YARBUS : Yet Another Rule Based belief Update System Jérémy Fix Hervé Frezza-Buet

Jérémy Fix, Hervé Frezza-Buet

vv (2015) pp doi: doi

YARBUS : Yet Another Rule Based belief Update System

Jérémy Fix JEREMY.FIX@CENTRALESUPELEC.FR
IMS - MaLIS & UMI 2958 (GeorgiaTech - CNRS)
Centale-Supélec, 2 rue Edouard Belin
57070 Metz, France

Hervé Frezza-Buet HERVE.FREZZA-BUET@CENTRALESUPELEC.FR
IMS - MaLIS & UMI 2958 (GeorgiaTech - CNRS)
Centale-Supélec, 2 rue Edouard Belin
57070 Metz, France

Editor:

Abstract

We introduce a new rule based system for belief tracking in dialog systems. Despite the simplicity of the rules being considered, the proposed belief tracker ranks favourably compared to the previous submissions on the second and third Dialog State Tracking challenges. The results of this simple tracker allows to reconsider the performances of previous submissions using more elaborate techniques.

Keywords: Belief tracking, rule based

1. Introduction

Spoken dialog systems are concerned with producing algorithms allowing a user to interact in natural language with a machine. Particularly important tasks of spoken dialog systems are form-filling tasks (Goddeau et al., 1996). Booking a flight ticket, querying a restaurant proposing a specific kind of food, looking for a bus from one destination to another can all be framed as a form-filling task. In these tasks, a predetermined and fixed set of slots is filled by the machine as the user interacts with it. The form is actually invisible to the user and allows the machine to bias the direction of the dialog in order to uncover the intents of the user. For example, to correctly indicate a bus from one destination to another, the machine has to obtain all the relevant informations such as the source and the destination as well as the time schedule. A typical dialog system is a loop connecting the speech of the user to the spoken synthesis of the machine reply (Rieser and Lemon, 2011), with a pipeline of processes. Several modules come into play in between these two spoken utterances. The utterance of the user is first processed by an Automatic Speech Recognizer (ASR), feeding a Spoken Language Understanding (SLU) module which outputs a probability distribution over so-called dialog acts. Sequences of dialog acts are handful representations of the intention of a user. These are then integrated by the dialog manager which is in charge of producing the dialog act of the machine which is converted to text and synthesized. There are various sources of noise that can impair the course of the dialog. One of interest for the following of the paper is the noise produced when recognizing what the user said and transcribing it into sequences of dialog acts. Typical semantic parsers (SLU) therefore produce a probability distribution over sequences of dialog acts that reflect

 Submitted xx/xx; Accepted yy/yy; Published online zz/zz

what the user might have said. In this paper we focus on integrating these hypotheses in order to infer what the user goal is. This task is the belief tracking, a subpart of the dialog manager.

As any machine learning implementation, evaluating the performances of an algorithm requires data. The recently proposed dialog state tracking challenges offer the opportunity to test belief tracking algorithms on common data (Black et al., 2011; Williams et al., 2013; Henderson et al., 2014b). The first challenge focused on form-filling, the second added the possibility of goal changes (especially when the constraints provided by the user are too restrictive to bring any possibility) and the third challenge added the difficulty that there are just few labeled data. In this paper, we work on the datasets for the second and third challenges that focus on the same domain of finding a restaurant by providing constraints on their location, food type, name and price range.

There has been a variety of methods for inferring and tracking the goal of the user submitted to this challenge. Some these methods directly work with the live SLU provided in the dataset (e.g. the focus baseline or the Heriot-Watt tracker (Wang and Lemon, 2013)). It turns out that the live SLU is of a rather poor quality and some authors suggested alternative semantic parsers (Sun et al., 2014b; Williams, 2014) or trackers working directly from the ASR output (Henderson et al., 2014c). As a dialog is performed turn by turn, belief tracking can be formulated as an iterative process in which new evidences provided by the SLU are integrated with the previous belief to produce the new belief. In Sun et al. (2014a), the authors consider the slots to be independent and learn an update rule of the marginal distributions over the goals. The rule they train, taking as one of the inputs the previous belief, is a polynomial function of the probabilities that the user informed or denied a value for a slot, informed or denied a value different from the one for which the output is computed. As the size of hypothesis space grows exponentially with the degree of the polynomial, constraints are introduced in order to prune the space of explored models and to render tractable the optimization problem. In Henderson et al. (2014c), the authors explore the ability of recurrent neural networks to solve the belief tracking problem by taking directly as input the speech recognizer output; it does not require any semantic parser. Their work benefits from the recent developments in deep learning. The number of parameters to learn is so large that if the network is not trained carefully, it would not be able to perform well. As shown by the authors, the recurrent neural network performs well on the dataset and their sensitivity to the history of the inputs certainly contribute to their performance. Williams (2014) brought several contributions. The first is the proposition of building up multiple SLU to feed a belief tracker. The second relies in identifying the problem of belief tracking with the problem of the ranking of relevance of answers (documents) to queries which leads the author to propose an interesting approach to the scoring of joint goals. Similarly to document ranking, features (around 2000 to 3000) are extracted from the SLU hypotheses and machine utterance and a regressor is trained to score the different joint goals accumulated so far in the dialog. The tracker proposed by Williams (2014) ranked first at the time of the challenge evaluation.

