \; \forall q_j \in Q \\ \text{use } q', & \text{otherwise} \end{cases}$$

Where $dist$ is a distance function used at the model smoothing strategy to find the nearest dialogue states. This mechanism avoids adding new dialogue states q' if they are too close to any already existing dialogue state of the initial model. Note that this comparison is only made with system dialogue states $q_j \in Q_S$ for DMs and with user dialogue states $q_j \in Q_U$ for UMs.

6.3 Hybrid Online Learning

In order to learn a DM incrementally using the OL algorithm, an initial A-PFSBA DM is required. Unfortunately, the reality between the scientific community and the industrial environments highly differs in the amount of available data to build an initial DM, which for industrial environments is usually zero.

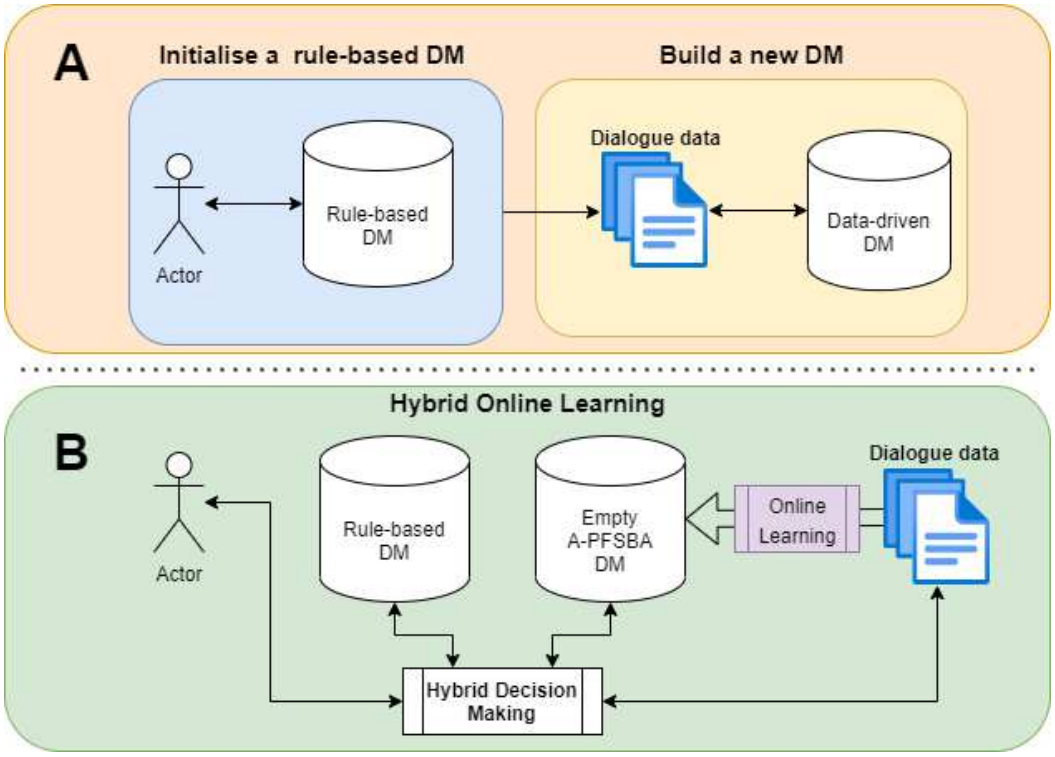


Fig. 6.4.: Classic DM iteration from rule-based DM to a data-driven DM diagram (A) and the proposed Hybrid Online Learning methodology diagram (B)

In this zero-data scenario, a common approach is to initialise a first version of the DM by using expert rules to define the policy. This DM gathers new dialogues through interaction with real users. Then, these dialogue samples are used to build a new data-driven DM. This approach requires the use of two different paradigms (rule-based and data-driven DMs), as depicted in Figure 6.4 Section A.

For these situations, a Hybrid Online Learning method is presented in this section. The HOL method, shown in Figure 6.4 Section B, has three main components:

- **Double DM initialisation**: two separate DM are built, one based on expert rules in order to allow decision-making when data is not available and another one as an empty A-PFSBA model $\hat{M}_\epsilon$ where $Q = \{q_0\}$, where q_0 is the empty state.

- **Hybrid Decision Making (HDM):** a decision making mechanism which hybridizes both the expert-rule and A-PFSBA based DM to generate dialogue samples $z' \in Z'$ when interacting with real users or with a UM.

- **Online Learning**: the OL algorithm is used over the empty A-PFSBA DM $\hat{M}_\epsilon$ in order to populate and augment it using the generated dialogue samples $z' \in Z'$.

Note that this methodology is intended for zero-data scenarios, where a rule-based DM has to be built anyway. HOL proposes a simple methodology to refine the rules and learn a data-driven A-PFSBA DM by using the TC metrics or soft human supervision as QM for OL. To successfully employ the HOL, the alphabets of the A-PFSBA (Δ, Σ, Ω) need to be defined beforehand, i.e. the user actions, the system actions and the attributes. Predefining these alphabets enables to define dialogue management rules that are compatible with the A-PFSBA formulation.

6.3.1 Hybrid Decision Making

The proposed Hybrid Decision Making methodology makes it possible the joint coexistence of a rule-based DM with policy Π_{rules} and an A-PFSBA based DM with policy $\Pi_{A-PFSBA}$. The method is shown in Figure 6.5.

The HDM approach works under the assumption that the dialogue samples used to learn the A-PFSBA model via OL are correct and validated. As a result, if the A-PFSBA can correctly track the current dialogue state $q' \in Q$, the A-PFSBA-based DM is used to continue with the dialogue. Otherwise, the DM based on expert rules is used.

Let Π_{rule} be the handcrafted rule-based policy and $\Pi_{A-PFSBA}$ the exploitation policy employed by the A-PFSBA model. Then, the hybridization technique works as follows:

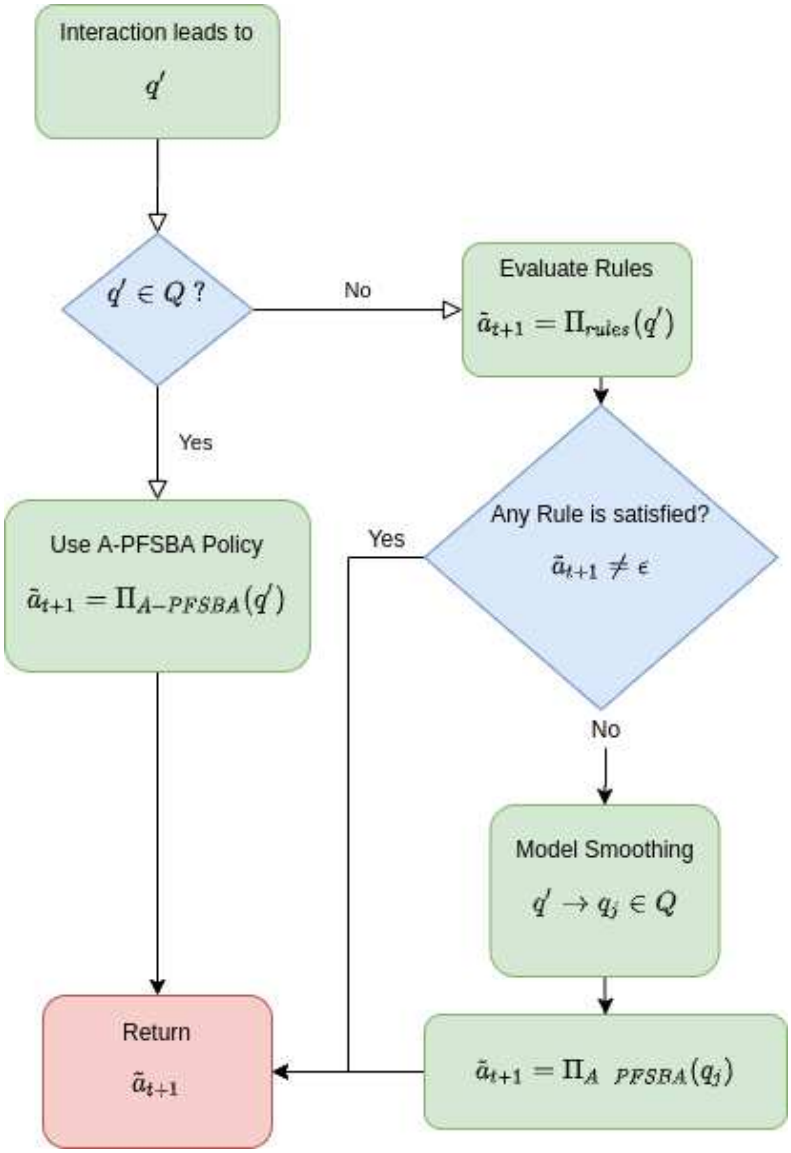


Fig. 6.5.: Flowchart of the Hybrid Decision Making algorithm

1. Dialogue state q' is reached during interaction. Two options arise: the dialogue state q' is either known and $dist(q', q_j) \leq r_\epsilon \; \forall q_j \in Q$, or not.

2. If q' is closer than $r_{epsilon}$ to a known state $q_j \in Q$, this dialogue state is used, and the next action is selected by the A-PFSBA based DM: $\tilde{a}_{t+1} = \Pi_{A-PFSBA}(q_j)$.

3. If $dist(q', q_j) > r_\epsilon \; \forall q_j \in Q$, the rule-based DM is used, so the next action $\tilde{a}_{t+1} = \Pi_{rule}(q')$. If any rule is satisfied and $\tilde{a}_{t+1} \neq \epsilon$, $\tilde{a}_{t+1}$ is returned.

4. If no rule is satisfied and the A-PFSBA DM is empty (i.e. $Q = \{q_0\}$), a predefined fallback action (e.g. "*Sorry, could you rephrase that*") is used as $\tilde{a}_{t+1}$.

5. If the A-PFSBA model is not empty, the model smoothing strategy is triggered in order to route the interaction to a known dialogue state $q_j \in Q$.

6. Finally, the A-PFSBA policy is used over the known dialogue state $q_j \in Q$. Then, $\tilde{a}_{t+1} = \Pi_{A-PFSBA}(q_j)$ is returned as the next system action.

This hybridization mechanism can be used to interact with the users and generate new dialogue samples $z' \in Z'$. Then, OL can be employed to learn and update an A-PFSBA based DM. The only requirement for the rule-based policy Π_{rules} in this context is that it should use the A-PFSBA alphabets Δ, Σ, Ω as input in order to output a valid system language item $\tilde{a}_{t+1} \in \Delta^{\leq m}$.

6.3.2 Rules over the A-PFSBA Model

A rule-based policy Π_{rules} can be defined as a tuple (ER, Sel) where:

- ER is a set of expert rules $(er_1, \cdots, er_{|ER|})$ to be evaluated over a given dialogue state q.

- Sel is a selection mechanism that decides which rules are selected in order to compose the next system action $\tilde{a}_{t+1}$. This selection criterion is employed when several rules may activate at the same time.

Then, an er over the A-PFSBA model can be defined as:

$$\begin{gathered} er : Q \times \Omega \rightarrow (\Delta^{\leq m}, \mathbb{R}^+) \\ er(q_i) \rightarrow \tilde{a}, score \end{gathered} \tag{6.1}$$

Which is a function that receives a dialogue state q_i, composition of the Δ, Σ and Ω alphabets, and returns a system response and a score. Each expert rule er has several conditions $\{c_1, \cdots, c_n\}$ which are evaluated upon single elements of the A-PFSBA alphabets.

For illustrative purposes, let us define the following rule: ***If the system has offered a venue and the user asks about the address, inform about the address***. The conditions of this rule are the following ones:

$$\begin{gathered} c_1 = offer(venue) \in \tilde{a}_{t-1} \text{ //The system has offered a venue to the user} \\ \text{c}_2 = request(address) \in \tilde{d}_t \text{ //User asks for the address} \end{gathered} \tag{6.2}$$

And the response to be given is $\tilde{a}_{t+1} = inform(address)$. In practice, the conditions that are evaluated over the user-alphabet Σ use different thresholds to account for uncertainty, e.g the score of "*request(address)*" is higher than 0.75. Then, let us define the score of an expert rule $er_{score}(q_j)$ as the product of the score of all the conditions $\{c_1, \cdots, c_n\}$ of the rule:

$$er_{score}(q_j) = \prod_{i=1}^{n} c_i(q_j)$$

If deterministic conditions are employed on the expert rules, the score given by an er is either 0 or 1 (all the conditions are satisfied or not).

Let $ER^+(q_j) = \{er_i \in ER : er_{score}(q_j) > 0\}$ be the set of those expert rules which give a positive score over the dialogue state q_j. Then, the expert-rule based policy can be described as:

$$\tilde{a}_{t+1} = \Pi_{rule}(q_j) = Sel(ER^+(q_j)) \tag{6.3}$$

Where the selection method Sel composes the next action from the expert rules that were activated in the dialogue state q_j. A simple selection method is to randomly choose one $er_j \in ER^+(q_j)$ and to use its response as the next system action.

6.3.2.1 Probabilistic Rules

A common and easy way to define expert rules is by using cause-effect relations. Despite their effectiveness in industrial environments, it is not trivial how to handle the uncertainty induced by the Speech To Text (STT) in the Natural Language Understanding (NLU) module. Usually, the user decodings $\tilde{d}_t$ have a degree of uncertainty, i.e. a confidence score between 0 and 1, which requires to set thresholds on the rule conditions.

More generally, considering $P(x|q_j)$ the probability of some item $x \in \{\Sigma, \Delta, \Omega\}$ (i.e. an user act, system response or attribute) in a dialogue state q_j, a common practice when building expert rules is to define a set of thresholds that need to be satisfied $\theta_{lower} < P(x|q_j) < \theta_{upper}$ to consider the condition satisfied. The definition of these boundaries has to be made for each rule, which increases the complexity of the rule-based policy as these boundaries may be hard to define.

In order to handle this uncertainty, the deterministic conditions of the expert rules can be converted into probabilistic ones. This allows the conversion of the following conditions into probabilistic ones by using simple transformations:

- **Exists** x: $P(x \mid q_j)$
- **Not Exists** x: $1 - P(x \mid q_j)$
- x **And** y: $P(x \mid q_j) \cdot P(y \mid q_j)$
- x **Or** y: $\max(P(x \mid q_j), P(y \mid q_j)))$
- x **Greater than** y: $P(x > y \mid q_j)$
- x **Lower than** y: $P(x < y \mid q_j)$

Using this schema, the deterministic conditions can be re-implemented into probabilistic ones, rendering probabilistic rules.

6.4 Continuous Learning Experimentation

As explained previously, different experimentation setups are employed to test different hypotheses. The first scenario corresponds to a Continuous Learning, i.e. when a first DM is deployed with enough data, and, then, the OL method is used with the data samples generated with users/UM to improve the DM. In this setup, the Let's Go corpus is used, where after each UM-generated dialogue the OL is applied to update the DM. The goal is to demonstrate that the OL algorithm is capable of improving an initial DM on a dialogue-by-dialogue basis.

The Quality Metric QM used for the OL in this scenario is the same as the TC metric of the Let's Go corpus depicted at Algorithm 6 of Chapter 3. This metric checks that a suitable query has been performed into the bus schedule database and that the result is informed to the user.

In order to test the performance of the algorithm, 400000 dialogues are generated using the Random Sampling policy for both UM and DM explained in Section 5.2.1.2. This policy randomly samples the next action from the available transitions of the departing dialogue state. Remember that the DM is trained over half of the Let's Go Corpus and the UM using the other half.

Previous results using the different local and path-based policies introduced in Chapter 5, Section 5.2 rendered the results depicted in Table 6.1 in the *Before Online Learning* section. Briefly explained, the employed policies are the following:

- **MP Local**: Local Maximum Probability policy, which employs the transition with highest probability.

- **MPP**: Maximum Probability Path policy, which employs the path that has the highest probability.

- **AP**: Attribute Path policy, which chooses the path with the highest probability and which completes as many dialogue attributes as possible.

- **TCP**: Task Completion Path, which chooses the path with the highest score according to the Task Completion rate, i.e. the path that satisfies most constraints to consider a dialogue successful.

After the continuous learning with the OL algorithm is performed, the *After Online Learning* section of Table 6.1 depicts the results achieved by each policy.

Tab. 6.1.: Results achieved by the different policies over the Let's Go Corpus before and after the Online Learning

DM Policy	Task Completion (%)	Average Dialogue Length
Before Online Learning		
MP Local	60.02 ± 1.36	30.98 ± 0.94
MPP	59.3 ± 0.6	32.2 ± 0.3
AP	59.5 ± 0.6	32.8 ± 0.3
TCP	61.2 ± 0.6	32.5 ± 0.3
After Online Learning		
MP Local	69.4 ± 1.4	31.5 ± 1.0
MPP	73.8 ± 0.5	$\mathbf{30.5} \pm 0.3$
AP	$\mathbf{74.9} \pm 0.5$	$\mathbf{29.9} \pm 0.3$
TCP	$\mathbf{75.0} \pm 0.5$	31.6 ± 0.3

The results clearly show the effectiveness of the OL method for incrementally learn a DM, where an improvement of 9% to 14% over the TC rate is achieved for different policies. Also, in this new scenario, and compared to pre-OL results, the path-based policies are the ones that perform the best, where the AP and TCP policies achieve statistically similar results. This result indicates that prior to adjusting the DM for the users, it is preferred to employ local policies, but, after the parameters of the model are adjusted, it is sensible to exploit this information by means of path-based policies.

To better inspect the behaviour of path-based policies after the OL is performed, Figure 6.6 shows the spline-smoothed TC rate for each policy. To depict the impact of the path depth D, i.e. how many steps into the future are used for the policy estimation and the user-awareness β, a grid search evaluation is performed. Remember

that user-awareness parameter assesses how the user-related transition probabilities are taken into account in the step-value function of Equation 5.5. The higher β is, the more importance is given to the user-estimated transition probabilities, and, if $\beta = 0$, all user-transitions have the same score or importance.

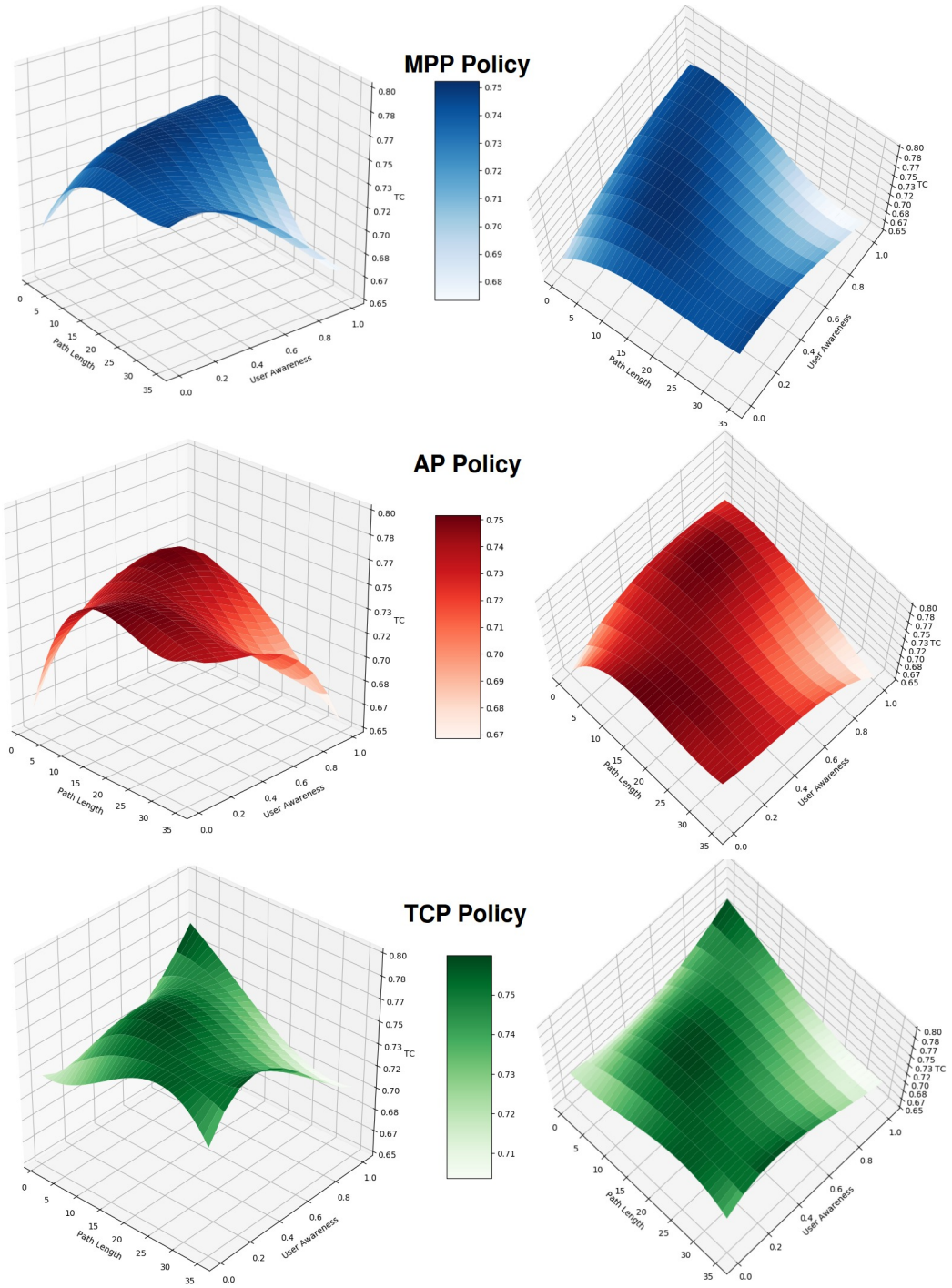


Fig. 6.6.: Spline-smoothed plots of the TC rate of the path-based policies after the OL with different perspectives of the same plot. The left-axis indicates the depth of the path D, the right axis indicates the value of the user-awareness parameter β and the z-axis indicates the TC rate. Left: front view, right: top view

Note that previous results depicted in Figure 5.3 concluded that lower user-awareness and depths were better for policy-making due to uncertain estimations of the transition probabilities. Also, shorter paths indicate that the A-PFSBA model is not correctly estimating the actions of the user. Once the OL is performed, these conclusions drastically change: a good combination of user-awareness ($\beta \sim 0.5$) and path depth ($D \sim 15$) achieve the best results for almost any path-based policy. This indicates that the OL correctly updates the A-PFSBA model $\hat{M}$ creating new dialogue-states and transitions that capture user behavior, improving the estimated transition probabilities and fine-tuning the initial DM to the user.

6.5 Low Data Experimentation

Using the DSTC2 corpus, two low-data experimental setups have been designed to test the potential of the OL algorithm and the HOL methodology in these scenarios. Note that these scenarios are common when facing a first development of a Dialogue System for a new domain. To this end, two scenarios are set up:

- **Data scarcity**: the initial DM is trained using less than 5% of the corpus. This DM is then improved using the OL algorithm which employs batches of 200 dialogues generated with the BAUM2 User Model for the update step. The goal is to prove the adequateness of the combination of the A-PFSBA framework and the OL algorithm for building DMs in low-resource scenarios.

- **Zero-data**: in this setup an A-PFSBA DM is built without initial data. In this situation, a rule-based DM is built as initialisation, and by means of the HOL methodology, an A-PFSBA DM is built using the dialogues generated with the BAUM2. The goal is to demonstrate the potential of the HOL methodology when no data is available.

For both setups, Table 6.2 shows the parameters shared by all the A-PFSBA UMs and DMs. Three Task Completion metrics are employed to evaluate the performance of the DM:

- **Valid Venue Offer**: if the DM offers a venue that satisfies the user's constraints.

- **Request Informed**: if the DM correctly answers to the items requested by the user.

- **Canthelp Inform**: if the DM correctly informs about impossible constraint combinations.

Each TC metric evaluates a specific functionality that the A-PFSBA DM has to satisfy over the DSTC2 corpus. This approach differs from the Let's Go TC, which performs an holistic evaluation of the dialogue.

To evaluate the incremental learning process according to these TCs, two BAUM2 are employed, one trained using the development set and the other using the test set. The UM trained over the development set is used to generate the dialogues from which the A-PFSBA DM will learn using the OL algorithm. Then, this DM will generate dialogues with the BAUM2 trained with the test set to obtain the TC rates.

Tab. 6.2.: Parameters shared by all the models of the DSTC2 low-data experiments

	A-PFSBA DM	**Dev - BAUM2**	**Test - BAUM2**
Delexicalisation method	Value Ranking	Goal-based	Goal-based
Spatial Relation	Euclidean dist.	Euclidean dist.	Euclidean dist.
Dialogue Smoothing - Policy	Nearest State	Nearest State	Nearest State
Chi-square spatial relation	No	No	No
State Pruning Method	None	SVM - 0.2	SVM - 0.2
Data Partition Used	Train	Development	Test
Override threshold	–	0.2	0.2
NLU Error- $\alpha_{corrupt}$	1	1	1
NLU Error n_rounds	10	10	10

6.5.1 Data scarcity scenario

In this second scenario, a data scarcity scenario is simulated. Instead of employing the 1612 dialogues of the DSTC2 training set, one dialogue per goal-type is selected. This is, one dialogue is selected per each different combination of constraint and request slots at the corpus. This renders an initial sampling of 76 dialogues, less than 5% of the corpus. Then, this minimal A-PFSBA model $\hat{M}$ is enhanced with the dialogues generated interacting with BAUM2.

In this setup, a total of 24 dialogue-generation batches $b = 1, \cdots, 24$ are performed using the **A-PFSBA DM** and **Dev-BAUM2**, each batch consisting of 200 dialogues. After a batch is finished, the dialogue samples $z' \in Z_b$ that satisfy all the TC metrics are used to update the A-PFSBA model of the DM via Online Learning. Once the update is completed, the **Test-BAUM2** is used to generate a total of 1000 dialogues which are used to evaluate the updated model. To add some channel noise to the interactions, the NLU error simulation is used with $\alpha_{corrupt}$=1 and $n_{rounds} = 10$. To deal with sparsity added by NLU error with, $r_\epsilon = 0.3$ is set. Note that the first batch $b = 0$ corresponds to the evaluation of the A-PFSBA DM **without** any learning, i.e. the A-PFSBA DM trained with 76 dialogue samples.

6.5.2 Results

The TC metrics evaluated over these 1,000 dialogues are the ones shown in the graphs below. In addition to the metrics, the percentage of the dialogues added at each batch is depicted using the green shaded area. Note that the % of added dialogues might be lower than the TC rate, as it requires to fully satisfy all the TC rates. Also, note that as the TC rates are not binary (i.e. if the user has requested 4 items and the system has correctly answered 3 of them, the Request Informed TC will have a value of 0.75) the percentage of added dialogues might be lower than the overall TC rate.

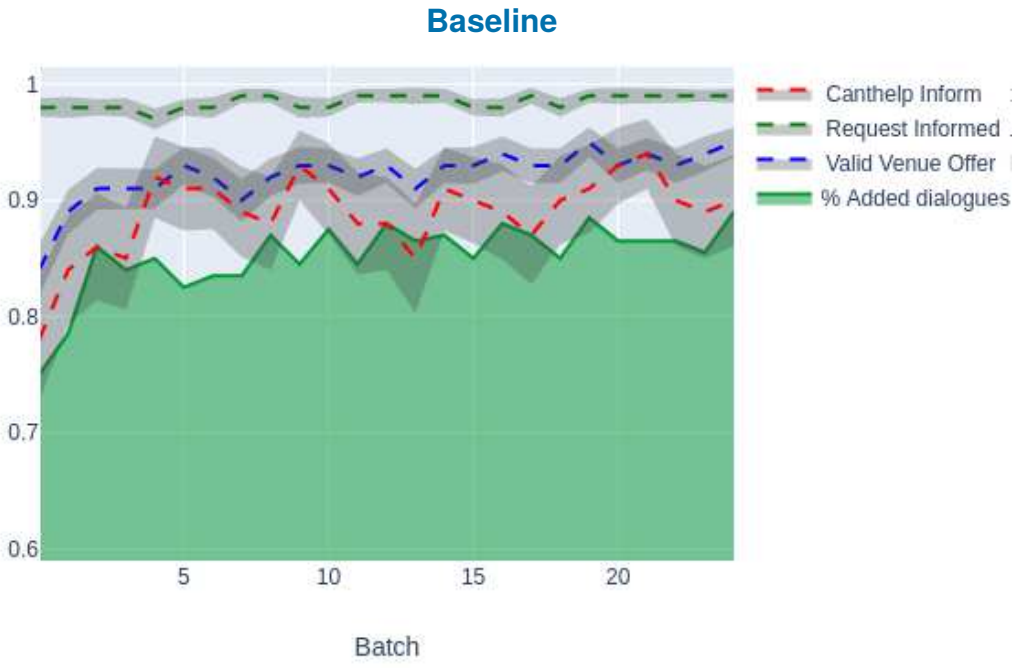


Fig. 6.7.: Task Completion metric evolution during the incremental learning for the *Baseline* configuration

Tab. 6.3.: Initial and ending Task Completion scores learning setup and results for the data scarcity scenario

Configuration	Metric	Initial Score ($b = 0$)	Final Score ($b = 24$)
Baseline	Valid venue offered	0.84±0.023	0.95±0.013
	Canthelp Informed	0.78±0.053	0.9±0.039
	Request Score	0.98±0.008	0.99±0.006

The initial and final results are described in Table 6.3. Results are shown in green if there is a statistically significant improvement, yellow if there is no statistically significant improvement and red if the results worsens. The results achieved indicate

that the OL strategy works fine for data scarcity scenario, even with uncertainty, where the saturation point is reached within just 5 batches. This demonstrates the simplicity and the validity of the OL algorithm for incrementally learning an A-PFSBA DM in low-data scenarios.

6.5.3 Zero data scenario

In this third scenario, the common zero-data setup is simulated over the DSTC2 corpus. To this end, a rule-based DM is built using the expert rule notation of the previous section. These rules can be seen in Appendix B, both the deterministic rules and the probabilistic ones. If no rule is satisfied, the fallback action is used. As an illustrative example, an algorithmic version of the rule "*If the food slot value confidence is below 0.25, request the user for the desired food type*" is shown below:

Listing 1 Rule for Requesting the **food** slot value

```
def request_food(*args, **kwargs):
    attributes = get_dialogue_attributes(kwargs)
    score = attributes.get('food-max-score', 0)
    if 0 <= score < 0.25:
            return True
    return False

HC_POLICY.add_rule(DialogueRule(request_food,
 ↪  'request(food)'))
```

For both rule types, deterministic and probabilistic, the same selection mechanism Sel is used.

Algorithm 6 Rule selection mechanism

$K_{rules} \leftarrow$ random_choice($[1, 2, 3, 4]$) ▷ Number of of rules to sample
$ER^+(q_j)$ ▷ Rules activated over the current dialogue-state q_j
Scores = $[er_{score}(q_j) \forall\ er \in ER^+]$
Normalize(Scores) so $|Scores| = 1$
$S_{ER^+(q_j)} = sample(ER^+(q_j), K_{rules})$ ▷ Selected rules
$a_{t+1} = er_{\tilde{a}}\ \forall\ er \in S_{ER^+(q_j)}$ ▷ Composition of rule-activated system actions
return a_{t+1}

As explained in the Pseudo-algorithm 6 a K_rules parameter is randomly chosen from 1 to 4. Then, using the retrieved scores of each expert rule er activated at the

dialogue state q_j, the scores are normalized to a multivariate distribution. Then, using this distribution, a total of K_rules are sampled. The system's responses of the sampled expert rules are used to produce the next system action $\tilde{a}_{t+1}$. If $ER^+(q_j) = \emptyset$, i.e. no rules are activated, a fallback action is defined, which for the DSTC2 domain is the *confirm-domain* action i.e. "*Are you looking for a restaurant?*".

6.5.3.1 Results

The HOL method is tested with both deterministic and probabilistic rules. Similar to the data scarcity scenario, for the zero-data scenario, a total of 24 dialogue-generation batches $b = 1, \cdots, 24$ are performed using the **A-PFSBA DM** and **Dev-BAUM2**, each batch consisting of 200 dialogues. After a batch is finished, the dialogue samples $z' \in Z_b$ that satisfy all the TC metrics are used to update the A-PFSBA model of the DM via Online Learning. Once the update is done, the test **Test-BAUM2** is used to generate a total of 1,000 dialogues, which are used to evaluate the updated model. In order to add some channel-noise to the interactions, the NLU error simulation is used with $\alpha_{corrupt}$=1 and $n_{rounds} = 10$. To deal with sparsity added by NLU error, the $r_\epsilon = 0.3$ is set. Note that the first batch $b = 0$ corresponds to the evaluation of the A-PFSBA DM **without** any learning. In this zero-data scenario this means that the rule-based DM is being evaluated, as the A-PFSBA DM consists only of the empty state q_ϵ. Note that in this HOL setup, the more dialogue samples $z' \in Z_b$ are learned with the OL, the more relevant the A-PFSBA based DM becomes, relegating the rule-based DM to the background.

The TC metrics evaluated over these 1,000 dialogues are the ones shown in the graphs below. Apart from the metrics, the percentage of the dialogues added in each batch is depicted using the green shaded area. Note that the % of added dialogues might be lower than the TC rate, as it requires to fully satisfy all the TC rates. Also note that as the TC rates are not binary (i.e. if the user has requested 4 items and the system has correctly answered 3 of them, the Request Informed TC will have a value of 0.75) the percentage of added dialogues might be lower than the overall TC rate.

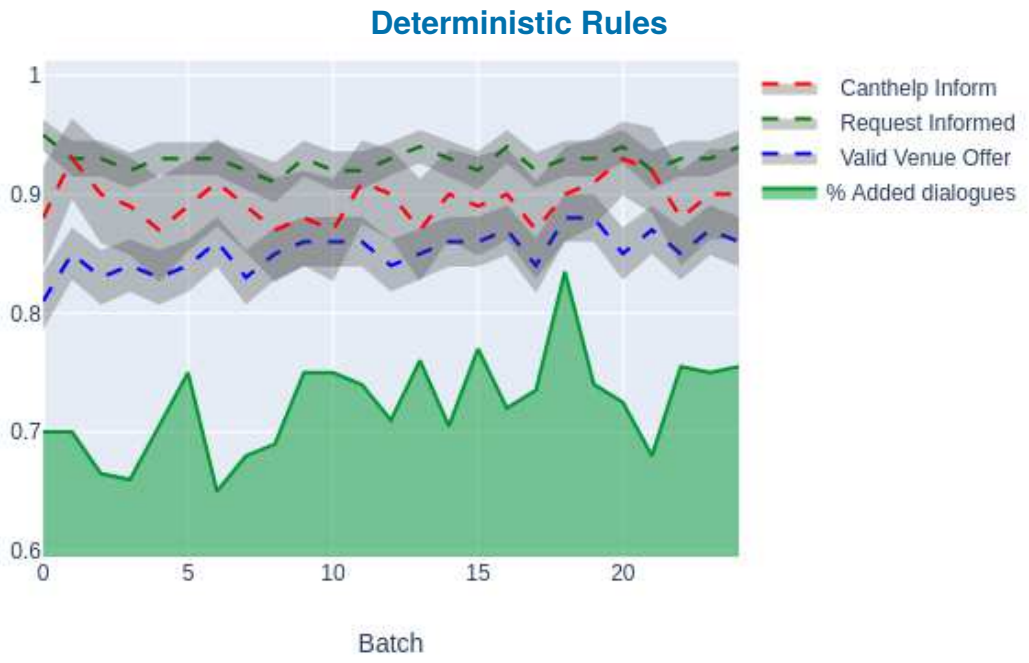


Fig. 6.8.: Task Completion metric evolution during the incremental learning for the *Deterministic Rules* configuration. The x axis denotes the Batch $b \in [0 \cdots, 24]$ and the y axis the achieved score for different TC metrics.

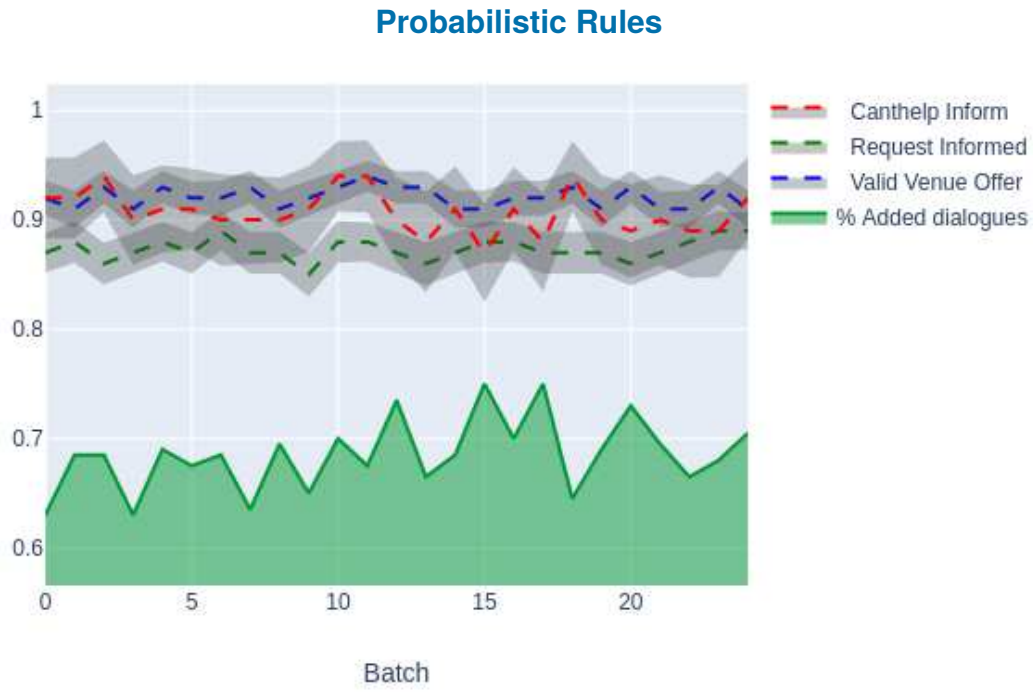


Fig. 6.9.: Task Completion metric evolution during the incremental learning for the *Probabilistic Rules* configuration. The x axis denotes the Batch $b \in [0 \cdots, 24]$ and the y axis the achieved score for different TC metrics.

Tab. 6.4.: Summary of the incremental learning results for the Zero Data Scenario

Initial DM	Metric	Initial Score ($b = 0$)	Final Score ($b = 24$)
Det. Rules	Valid venue offered	0.81±0.024	0.86±0.021
	Canthelp Informed	0.88±0.043	0.9±0.034
	Request Score	0.95±0.013	0.94±0.014
Prob. Rules	Valid venue offered	0.92±0.016	0.91±0.017
	Canthelp Informed	0.92±0.037	0.92±0.037
	Request Score	0.87±0.013	0.89±0.017

Table 6.4 compares the initial batch scores and final batch scores when performing the HOL experiment. The deterministic rules demonstrated to have a higher learning potential than the probabilistic rules. However, the probabilistic ones yield better results when offering a suitable restaurant for the user, which is the main goal of the system, as the **valid venue offer** TC metric demonstrates. As the probabilistic rules produce a higher variability in the system responses, they can explore multiple strategies which are more likely to satisfy an automatic TC metric. Nevertheless, this same variability may make it harder to capture a consistent pattern to learn DM strategies.

From and industrial point of view, deterministic rules, despite their lower TC rate, are easier to interpret and explain. Therefore, they improve the auditability of the DM and produce more consistent interactions with the user. Nevertheless, the achieved results indicate the potential of the A-PFSBA structure to bridge rule-based and data-driven DMs in a smooth way.

CAPITULO 7 SUJETO A CONFIDENCIALIDAD POR EL AUTOR

8 Conclusions and Future Work

During this thesis the Attributed Probabilistic Finite State Bi-Automata (A-PFSBA) framework has been used for Dialogue Management and User Modelling in different corpora and applications. On the whole, the main contributions can be summarized in these points:

- Two use cases have been used to carry out the experimental validation of the proposed contributions: Let's Go and Dialogue State Tracking Challenge 2 (DSTC2) Corpus. The first one provides a challenging corpus collected from real users. The second one provides a statistical Natural Language Understanding (NLU) which allows to test the capability of the A-PFSBA framework to handle uncertainty and provides user-related goals for better user modelling.

- Several new A-PFSBA model smoothing techniques have been proposed in order to improve generalisation for new interactions and unseen situations.

- The A-PFSBA theoretical framework has been extended by defining the policy-making notation over this framework. Different policies following this notation have been implemented.

- In order to enable Incremental Learning of the initial Dialogue Manager (DM), an Online Learning (OL) algorithm and a DM hybridisation mechanism have been presented.

- Three different real-word applications which use the A-PFSBA structure for DM have been presented. These applications cover a wide range of domains, such as industrial maintenance, gerontological data registration and eGoverment.

All contributions have been experimentally validated over the Let's Go corpus and/or the DSTC2 corpus. Segmented by main contributions, the following bullet points summarize the conclusions:

- **Model Smoothing:** The capability of adapting and generalising to unseen situations at decoding time is a key feature for any DM. As demonstrated in this

thesis, the A-PFSBA framework can employ several mechanisms to achieve this generalisation, such as using simple spatial relations, K-Nearest dialogue state voting mechanisms, Machine Learning algorithms trained over the A-PFSBA structure for state pruning and crafting task-dependant distance or similarities. The flexibility and variety of mechanisms to achieve a generalisation to unseen dialogues gives the A-PFSBA DMs the potentiality of successfully covering a wide range of tasks and scenarios as showed in the experiments carried out. As a demonstration of the success when modifying the model smoothing strategy, it must be remembered that when using the state pruning with SVMs in the Bi-Automata User Model (BAUM), the A-PFSBA based model outperformed the Deep Learning (DL) based UM.

- **Exploitation Policies:** One of the main benefits of the A-PFSBA framework is the separation between structural learning and its exploitation. This separation grants freedom to adjust the decision-making policy to the current application/task of the DM, in which the policy can be a rule-based one, a maximum probability one or any other complex policy which employs external information such as the Task Completion (TC). During this dissertation, the proposed theoretical framework has been tested by implementing multiple exploitation policies with different degrees of complexity.

- **Incremental Learning:** As the A-PFSBA is a generative structure, it allows to incrementally learn new states and transitions by using unseen interactions. The proposed Online Learning algorithm proved to be a simple and effective method to improve an initial A-PFSBA DM in terms of TC. The OL algorithm, combined with the presented hybridization methodology, which initializes jointly a rule-based and A-PFSBA based DMs, can be potentially used to incrementally learn a data-driven A-PFSBA DM even in zero-data domains.

- **Industrial Applications:** The experimental validations and the developed applications, which were tested by real users and were backed up by different projects and companies, demonstrate that the A-PFSBA framework is a suitable methodology for building DMs in industrial applications and that it has the potential of bridging rule-based and data-driven systems.

On the whole, the contributions presented in this thesis demonstrate the potential of the A-PFSBA framework for Dialogue Management at Dialogue Systems. This statistical framework can handle the intent/entity/value structure of the NLU even in presence of uncertainty. The A-PFSBA structure is suitable to model the task to perform by the system and extract relevant patterns from the data, using a computationally-light model.

Although the demonstrated potentiality of the A-PFSBA framework to model DMs in different environments through this dissertation is proven, some limitations still remain, such as the need of crafting the Δ, Σ and Ω alphabets and the delexicalisation mechanisms to avoid excessive data-sparsity. As this process is highly-domain dependant, the transferability of the model to other domains remains a challenge to be solved.

Under this scenario, multiple research lines are open to further improve and validate this framework.

As future work, and for better generalisation, methods and heuristics to include path-information in the model smoothing strategy will be further researched.

As the A-PFSBA structure can be combined with DL methods, the use of pre-trained language models such as BERT/GPT and their variants to infer the user decoding alphabet Σ will be researched. In addition, the composition of the A-PFSBA framework with other automata employed for grammar inference would also be feasible due to the generative nature of the framework.

In terms of DM policy, the harmonisation of the A-PFSBA policies with other policy-inference methodologies such as the Reinforcement Learning will be researched. In addition, the multi-modal information handling is also of special relevance due to the increasing interest in these interactive systems.

Also, in order to handle multi-task scenarios, the composition of several A-PFSBA DMs to handle multi-task systems needs to be investigated, in which the automata and transducer literature already provides theoretical mechanisms for this composition.

Finally, and to fully explore the potential of the A-PFSBA of bridging rule-based and data-driven DM, further research will be performed on building robust A-PFSBA DMs incrementally in data scarcity and/or zero-data scenarios.

9 Related Publications

The development and writing of this thesis has resulted in several peer-reviewed articles.

Online learning of attributed bi-automata for dialogue management in spoken dialogue systems

Serras, M., Torres, M. I., Del Pozo, A. (2017, June). Online learning of attributed bi-automata for dialogue management in spoken dialogue systems. In Iberian Conference on Pattern Recognition and Image Analysis (pp. 22-31). Springer, Cham.

- **DOI:** `https://doi.org/10.1007/978-3-319-58838-4_3`

Abstract: Online learning of dialogue managers is a desirable but often costly property to obtain. Probabilistic Finite State Bi-Automata (PFSBA) have shown to provide a flexible and adaptive framework to achieve this goal. In this paper, an Attributed PFSBA (A-PSFBA) is implemented and experimentally compared with previous non-attributed PFSBA proposals. Then, a simple yet effective online learning algorithm that adapts the probabilistic structure of the Bi-Automata on the run is presented and evaluated. To this end, the User Model is also represented by an A-PFSBA and the impact of different user behaviors is tested. The proposed approaches are evaluated on the Let's Go corpus, showing significant improvements on the dialogue success rates reported in previous works.

Regularized Neural User Model for Goal-oriented Spoken Dialogue Systems

Serras, M., Torres, M. I., Del Pozo, A. (2019). Regularized neural user model for goal-oriented spoken dialogue systems. In Advanced Social Interaction with Agents (pp. 235-245). Springer, Cham.

- **DOI:** `https://doi.org/10.1007/978-3-319-92108-2_24`

Abstract: User simulation is widely used to generate artificial dialogues in order to train statistical spoken dialogue systems and perform evaluations. This paper presents a neural network approach for user modeling that exploits an encoder-decoder bidirectional

architecture with a regularization layer for each dialogue act. In order to minimize the impact of data sparsity, the dialogue act space is compressed according to the user goal. Experiments on the Dialogue State Tracking Challenge 2 (DSTC2) dataset provide significant results at dialogue act and slot level predictions, outperforming previous neural user modeling approaches in terms of F1 score.

User-aware dialogue management policies over attributed bi-automata

Serras, M., Torres, M. I., Del Pozo, A. (2019). User-aware dialogue management policies over attributed bi-automata. Pattern Analysis and Applications, 22(4), 1319-1330.

- **DOI:** `https://doi.org/10.1007/s10044-018-0743-y`

Abstract: Designing dialogue policies that take user behavior into account is complicated due to user variability and behavioral uncertainty. Attributed probabilistic finite-state bi-automata (A-PFSBA) have proven to be a promising framework to develop dialogue managers that capture the users' actions in its structure and adapt to them online, yet developing policies robust to high user uncertainty is still challenging. In this paper, the theoretical A-PFSBA dialogue management framework is augmented by formally defining the notation of exploitation policies over its structure. Under such definition, multiple path-based policies are implemented, those that take into account external information and those which do not. These policies are evaluated on the Let's Go corpus, before and after an online learning process whose goal is to update the initial model through the interaction with end users. In these experiments the impact of user uncertainty and the model structural learning is thoroughly analyzed.

Goal-conditioned User Modeling for Dialogue Systems using Stochastic Bi-Automata

Serras, M., Torres, M. I., del Pozo, A. (2019, February). Goal-conditioned User Modeling for Dialogue Systems using Stochastic Bi-Automata. In ICPRAM (pp. 128-134).

- **DOI:** `https://doi.org/10.5220/0007359401280134`

Abstract: User Models (UM) are commonly employed to train and evaluate dialogue systems as they generate dialogue samples that simulate end-user behavior. This paper presents a stochastic approach for user modeling based in Attributed Probabilistic Finite State Bi-Automata (A-PFSBA). This framework allows the user model to be conditioned by the dialogue goal in task-oriented dialogue scenarios. In addition, the work proposes two novel smoothing policies that employ the K-nearest A-PFSBA states to infer the next UM action in unseen interactions. Experiments on the Dialogue State Tracking Challenge 2 (DSTC2) corpus provide results similar to the ones obtained through deep learning based

user modeling approaches in terms of F1 measure. However the proposed Bi-Automata User Model (BAUM) requires less resources both of memory and computing time.

Improving Dialogue Smoothing with A-priori State Pruning

Serras, M., Torres, M. I., del Pozo, A. (2020). Improving Dialogue Smoothing with A-priori State Pruning. In ICPRAM (pp. 607-614)

- **DOI:** `https://doi.org/10.5220/0009184206070614`

Abstract: When Dialogue Systems (DS) face real usage, a challenge to solve is managing unforeseen situations without breaking the coherence of the dialogue. One way to achieve this is by redirecting the interaction to known dialogue states in a transparent way. This work proposes a simple a-priori pruning method to rule out invalid candidates when searching for similar dialogue states in unexpected scenarios. The proposed method is evaluated on a User Model (UM) based on Attributed Probabilistic Finite State Bi-Automata (A-PFSBA), trained on the Dialogue State Tracking Challenge 2 (DSTC2) corpus. Results show that the proposed technique improves response times and achieves higher F1 scores than previous A-PFSBA implementations and deep learning models.

Dialogue enhanced extended reality: Interactive system for the operator 4.0

Serras, M., García-Sardiña, L., Simões, B., Álvarez, H., Arambarri, J. (2020). Dialogue enhanced extended reality: Interactive system for the operator 4.0. Applied Sciences, 10(11), 3960.

- **DOI:** `https://doi.org/10.3390/app10113960`

Abstract: The nature of industrial manufacturing processes and the continuous need to adapt production systems to new demands require tools to support workers during transitions to new processes. At the early stage of transitions, human error rate is often high and the impact in quality and production loss can be significant. Over the past years, eXtended Reality (XR) technologies (such as virtual, augmented, immersive, and mixed reality) have become a popular approach to enhance operators' capabilities in the Industry 4.0 paradigm. The purpose of this research is to explore the usability of dialogue-based XR enhancement to ease the cognitive burden associated with manufacturing tasks, through the augmentation of linked multi-modal information available to support operators. The proposed Interactive XR architecture, using the Spoken Dialogue Systems' modular and user-centred architecture as a basis, was tested in two use case scenarios: the maintenance of a robotic gripper and as a shop-floor assistant for electric panel assembly. In both cases, we have confirmed a high user acceptance rate with an efficient knowledge communication and distribution even for operators without prior experience or with cognitive impairments, therefore demonstrating

the suitability of the solution for assisting human workers in industrial manufacturing processes. The results endorse an initial validation of the Interactive XR architecture to achieve a multi-device and user-friendly experience to solve industrial processes, which is flexible enough to encompass multiple tasks

AREVA: Augmented Reality Voice Assistant for Industrial Maintenance

Serras, M., García-Sardiña, L., Sim, B., Ávarez, H., Arambarri, J. (2020). AREVA: Augmented Reality Voice Assistant for Industrial Maintenance. Procesamiento del Lenguaje Natural, 65, 135-138.

- **DOI:** `https://doi.org/10.26342/2020-65-21`

Abstract: Within the context of Industry 4.0, AREVA is presented: a Voice Assistant with Augmented Reality visualisations for the support and guidance of operators when carrying out tasks and processes in industrial environments. With the aim of validating its use for the training of new operators, first evaluations were performed by a group of non-expert users who were asked to carry out a maintenance task on a Universal Robot.

RESIVOZ: Dialogue System for Voice-based Information Registration in Eldercare

García-Sardiña, L., Serras, M., del Pozo, A., Fernández-Bhogal, M. D. (2020). RESIVOZ: Dialogue System for Voice-based Information Registration in Eldercare. Procesamiento del Lenguaje Natural, 65, 123-126.

- **DOI:** `https://doi.org/10.26342/2020-65-18`

Abstract: RESIVOZ is a spoken dialogue system aimed at helping geriatric nurses easily register resident caring information. Compared to the traditional use of computers installed at specific control points for information recording, RESIVOZ's hands-free and mobile nature allows nurses to enter their activities in a natural way, when and where needed. Besides the core spoken dialogue component, the presented prototype system also includes an administration panel and a mobile phone App designed to visualise and edit resident caring information.

DSTC2 Attribute Iteration Rules

A

This appendix describes the attribute iteration rules for the Bi-Automata User Model described at Section 3.2.2.2. As previously said, these rules follow an *if-this-then-that* schema.

Note that the activation of an attribute means to set its value to 1 and the deactivation to set its value to 0.

A.1 BAUM Attribute Iteration Rules

This section describes the attribute iteration rules used for the first BAUM model, i.e. the one which only uses the information explicitly labelled in the JSON of the DSTC2 samples.

Write Goal Constraints Attributes

This rule is used at the beginning of the dialogue, before the first turn, so the user goal related constraints are taken into account throughout the dialogue.

1. **IF:** <slot-name> is in the given dialogue goal constraint set. **THEN:**
    - **ACTIVATE** the attribute "*constraint-<slot-name>*"

As example, if the food=chinese is given as a constraint to the Mechanikal Turker, then the attribute "*constraint-food*" receives the value 1.

Write Goal Request Attributes

This rule is used at the beginning of the dialogue, so the user goal related requests are taken into account throughout the dialogue.

1. **IF:** <slot-name> is in the given goal's requests. **THEN:**
    - **ACTIVATE** the attribute "*request-<slot-name>*"

Write Received Venue Information Rule

This attribute iteration rule is evaluated every time that the system gives a response. Writes the information that the user has received from a venue.

1. **IF:** system has offered a venue **AND** informed <slot-name> at the same time. **THEN:**

    - **ACTIVATE** the attribute "*received-venue-<slot-name>*"

For example, if the system offers a venue and informs about the venue's address, the "*received-venue-address*" attribute is set to 1.

System Understood Constraints Goal Rule

This attribute iteration rule is evaluated every time that the system gives a response. It is activated when the system demonstrates that it understood the user-given goal information.

1. **IF:** system <intent> in *['expl-conf', 'impl-conf', 'inform', 'canthelp']* **AND NOT** informed <slot-name> in *['count', 'addr','phone', 'postcode', 'signature']* **AND** <slot-value> is *part of the given goal constraints*. **THEN:**

    - **ACTIVATE** the attribute "*system-understood-<slot-name>-goal*"

    - **DEACTIVATE** the attribute "*system-understood-<slot-name>-other*"

System Understood Constraints Other Rule

This attribute iteration rule is evaluated every time that the system gives a response. It is activated when the system demonstrates that it misunderstood the user-given goal information.

1. **IF:** system <intent> in *['expl-conf', 'impl-conf', 'inform', 'canthelp']* **AND NOT** informed <slot-name> in *['count', 'addr','phone', 'postcode', 'signature']* **AND** <slot-value> is *not in the given goal constraints*. **THEN:**

    - **DEACTIVATE** the attribute "*system-understood-<slot-name>-goal*"

    - **ACTIVATE** the attribute "*system-understood-<slot-name>-other*"

System Understood Constraints Don't care Rule

This attribute iteration rule is evaluated every time that the system gives a response. It is activated when the system demonstrates that it understood the user doesn't care about a particular slot's value.

1. **IF:** system <intent> in *['expl-conf', 'impl-conf', 'inform', 'canthelp']* **AND NOT** informed <slot-name> in *['count', 'addr','phone', 'postcode', 'signature']* **AND** <slot-value> is "*dontcare*". **THEN:**

- **ACTIVATE** the attribute *system-understood-<slot-name>-goal*
- **ACTIVATE** the attribute *system-understood-<slot-name>-other*

User Informed Constraints Goal Rule

This attribute iteration rule is evaluated every time that the user gives a response. It is activated when the user informs about a particular slot's value to the system that matches the given goal constraint.

1. **IF:** user <intent> in *['confirm', 'inform']* **AND** <slot-value> is "*goal*". **THEN:**
   - **ACTIVATE** the attribute *user-informed-<slot-name>-goal*
   - **DEACTIVATE** the attribute *system-understood-<slot-name>-other*

User Informed Constraints Other Rule

This attribute iteration rule is evaluated every time that the user gives a response. It is activated when the user informs about a particular slot's value to the system that is different than the given goal constraint.

1. **IF:** user <intent> in *['confirm', 'inform']* **AND** <slot-value> is "*other*". **THEN:**
   - **ACTIVATE** the attribute *user-informed-<slot-name>-other*
   - **DEACTIVATE** the attribute *system-understood-<slot-name>-goal*

User Informed Constraints Don't care Rule

This attribute iteration rule is evaluated every time that the user gives a response. It is activated when the user informs that he/she doesn't care about a particular slot's value to the system.

1. **IF:** user <intent> in *['confirm', 'inform']* **AND** <slot-value> is "*dontcare*". **THEN:**
   - **ACTIVATE** the attribute *user-informed-<slot-name>-other*
   - **ACTIVATE** the attribute *system-understood-<slot-name>-goal*

The *dontcare* slot-value is treated in a special way, so both goal and non-goal attribute values are activated.

Restaurant Change Rule

This attribute iteration rule is evaluated every time that the system gives a response. It is activated when the restaurant from the current response differs from the previous one.

1. **IF:** system <intent> is *'offer'* **AND** <restaurant-name> is different than <previous-restaurant-name>. **THEN:**

   - **ACTIVATE** the attribute *offered-new-venue*

   ELSE:

   - **DEACTIVATE** the attribute *offered-new-venue*

User Requested Alternatives Rule

This attribute iteration rule is evaluated every time that the user gives a response. It is activated when the user requested a different restaurant.

1. **IF:** user <intent> is *'reqalts'*. **THEN:**

   - **DEACTIVATE** the attribute *offered-new-venue*

A.1.1 BAUM2 Attribute Iteration Rules

This section briefly describes the differences in the attribute iteration rules used for the BAUM2 model, i.e. the one which infers the user goal using the text given to the participants/mechanical turkers. This goal also encodes information to handle unfeasible constraints and the requesting of alternative venues.

The attribute iteration rules' logic is practically the same, but they now include the *secondary-goal'* concept in their *'if-this-then-that'* schema to take into account two different constraint sets given to the user in one dialogue. In addition, now the "*request-alternatives*" attribute can be activated, when the Mechanical Turker is asked explicitly to ask for an alternative venue rather than the first one offered by the system.

As an example, the goal-constraint writing rule is modified in this way:

Write Goal Constraints Attributes - BAUM2

1. **IF:** <slot-name> is in the given dialogue goal constraint set **as first option**. **THEN:**

   - **ACTIVATE** the attribute *constraint-<slot-name>*

2. **IF:** <slot-name> is in the given dialogue goal constraint set **as second option**. **THEN:**

- **ACTIVATE** the attribute *constraint-secondary-<slot-name>*

The same schema is followed in all the other rules, for example, the attribute iteration rule which takes into account if the user has given information about some slot is also transformed:

User Informed Constraints Goal Rule - BAUM2

1. **IF:** user <intent> in *['confirm', 'inform']* **AND** <slot-value> is "*goal-primary*". **THEN:**
   - **ACTIVATE** the attribute *user-informed-<slot-name>-goal*
   - **DEACTIVATE** the attribute *system-understood-<slot-name>-other*
2. **IF:** user <intent> in *['confirm', 'inform']* **AND** <slot-value> is "*goal-secondary*". **THEN:**
   - **ACTIVATE** the attribute *user-informed-<slot-name>-goal-secondary*
   - **DEACTIVATE** the attribute *system-understood-<slot-name>-other*

Note that now, the "other" slot value refers to any slot value that is different from the first or second constraint set for that slot.

B

Dialogue Management Rules for the DSTC2

In this section the logic of the deterministic and probabilistic rules used at Chapter 6 are described in terms of pseudo-algorithms. Note that the code has been polished for illustrative purposes and there might be small divergences to the final implementation.

If two rules with overlapping system responses are activated, both system actions are combined, without repetitions. As an example, if the rule with the outcome:

"*offer(venue=<best-venue>)&inform(phone=<venue-phone)*"

and the one with:

"*offer(venue=<best-venue>)&inform(address=<venue-address)*"

are activated, the union of both actions is used:

"*offer(venue=<best-venue>)&inform(address=<venue-address>)&inform(phone=<venue-phone>)*".

The delexicalised slot values, such as <best-venue>, <venue-address> and <venue-phone> are retrieved from the mapping blackboard detailed in Chapter 3 at Section 3.8.

As notation, during this Appendix the HC_POLICY refers to the deterministic hand-crafted policy and P_HC_POLICY to the probabilistic hand-crafted policy.

B.1 Helper Functions

Different helper functions have been defined for both probabilistic and deterministic rule policy . The one below calculates the maximum score achievable by a venue according to the user-informed constraints. I.e. if the constraints of food and area are known each one with 0.5 probability, the maximum score achievable by a venue is $\frac{0.5+0.5}{2} = 0.5$

Listing 2 Helper Functions that extracts the maximum score that a venue can get

```
def get_venue_score(attributes, blackboard):
    """Get the current venue best score"""
    food_score = f_s = attributes.get('food-max-score', 0)
    area_score = a_s = attributes.get('area-max-score', 0)
    price_score = p_s = attributes.get('pricerange-max-score',
    ↪ 0)
    # Logic behind: if the venue score is low and the slot
    ↪ scores are high, we have an issue.
    max_list = []
    if blackboard.get('<best-area-0>', 'dontcare') !=
    ↪ 'dontcare':
        if a_s != 0:
            max_list.append(a_s)
    if blackboard.get('<best-food-0>', 'dontcare') !=
    ↪ 'dontcare':
        if f_s != 0:
            max_list.append(f_s)
    if blackboard.get('<best-pricerange-0>', 'dontcare') !=
    ↪ 'dontcare':
        if p_s != 0:
            max_list.append(p_s)
    total_score = sum(max_list)
    if total_score == 0:
        return 0
    max_possible_score = total_score/len(max_list)
    return max_possible_score
```

The next helper function retrives the user action $\tilde{d}$, system action $\tilde{a}$ and the attributes ω from the current dialogue state q.

Listing 3 Helper function that extracts the items of the dialogue state from the input information **kwargs

```
def get_dialogue_state(kwargs):
    user = kwargs['user-dict']
    system = kwargs['system-dict']
    attributes = kwargs['attribute-dict']
    return user, system, attributes
```

The next helper function calculates the score relative to offer the current best venue. To that end, the maximum score and the difference with the next restaurant cluster are multiplied together. It supposes that the higher score the best venue has, and the higher is the difference with respect to the other venue types (venues with other characteristics), the higher is the chance to do a proper offering.

Listing 4 Helper function that calculates the score of offering some venue to the user

```
def _offer_score(attributes):
    venue_max_score = attributes.get('venue-max-score', 0)
    cluster_difference = attributes.get('score-difference', 0)
    # The score difference between the first cluster of venues
    ↪  an the second one
    return venue_max_score*venue_max_score
```

B.2 Deterministic Rules

In this section the pseudo-code of the implemented deterministic rules is described. Note that these rules are deterministic in terms that the outcome of their score can only be 1 (True) or 0 (False).

The first rule is used to greet the user, it simply checks that the current state is empty.

Listing 5 Rule for Giving the Welcome Message

```
def welcomemsg(*args, **kwargs):
    # First action
    u, s, a = get_dialogue_state(kwargs)
    if not (u or s or a):
        return True
    return False

HC_POLICY.add_rule(DialogueRule(welcomemsg, 'welcomemsg'))
```

The second set of rules determines when to request the user for particular slots:

Listing 6 Rule for Requesting the **food** slot value

```
def request_food(*args, **kwargs):
    u, s, a = get_dialogue_state(kwargs)
    if _offer_score(a) > 0.5:
        return False
    score = a.get('food-max-score', 0)
    if 0 <= score < 0.25:
            return True
    return False

HC_POLICY.add_rule(DialogueRule(request_food,
 ↪  'request(food)'))
```

Note that this same rule schema is repeated with the slots **area** and **pricerange**

The request/information retrieval logic is conditioned according to the confidence of the slot. If the confidence is low, an explicit confirmation is asked:

Listing 7 Rule for Explicitly Confirming the **food** slot value

```
def explicit_conf_food(*args, **kwargs):
    u, s, a = get_dialogue_state(kwargs)
    if _offer_score(a) > 0.5:
        return False
    score = a.get('food-max-score', 0)
    if 0.25 <= score < 0.5:
        return True
    return False


HC_POLICY.add_rule(DialogueRule(explicit_conf_food,
 ↪  'expl-conf(food=<best-food-0>)'))
```

Note that this same rule schema is repeated with the slots **area** and **pricerange**

If the confidence is medium, an implicit confirmation is made.

Listing 8 Rule for Implicitly Confirming the **food** slot value

```
def implicit_conf_food(*args, **kwargs):
    u, s, a = get_dialogue_state(kwargs)
    if _offer_score(a) > 0.5:
        return False
    score = a.get('food-max-score', 0)
    if 0.5 <= score < 0.75:
        return True
    return False


HC_POLICY.add_rule(DialogueRule(implicit_conf_food,
 ↪  'impl-conf(food=<best-food-0>)'))
```

Note that this same rule schema is repeated with the slots **area** and **pricerange**

The next rule handles the offering of a suitable venue when it gets has score to do so. In addition, three additional rules are added that are used to jointly offer the venue and inform about the registered constraint values of the venue.

Listing 9 Rule for offering a suitable venue to the user

```
def offer(*args, **kwargs):
    u, s, a = get_dialogue_state(kwargs)
    if _offer_score(a) > 0.75:
        return True
    return False


HC_POLICY.add_rule(DialogueRule(offer,
 ↪  'offer(name=<best-venue>)'))
```

Listing 10 Rule for offering a suitable venue and informing its food value to the user

```
def inform_venue_food(*args, **kwargs):
    u, s, a = get_dialogue_state(kwargs)
    if _offer_score(a) > 0.75 and a.get('food-max-score', 0)
    ↪  >= 0.75:
        return True
    return False


HC_POLICY.add_rule(DialogueRule(inform_food_request,
↪  ['offer(name=<best-venue>)',
↪  'inform(food=<venue-food>)']))
```

Note that this same rule schema is repeated with the slots **area** and **pricerange**.

The next set of rules handles the requests made by the user once a venue has enough score to get offered.

Listing 11 Rule for responding to the requests made by the user for the slot **address**

```
def inform_address_request(*args, **kwargs):
    u, s, a = get_dialogue_state(kwargs)
    score = u.get('request(addr')), 0)
    if _offer_score(a) < 0.5:
        return False
    if score > 0.5:
        return True
    return False


HC_POLICY.add_rule(DialogueRule(inform_address_request,
↪  ['offer(name=<best-venue>)',
↪  'inform(addr=<venue-addr>)']))
```

Note that this same rule schema is repeated with the slots **area**, **food**, **pricerange**, **phone**, **postcode** and **signature**.

Finally, the next rules handle the scenario where some constraint cannot be met with the venues of the current database.

Listing 12 Rule for informing the user that the food constraint given cannot be satisfied

```
def canthelp_food(*args, **kwargs):
    # See attributes
    u, s, a = get_dialogue_state(kwargs)
    # The blackboard stores the specific values of the slots
    blackboard_dict = kwargs.get('blackboard', {})

    if a.get('venue-max-score', 0) < get_venue_score(a,
    ↪  blackboard) and a.get('food-max-score', 0) >= 0.5:
        return True
    return False

HC_POLICY.add_rule(DialogueRule(canthelp_food,
↪  'canthelp(food=<best-food-0>)'))
```

Note that this same rule schema is repeated with the slots **area** and **pricerange**.

The next rule is activated when there are no more alternative restaurants to be offered from the current cluster.

Listing 13 Rule for when there are no more alternative venues to offer

```
def canthelp_exception(*args, **kwargs):
    u, s, a = get_dialogue_state(kwargs)
    if (a.get('pricerange-max-score', 0) +
    ↪  a.get('area-max-score', 0) + a.get('food-max-score',
    ↪  0)) > 0:
        if a.get('venue-max-score', 0) > 0.5 and
        ↪  a.get('score-difference', 0) < 0.4:
            return True, {}
    return False, {}

HC_POLICY.add_rule(DialogueRule(canthelp_exception,
↪  'canthelp.exception(name=<best-venue>')))
```

Finally, the request-for-more rule is added which handles if the user wants something else at the end of the dialogue:

Listing 14 Rule for requesting the user if he/she wants something more

```
def reqmore(*args, **kwargs):
    # After an offer if the user says nothing or affirms
    u, s, a = get_dialogue_state(kwargs)
    if 'offer(venue=<best-venue>)' in s:
        if 'thankyou' u and 'bye' not in u:
            return True
        if  u.get('affirm',0) > 0.25:
            return True
        if not u: # Is empty
            return True
    return False

HC_POLICY.add_rule(DialogueRule(reqmore, 'reqmore'))
```

B.2.1 Conversion to Probabilistic Rules

The probabilistic rules follow the same logic as the deterministic ones, but instead of returning 1 or 0 as score, their score is calculated according to the probabilities and confidences of the user actions and the attributes of the current dialogue state.

To transform the deterministic rules into probabilistic, as explained in Chapter 6, the following conversions are applied:

- **Exists** x: $P(x \mid q_j)$
- **Not Exists** x: $1 - P(x \mid q_j)$
- x **And** y: $P(x \mid q_j) \cdot P(y \mid q_j)$
- x **Or** y: $\max(P(x \mid q_j), P(y \mid q_j)))$
- x **Greater than** y: $P(x > y \mid q_j)$
- x **Lower than** y: $P(x < y \mid q_j)$

Then, for example, the rule for giving information about a venue slot (food in this case) after a user request is converted to:

Listing 15 Probabilistic Rule for informing the food type of the venue after the user request.

```
def inform_food_request_p(*args, **kwargs):
    u, s, a = get_items(kwargs)
    score = u.get('request|food', 0)*_offer_score(a)
    return score

P_HC_POLICY.add_rule(ProbaDialogueRule(inform_food_request_p,
 ↪  ['offer(name=<best-venue>'),
 ↪  'inform(food=<venue-food>')]))
```

The same changes are applied to all the rules defined in the Deterministic Rules section of this appendix.

Bibliography

Adiwardana, Daniel, Minh-Thang Luong, David R So, et al. (2020). „Towards a human-like open-domain chatbot“. In: *Computing Research Repository* arXiv:2001.09977 (cit. on p. 16).

Arzelus, Haritz, Aitor Alvarez, Conrad Bernath, et al. (2018). „The Vicomtech-PRHLT Speech Transcription Systems for the IberSPEECH-RTVE 2018 Speech to Text Transcription Challenge.“ In: *IberSPEECH*, pp. 267–271 (cit. on p. 16).

Asher, Nicholas and Alex Lascarides (2001). „Indirect Speech Acts“. In: *Synthese* 128.1, pp. 183–228 (cit. on p. 10).

Azpeitia, Andoni, Manex Serras, Laura Garcıa-Sardiña, Mikel D Fernández-Bhogal, and Arantza del Pozo (2020). „Towards Similar User Utterance Augmentation for Out-of-Domain Detection“. In: *Conversational Dialogue Systems for the Next Decade*. Springer, pp. 289–302 (cit. on p. 16).

Bangor, Aaron, Philip Kortum, and James Miller (2009). „Determining what individual SUS scores mean: Adding an adjective rating scale“. In: *Journal of usability studies* 4.3, pp. 114–123 (cit. on p. 127).

Bellman, Richard (1957). „A Markovian decision process“. In: *Journal of mathematics and mechanics*, pp. 679–684 (cit. on p. 11).

Bocklisch, Tom, Joey Faulkner, Nick Pawlowski, and Alan Nichol (2017). „Rasa: Open source language understanding and dialogue management“. In: *arXiv preprint arXiv:1712.05181* (cit. on pp. 7, 21, 26, 131).

Bohus, Dan and Alexander I Rudnicky (2003). „RavenClaw: Dialog management using hierarchical task decomposition and an expectation agenda“. In: *Eighth European Conference on Speech Communication and Technology* (cit. on pp. 7, 30, 88).

– (2005a). „Error handling in the RavenClaw dialog management framework“. In: *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 225–232 (cit. on pp. 7, 26, 30).

– (2005b). „LARRI: A language-based maintenance and repair assistant“. In: *Spoken multimodal human-computer dialogue in mobile environments*. Springer, pp. 203–218 (cit. on pp. 2, 7, 24).

– (2009). „The RavenClaw dialog management framework: Architecture and systems“. In: *Computer Speech & Language* 23.3, pp. 332–361 (cit. on pp. 7, 30, 88).

Bordes, Antoine, Y-Lan Boureau, and Jason Weston (2016). „Learning end-to-end goal-oriented dialog“. In: *arXiv preprint arXiv:1605.07683* (cit. on p. 17).

Breiman, Leo (2001). „Random forests“. In: *Machine learning* 45.1, pp. 5–32 (cit. on p. 74).

Brooke, J (1986). „System Usability Scale (SUS): A Quick-and-dirty Method of System Evaluation User Information. Digital equipment co ltd“. In: *Reading, UK*, pp. 1–7 (cit. on p. 127).

Budzianowski, Paweł, Stefan Ultes, Pei-Hao Su, et al. (2017). „Sub-domain modelling for dialogue management with hierarchical reinforcement learning“. In: *arXiv preprint arXiv:1706.06210* (cit. on pp. 16, 22).

Cañas, Pablo and David Griol (2020). „Implementation of a Statistical Dialogue Manager for Commercial Conversational Systems“. In: *International Workshop on Soft Computing Models in Industrial and Environmental Applications*. Springer, pp. 383–393 (cit. on pp. 7, 8, 21, 26).

Casanueva, Inigo, Paweł Budzianowski, Pei-Hao Su, et al. (2018). „Feudal reinforcement learning for dialogue management in large domains“. In: *arXiv preprint arXiv:1803.03232* (cit. on pp. 16, 22).

Casanueva, Iñigo, Paweł Budzianowski, Pei-Hao Su, et al. (2017). „A Benchmarking Environment for Reinforcement Learning Based Task Oriented Dialogue Management“. In: *stat* 1050, p. 29 (cit. on pp. 11, 32).

Chandramohan, Senthilkumar, Matthieu Geist, Fabrice Lefevre, and Olivier Pietquin (2011). „User simulation in dialogue systems using inverse reinforcement learning“. In: *Interspeech 2011*, pp. 1025–1028 (cit. on p. 25).

Chen, Hongshen, Xiaorui Liu, Dawei Yin, and Jiliang Tang (2017). „A survey on dialogue systems: Recent advances and new frontiers“. In: *Acm Sigkdd Explorations Newsletter* 19.2, pp. 25–35 (cit. on pp. 1, 22, 24).

Colby, Kenneth Mark, Sylvia Weber, and Franklin Dennis Hilf (1971). „Artificial paranoia“. In: *Artificial Intelligence* 2.1, pp. 1–25 (cit. on p. 7).

Core, Mark G and James Allen (1997). „Coding dialogs with the DAMSL annotation scheme“. In: *AAAI fall symposium on communicative action in humans and machines*. Vol. 56. Boston, MA (cit. on p. 10).

Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer (2006). „Online passive-aggressive algorithms“. In: *Journal of Machine Learning Research* 7.Mar, pp. 551–585 (cit. on p. 74).

Crook, Paul A, Simon Keizer, Zhuoran Wang, Wenshuo Tang, and Oliver Lemon (2014). „Real user evaluation of a POMDP spoken dialogue system using automatic belief compression“. In: *Computer Speech & Language* 28.4, pp. 873–887 (cit. on p. 2).

Crook, Paul A and Alex Marin (2017). „Sequence to Sequence Modeling for User Simulation in Dialog Systems.“ In: *INTERSPEECH*, pp. 1706–1710 (cit. on pp. 16, 25).

Cuayáhuitl, Heriberto (2017). „Simpleds: A simple deep reinforcement learning dialogue system“. In: *Dialogues with Social Robots*. Springer, pp. 109–118 (cit. on pp. 16, 22).

Cuayáhuitl, Heriberto, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira (2005). „Human-computer dialogue simulation using hidden markov models“. In: *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*. IEEE, pp. 290–295 (cit. on pp. 25, 56, 69).

Cuayáhuitl, Heriberto, Seunghak Yu, Ashley Williamson, and Jacob Carse (2016). „Deep reinforcement learning for multi-domain dialogue systems“. In: *arXiv preprint arXiv:1611.08675* (cit. on pp. 16, 22).

Curry, Amanda Cercas, Ioannis Papaioannou, Alessandro Suglia, et al. (2018). „Alana v2: Entertaining and informative open-domain social dialogue using ontologies and entity linking“. In: *Alexa Prize Proceedings* (cit. on p. 26).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). „Bert: Pre-training of deep bidirectional transformers for language understanding“. In: *arXiv preprint arXiv:1810.04805* (cit. on pp. 16, 131).

Eckert, Wieland, Esther Levin, and Roberto Pieraccini (1997). „User modeling for spoken dialogue system evaluation“. In: *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*. IEEE, pp. 80–87 (cit. on pp. 24, 25).

Eshghi, Arash, Igor Shalyminov, and Oliver Lemon (2017). „Bootstrapping incremental dialogue systems from minimal data: the generalisation power of dialogue grammars“. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2220–2230 (cit. on pp. 14, 22, 24).

Etchegoyhen, Thierry, Haritz Arzelus, Harritxu Gete, et al. (2020). „MINTZAI: End-to-end Deep Learning for Speech Translation.“ In: *Procesamiento del Lenguaje Natural* 65 (cit. on p. 16).

Fromer, Jeanne C (1998). „Learning optimal discourse strategies in a spoken dialogue system“. PhD thesis. Massachusetts Institute of Technology (cit. on p. 14).

Garcıa-Sardiña, Laura, Manex Serras, Arantza del Pozo, and Mikel D Fernández-Bhogal (2020). „RESIVOZ: Dialogue System for Voice-based Information Registration in Eldercare“. In: *Procesamiento del Lenguaje Natural* 65, pp. 123–126 (cit. on p. 2).

Gašić, Milica, Catherine Breslin, Matthew Henderson, et al. (2013). „On-line policy optimisation of bayesian spoken dialogue systems via human interaction“. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 8367–8371 (cit. on pp. 15, 23, 28).

Gašić, Milica, Filip Jurčiček, Simon Keizer, et al. (2010). „Gaussian processes for fast policy optimisation of POMDP-based dialogue managers“. In: *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, pp. 201–204 (cit. on pp. 8, 11, 15, 22, 24).

Gašić, Milica, Filip Jurčiček, Blaise Thomson, Kai Yu, and Steve Young (2011). „On-line policy optimisation of spoken dialogue systems via live interaction with human subjects“. In: *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE, pp. 312–317 (cit. on pp. 23, 28).

Gasic, Milica, Simon Keizer, Francois Mairesse, et al. (2008). „Training and evaluation of the HIS POMDP dialogue system in noise“. In: *Proceedings of the 9th SIGDIAL Workshop on Discourse and Dialogue*, pp. 112–119 (cit. on p. 23).

Ghigi, Fabrizio and M Inés Torres (2015). „Decision making strategies for finite-state bi-automaton in dialog management“. In: *Natural Language Dialog Systems and Intelligent Assistants*. Springer, pp. 209–221 (cit. on pp. 8, 25, 27, 30, 32, 59–61, 96).

Gocgun, Yasin, Brian W Bresnahan, Archis Ghate, and Martin L Gunn (2011). „A Markov decision process approach to multi-category patient scheduling in a diagnostic facility“. In: *Artificial intelligence in medicine* 53.2, pp. 73–81 (cit. on p. 11).

Goodfellow, Ian J, Mehdi Mirza, Aaron Courville Da Xiao, and Yoshua Bengio (2014). „An empirical investigation of catastrophic forgeting in gradientbased neural networks“. In: *In Proceedings of International Conference on Learning Representations (ICLR*. Citeseer (cit. on pp. 23, 28).

Graesser, Arthur C, Kurt VanLehn, Carolyn P Rosé, Pamela W Jordan, and Derek Harter (2001). „Intelligent tutoring systems with conversational dialogue“. In: *AI magazine* 22.4, pp. 39–39 (cit. on p. 2).

Greco, Claudio, Barbara Plank, Raquel Fernández, and Raffaella Bernardi (2019). „Psycholinguistics Meets Continual Learning: Measuring Catastrophic Forgetting in Visual Question Answering“. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3601–3605 (cit. on pp. 23, 28).

Griol, David, Lluıs F Hurtado, Encarna Segarra, and Emilio Sanchis (2008). „A statistical approach to spoken dialog systems design and evaluation“. In: *Speech Communication* 50.8-9, pp. 666–682 (cit. on pp. 8, 11, 88).

Gupta, Narendra, Gokhan Tur, Dilek Hakkani-Tur, et al. (2005). „The AT&T spoken language understanding system“. In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.1, pp. 213–222 (cit. on p. 7).

Hancher, Michael (1979). „The classification of cooperative illocutionary acts“. In: *Language in society*, pp. 1–14 (cit. on p. 10).

Henderson, Matthew, Blaise Thomson, and Jason D Williams (2014). „The second dialog state tracking challenge“. In: *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pp. 263–272 (cit. on pp. 16, 33).

Henderson, Matthew, Blaise Thomson, and Steve Young (2014). „Word-based dialog state tracking with recurrent neural networks“. In: *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pp. 292–299 (cit. on p. 16).

Hernaez, Inma, Eva Navas, Juan Luis Murugarren, and Borja Etxebarria (2001). „Description of the AhoTTS system for the Basque language“. In: *4th ISCA Tutorial and Research Workshop (ITRW) on Speech Synthesis* (cit. on p. 120).

Higashinaka, Ryuichiro, Kotaro Funakoshi, Yuka Kobayashi, and Michimasa Inaba (2016). „The dialogue breakdown detection challenge: Task description, datasets, and evaluation metrics“. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 3146–3150 (cit. on p. 26).

Hinton, Geoffrey E (1990). „Connectionist learning procedures“. In: *Machine learning*. Elsevier, pp. 555–610 (cit. on p. 74).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). „Long short-term memory“. In: *Neural computation* 9.8, pp. 1735–1780 (cit. on p. 44).

Janarthanam, Srini (2017). *Hands-on chatbots and conversational UI development: build chatbots and voice user interfaces with Chatfuel, Dialogflow, Microsoft Bot Framework, Twilio, and Alexa Skills*. Packt Publishing Ltd (cit. on pp. 7, 21, 25, 26).

Jurčíček, Filip, Blaise Thomson, and Steve Young (2011). „Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as POMDPs“. In: *ACM Transactions on Speech and Language Processing (TSLP)* 7.3, pp. 1–26 (cit. on p. 23).

Kaiser, Edward C (1999). „Robust, finite-state parsing for spoken language understanding“. In: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, pp. 573–578 (cit. on p. 30).

Khatri, Chandra, Behnam Hedayatnia, Anu Venkatesh, et al. (2018). „Advancing the state of the art in open domain dialog systems through the alexa prize“. In: *arXiv preprint arXiv:1812.10757* (cit. on pp. 7, 21, 26).

Kim, Seokhwan, Luis Fernando D'Haro, Rafael E Banchs, Jason D Williams, and Matthew Henderson (2017). „The fourth dialog state tracking challenge“. In: *Dialogues with Social Robots*. Springer, pp. 435–449 (cit. on p. 16).

Kim, Seokhwan, Luis Fernando D'Haro, Rafael E Banchs, et al. (2016). „The fifth dialog state tracking challenge“. In: *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 511–517 (cit. on p. 16).

Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, et al. (2017). „Overcoming catastrophic forgetting in neural networks“. In: *Proceedings of the national academy of sciences* 114.13, pp. 3521–3526 (cit. on pp. 23, 28).

Layla, El Asri, He Jing, and Kaheer Suleman (2016). „A Sequence-to-Sequence Model for User Simulation in Spoken Dialogue Systems“. In: *Interspeech* (cit. on pp. 16, 25, 43, 56, 69, 70).

Lee, Sang-Woo, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang (2017). „Overcoming catastrophic forgetting by incremental moment matching“. In: *Advances in neural information processing systems*, pp. 4652–4662 (cit. on pp. 23, 28).

Lei, Shuyu, Xiaojie Wang, and Caixia Yuan (2019). „Word-Based POMDP Dialog Management via Hybrid Learning“. In: *IEEE Access* 7, pp. 39236–39243 (cit. on p. 15).

Lemon, Oliver, Alexander Gruenstein, Alexis Battle, and Stanley Peters (2002). „Multi-tasking and collaborative activities in dialogue systems“. In: *Proceedings of the Third SIGdial Workshop on Discourse and Dialogue*, pp. 113–124 (cit. on p. 7).

Levin, Esther, Roberto Pieraccini, and Wieland Eckert (1997). „Learning dialogue strategies within the markov decision process framework“. In: *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*. IEEE, pp. 72–79 (cit. on pp. 11, 12).

– (1998). „Using Markov decision process for learning dialogue strategies“. In: *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*. Vol. 1. IEEE, pp. 201–204 (cit. on pp. 8, 11, 12, 14).

– (2000). „A stochastic model of human-machine interaction for learning dialog strategies“. In: *IEEE Transactions on speech and audio processing* 8.1, pp. 11–23 (cit. on pp. 14, 25).

Li, Jiwei, Will Monroe, Tianlin Shi, et al. (2017). „Adversarial Learning for Neural Dialogue Generation“. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2157–2169 (cit. on p. 16).

Liang, Weixin, Youzhi Tian, Chengcai Chen, and Zhou Yu (2020). „MOSS: End-to-End Dialog System Framework with Modular Supervision.“ In: *AAAI*, pp. 8327–8335 (cit. on p. 17).

Lipton, Zachary C, Sharad Vikram, and Julian McAuley (2015). „Generative concatenative nets jointly learn to write and classify reviews“. In: *arXiv preprint arXiv:1511.03683* (cit. on p. 43).

Lipton, Zachary, Xiujun Li, Jianfeng Gao, et al. (2018). „Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems“. In: *Thirty-Second AAAI Conference on Artificial Intelligence* (cit. on pp. 16, 22).

Lison, Pierre (2015). „A hybrid approach to dialogue management based on probabilistic rules“. In: *Computer Speech & Language* 34.1, pp. 232–255 (cit. on pp. 7, 27).

Lison, Pierre and Casey Kennington (2016). „Opendial: A toolkit for developing spoken dialogue systems with probabilistic rules“. In: *Proceedings of ACL-2016 system demonstrations*, pp. 67–72 (cit. on pp. 7, 27).

Liu, Bing and Ian Lane (2018). „End-to-end learning of task-oriented dialogs“. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pp. 67–73 (cit. on p. 17).

Lopes, José (2017). „How Generic Can Dialogue Breakdown Detection Be? The KTH entry to DBDC3“. In: *Proceedings of Dialog System Technology Challenge* 6 (cit. on p. 26).

Lu, Keting, Shiqi Zhang, and Xiaoping Chen (2019). „Goal-oriented dialogue policy learning from failures“. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 2596–2603 (cit. on p. 57).

Lubold, Nichola, Erin Walker, and Heather Pon-Barry (2016). „Effects of voice-adaptation and social dialogue on perceptions of a robotic learning companion“. In: *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, pp. 255–262 (cit. on p. 2).

Lucas, Bruce (2000). „VoiceXML for Web-based distributed conversational applications“. In: *Communications of the ACM* 43.9, pp. 53–57 (cit. on p. 7).

Milhorat, Pierrick, Divesh Lala, Koji Inoue, et al. (2019). „A conversational dialogue manager for the humanoid robot ERICA“. In: *Advanced Social Interaction with Agents*. Springer, pp. 119–131 (cit. on pp. 22, 26).

Orozko, Odei Rey and M Inés Torres (2015). „Online learning of stochastic bi-automaton to model dialogues“. In: *Iberian Conference on Pattern Recognition and Image Analysis*. Springer, pp. 441–451 (cit. on pp. 25, 27, 28, 30, 32, 59–61, 93, 98, 99).

Pablos, Aitor Garcıa, Naiara Perez, and Montse Cuadros (2020). „Sensitive Data Detection and Classification in Spanish Clinical Text: Experiments with BERT“. In: *Proceedings of The 12th Language Resources and Evaluation Conference*, pp. 4486–4494 (cit. on p. 16).

Paek, Tim and Roberto Pieraccini (2008). „Automating spoken dialogue management design using machine learning: An industry perspective“. In: *Speech communication* 50.8-9, pp. 716–729 (cit. on pp. 8, 15, 21, 22, 24–26).

Papaioannou, Ioannis, Amanda Cercas Curry, Jose L Part, et al. (2017). „Alana: Social dialogue using an ensemble model and a ranker trained on user feedback“. In: *Alexa Prize Proceedings* (cit. on p. 26).

Parr, Ronald and Stuart Russell (1995). „Approximating optimal policies for partially observable stochastic domains“. In: *IJCAI*. Vol. 95, pp. 1088–1094 (cit. on p. 14).

Pedregosa, F., G. Varoquaux, A. Gramfort, et al. (2011). „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12, pp. 2825–2830 (cit. on p. 74).

Pérez, Alicia, M Inés Torres, and Francisco Casacuberta (2008). „Joining linguistic and statistical methods for Spanish-to-Basque speech translation“. In: *Speech Communication* 50.11-12, pp. 1021–1033 (cit. on p. 27).

Perez, Naiara, Manex Serras, and Arantza Del Pozo (2019). „Vicomtech at MEDDOCAN: Medical Document Anonymization.“ In: (cit. on p. 16).

Pietquin, Olivier (2005). *A framework for unsupervised learning of dialogue strategies*. Presses univ. de Louvain (cit. on p. 25).

Pietquin, Olivier and Thierry Dutoit (2006). „A probabilistic framework for dialog simulation and optimal strategy learning“. In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.2, pp. 589–599 (cit. on p. 25).

Pineau, Joelle, Michael Montemerlo, Martha Pollack, Nicholas Roy, and Sebastian Thrun (2003). „Towards robotic assistants in nursing homes: Challenges and results“. In: *Robotics and autonomous systems* 42.3-4, pp. 271–281 (cit. on p. 2).

Pineau, Joelle and Sebastian Thrun (2001). „Hierarchical POMDP decomposition for a conversational robot“. In: *ICML Workshop on Hierarchy and Memory* (cit. on p. 16).

Platt, John C. (1999). „Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods“. In: *ADVANCES IN LARGE MARGIN CLASSIFIERS*. MIT Press, pp. 61–74 (cit. on p. 74).

Quarteroni, Silvia, Meritxell González, Giuseppe Riccardi, and Sebastian Varges (2010). „Combining user intention and error modeling for statistical dialog simulators.“ In: *INTERSPEECH*, pp. 3022–3025 (cit. on p. 56).

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). *Improving language understanding by generative pre-training* (cit. on p. 16).

Radford, Alec, Jeffrey Wu, Rewon Child, et al. (2019). „Language models are unsupervised multitask learners“. In: *OpenAI Blog* 1.8, p. 9 (cit. on p. 16).

Ram, Ashwin, Rohit Prasad, Chandra Khatri, et al. (2018). „Conversational ai: The science behind the alexa prize“. In: *arXiv preprint arXiv:1801.03604* (cit. on p. 26).

Raux, Antoine, Dan Bohus, Brian Langner, Alan W Black, and Maxine Eskenazi (2006). „Doing research on a deployed spoken dialogue system: One year of Let's Go! experience“. In: *Ninth International Conference on Spoken Language Processing* (cit. on pp. 2, 7, 32).

Raux, Antoine, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi (2005). „Let's Go Public! Taking a spoken dialog system to the real world“. In: *Ninth European conference on speech communication and technology* (cit. on pp. 7, 32).

Reidsma, Dennis, Vicky Charisi, Daniel Davison, et al. (2016). „The EASEL project: Towards educational human-robot symbiotic interaction“. In: *Conference on Biomimetic and Biohybrid Systems*. Springer, pp. 297–306 (cit. on p. 2).

Rieser, Verena (2008). *Bootstrapping reinforcement learning-based dialogue strategies from wizard-of-oz data*. DFKI (cit. on p. 24).

Roller, Stephen, Emily Dinan, Naman Goyal, et al. (2020). „Recipes for building an open-domain chatbot“. In: *Computing Research Repository* arXiv:2004.13637 (cit. on p. 16).

Roy, Nicholas, Joelle Pineau, and Sebastian Thrun (2000). „Spoken dialogue management using probabilistic reasoning“. In: *Proceedings of the 38th annual meeting of the association for computational linguistics*, pp. 93–100 (cit. on pp. 12, 14).

Russell, Stuart, Peter Norvig, and A Intelligence (1995). „A modern approach“. In: *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs* 25.27, pp. 79–80 (cit. on p. 14).

Schatzmann, Jost, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young (2007). „Agenda-based user simulation for bootstrapping a POMDP dialogue system“. In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*. Association for Computational Linguistics, pp. 149–152 (cit. on pp. 14, 25, 34, 38, 40).

Schatzmann, Jost, Blaise Thomson, and Steve Young (2007a). „Error simulation for training statistical dialogue systems“. In: *2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*. IEEE, pp. 526–531 (cit. on pp. 14, 51).

– (2007b). „Statistical user simulation with a hidden agenda“. In: *Proc SIGDial, Antwerp* 273282.9 (cit. on p. 15).

Schatzmann, Jost, Karl Weilhammer, Matt Stuttle, and Steve Young (2006). „A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies“. In: *The knowledge engineering review* 21.2, pp. 97–126 (cit. on pp. 3, 22, 24, 56).

Schütze, Hinrich, Christopher D Manning, and Prabhakar Raghavan (2008). „Introduction to information retrieval“. In: *Proceedings of the international communication of association for computing machinery conference*, p. 260 (cit. on pp. 74, 88).

Serban, Iulian V, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau (2015). „Building end-to-end dialogue systems using generative hierarchical neural network models“. In: *arXiv preprint arXiv:1507.04808* (cit. on p. 16).

Serras, Manex, Laura Garcıa-Sardiña, Bruno Simões, Hugo Álvarez, and Jon Arambarri (2020a). „AREVA: Augmented Reality Voice Assistant for Industrial Maintenance“. In: *Procesamiento del Lenguaje Natural* 65, pp. 135–138 (cit. on pp. 2, 7, 120).

– (2020b). „Dialogue Enhanced Extended Reality: Interactive System for the Operator 4.0“. In: *Applied Sciences* 10.11, p. 3960 (cit. on p. 2).

Serras, Manex, Marıa Inés Torres, and Arantza Del Pozo (2017). „Online learning of attributed bi-automata for dialogue management in spoken dialogue systems“. In: *Iberian Conference on Pattern Recognition and Image Analysis*. Springer, pp. 22–31 (cit. on pp. 11, 24, 25, 60, 61).

– (2019a). „Goal-conditioned User Modeling for Dialogue Systems using Stochastic Bi-Automata“. In: *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,* INSTICC. SciTePress, pp. 128–134 (cit. on pp. 69, 72).

– (2019b). „Regularized neural user model for goal-oriented spoken dialogue systems“. In: *Advanced Social Interaction with Agents*. Springer, pp. 235–245 (cit. on pp. 17, 25, 43, 69, 74).

– (2019c). „User-aware dialogue management policies over attributed bi-automata“. In: *Pattern Analysis and Applications* 22.4, pp. 1319–1330 (cit. on pp. 60, 61).

– (2020). „Improving Dialogue Smoothing with A-priori State Pruning“. In: *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,* INSTICC. SciTePress, pp. 607–614 (cit. on p. 25).

Sezer, Volkan (2018). „Intelligent decision making for overtaking maneuver using mixed observable markov decision process“. In: *Journal of Intelligent Transportation Systems* 22.3, pp. 201–217 (cit. on p. 11).

Shah, Pararth, Dilek Hakkani-Tur, Bing Liu, and Gokhan Tur (2018). „Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning“. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pp. 41–51 (cit. on pp. 22, 24).

Shalyminov, Igor, Sungjin Lee, Arash Eshghi, and Oliver Lemon (2019a). „Data-Efficient Goal-Oriented Conversation with Dialogue Knowledge Transfer Networks“. In: *CoRR* abs/1910.01302. arXiv: 1910.01302 (cit. on pp. 17, 24).

– (2019b). „Few-Shot Dialogue Generation Without Annotated Data: A Transfer Learning Approach“. In: *CoRR* abs/1908.05854. arXiv: 1908.05854 (cit. on pp. 17, 24).

Singh, Satinder, Michael Kearns, Diane J Litman, Marilyn A Walker, et al. (2000). „Empirical evaluation of a reinforcement learning spoken dialogue system“. In: *AAAI/IAAI*, pp. 645–651 (cit. on p. 14).

Sutton, Richard S and Andrew G Barto (1998). *Reinforcement learning: An introduction*. MIT press (cit. on pp. 14, 87).

Thomson, Blaise, Jost Schatzmann, Karl Weilhammer, Hui Ye, and Steve Young (2007). „Training a real-world POMDP-based dialog system“. In: *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, pp. 9–16 (cit. on p. 16).

Thomson, Blaise, Kai Yu, Simon Keizer, et al. (2010). „Bayesian dialogue system for the Let's Go spoken dialogue challenge“. In: *2010 IEEE Spoken Language Technology Workshop*. IEEE, pp. 460–465 (cit. on p. 88).

Torres, Inés and Amparo Varona (2001). „k-TSS language models in speech recognition systems“. In: *Computer Speech & Language* 15.2, pp. 127–148 (cit. on p. 27).

Torres, M Inés (2013). „Stochastic bi-languages to model dialogs“. In: *Proceedings of the 11th international conference on finite state methods and natural language processing*, pp. 9–17 (cit. on pp. 4, 19).

Torres, M Inés and Francisco Casacuberta (2011). „Stochastic K-TSS bi-languages for Machine Translation“. In: *Proceedings of the 9th International Workshop on Finite State Methods and Natural Language Processing*, pp. 98–106 (cit. on p. 18).

Walker, Marilyn A, Diane J Litman, Candace A Kamm, and Alicia Abella (1997). „PARADISE: a framework for evaluating spoken dialogue agents“. In: *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 271–280 (cit. on pp. 14, 32).

Wang, Weikang, Jiajun Zhang, Qian Li, et al. (2019). „Incremental Learning from Scratch for Task-Oriented Dialogue Systems“. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3710–3720 (cit. on pp. 23, 24, 28).

Wang, Ye-Yi (1999). „A robust parser for spoken language understanding“. In: *Sixth European Conference on Speech Communication and Technology* (cit. on p. 30).

Ward, Wayne (1990). „The CMU air travel information service: Understanding spontaneous speech“. In: *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990* (cit. on pp. 30, 119, 126).

Ward, Wayne (1994). „Extracting information in spontaneous speech“. In: *Third International Conference on Spoken Language Processing* (cit. on p. 7).

Weizenbaum, Joseph (1966). „ELIZA—a computer program for the study of natural language communication between man and machine“. In: *Communications of the ACM* 9.1, pp. 36–45 (cit. on p. 7).

Wen, Tsung-Hsien, David Vandyke, Nikola Mrksic, et al. (2016). „A network-based end-to-end trainable task-oriented dialogue system“. In: *arXiv preprint arXiv:1604.04562* (cit. on p. 16).

Wessel, Michael, Girish Acharya, James Carpenter, and Min Yin (2019). „OntoVPA—an ontology-based dialogue management system for virtual personal assistants“. In: *Advanced Social Interaction with Agents*. Springer, pp. 219–233 (cit. on p. 25).

White III, Chelsea C and Douglas J White (1989). „Markov decision processes“. In: *European Journal of Operational Research* 39.1, pp. 1–16 (cit. on pp. 11, 12).

Williams, Jason D (2008). „The best of both worlds: Unifying conventional dialog systems and POMDPs“. In: *Ninth Annual Conference of the International Speech Communication Association* (cit. on p. 27).

Williams, Jason D, Matthew Henderson, Antoine Raux, et al. (2014). „The dialog state tracking challenge series“. In: *AI Magazine* 35.4, pp. 121–124 (cit. on p. 16).

Williams, Jason D, Antoine Raux, and Matthew Henderson (2016). „The dialog state tracking challenge series: A review“. In: *Dialogue & Discourse* 7.3, pp. 4–33 (cit. on p. 16).

Williams, Jason D and Steve Young (2005). „Scaling up POMDPs for Dialog Management: The“Summary POMDP”Method“. In: *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.* IEEE, pp. 177–182 (cit. on p. 16).

– (2007). „Scaling POMDPs for spoken dialog management“. In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.7, pp. 2116–2129 (cit. on pp. 15, 16).

Wolf, Thomas, Victor Sanh, Julien Chaumond, and Clement Delangue (2019). „Transfertransfo: A transfer learning approach for neural network based conversational agents“. In: *Computing Research Repository* arXiv:1901.08149 (cit. on p. 16).

Wu, Yonghui, Mike Schuster, Zhifeng Chen, et al. (2016). „Google’s neural machine translation system: Bridging the gap between human and machine translation“. In: *arXiv preprint arXiv:1609.08144* (cit. on p. 131).

Xie, Zeying and Guang Ling (2017). „Dialogue breakdown detection using hierarchical bi-directional LSTMs“. In: *Proceedings of the Dialog System Technology Challenges Workshop (DSTC6)* (cit. on p. 26).

Xu, Lin, Qixian Zhou, Ke Gong, et al. (2019). „End-to-end knowledge-routed relational dialogue system for automatic diagnosis“. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 7346–7353 (cit. on p. 17).

Yoshino, Koichiro, Shinji Watanabe, Jonathan Le Roux, and John R Hershey (2013). „Statistical dialogue management using intention dependency graph“. In: *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pp. 962–966 (cit. on p. 27).

Young, Steve (2006). „Using POMDPs for dialog management“. In: *2006 IEEE Spoken Language Technology Workshop*. IEEE, pp. 8–13 (cit. on pp. 15, 47, 88).

Young, Steve J (2000). „Probabilistic methods in spoken–dialogue systems“. In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 358.1769, pp. 1389–1402 (cit. on p. 14).

Young, Steve, Milica Gašić, Simon Keizer, et al. (2010). „The hidden information state model: A practical framework for POMDP-based spoken dialogue management“. In: *Computer Speech & Language* 24.2, pp. 150–174 (cit. on pp. 15, 16, 47).

Zhang, Bo, Qingsheng Cai, Jianfeng Mao, Eric Chang, and Baining Guo (2001). „Spoken dialogue management as planning and acting under uncertainty“. In: *Seventh European conference on speech communication and technology* (cit. on p. 15).

Zhao, Tiancheng and Maxine Eskenazi (2016). „Towards End-to-End Learning for Dialog State Tracking and Management using Deep Reinforcement Learning“. In: *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 1–10 (cit. on p. 17).

Zhu, Wenya, Kaixiang Mo, Yu Zhang, et al. (2017). „Flexible end-to-end dialogue system for knowledge grounded conversation“. In: *arXiv preprint arXiv:1709.04264* (cit. on p. 17).

List of Figures

List of Tables