One of the attractiveness of the last two methods is their ability to solve the belief tracking problem without requiring much of expert knowledge in dialog systems. These methods extract a medium to large set of features feeding a regressor trained on the datasets. However, there is one potential caveat in these methods which comes from their black-box approach. Indeed, the results of the authors certainly show that the method consisting in exploding the number of attributes extracted from a turn (or previous turns as well) and then training a regressor on this large set of features performs favourably. Nevertheless, at the same time, it tends to lose the grip on the very nature of the data that are processed. As we shall see in this paper, the very limited set of rules employed in YARBUS is extremely simple, yet effective. The paper is organized as follows. Section 2 presents

all the steps as well as the rules employed in the YARBUS belief tracker. The results of the YARBUS tracker on the DSTC2 and DSTC3 challenges are given in section 3 and a discussion concludes the paper. The source code used to produce the results of the paper are shared online(Fix and Frezza-Buet, 2015).

2. Methods

The belief tracker we propose computes a probability distribution over the joint goals iteratively. At each turn of a dialog, the utterance of the user is processed by the Automatic Speech Recognized (ASR) module which inputs the semantic parser (SLU) which outputs a probability distribution over dialog acts, called the SLU hypotheses in the rest of the paper. In this paper, three SLU are considered: the live SLU provided originally in the dataset as well as the SLU proposed in (Sun et al., 2014b) for the DSTC2 challenge and in (Zhu et al., 2014) for the DSTC3 challenge. YARBUS proceeds as following: some pre-processing are operated on the machine acts (getting rid of REPEAT () acts) and on the SLU hypotheses (solving the reference of `"this"`), then informations are extracted from these reprocessed hypotheses and the belief is updated. Before explaining in details all these steps in the next sections, we introduce some handful notations.

2.1 Dialog State Tracking Challenge datasets

In the following of the paper, we focus on the datasets provided by the Dialog State Tracking Challenges 2 and 3. These contain labeled dialogs for the form-filling task of finding a restaurant. In this section, we provide the keypoints about these datasets and a full detailed description and the data can be found in Henderson et al. (2013–2014). There are four slots to fill in the DSTC-2 challenge[1] : $\mathbf{area}$ (6 values), $\mathbf{name}$ (114 values), $\mathbf{food}$ (92 values) and $\mathbf{pricerange}$ (4 values). This leads to a joint goal space of 374325 elements including the fact one slot might be unknown. The joint goal space is significantly larger in the third challenge. Indeed, in the third challenge, there are 9 slots :$\mathbf{area}$ (16 values),$\mathbf{chilrenallowed}$ (3 values), $\mathbf{food}$ (29 values), $\mathbf{hasinternet}$(3 values), $\mathbf{hastv}$ (3 values), $\mathbf{name}$ (164 values), $\mathbf{near}$ (53 values), $\mathbf{pricerange}$ (5 values) and $\mathbf{type}$ (4 values). This leads to a joint goal space of more than 8.10^9 elements.

The DSTC-2 challenge contains three datasets: a training set (dstc2_train) of 1612 dialogs with a total of 11677 turns, a development set (dstc2_dev) of 506 dialogs with 3934 turns and a test set (dstc2_test) of 1117 dialogs with 9890 turns. At the time of the challenge, the labeled of only the two first sets were released but we now have the labels for the third subset. The DSTC-3 challenge, which addressed the question of belief tracking when just few labeled data were available and also used a larger set of informable slots, contains two subsets : a training subset (dstc3_seed) of 10 labeled dialogs with 109 turns and a test set (dstc3_test) of 2264 dialogs with 18715 turns. The DSTC-2 challenge data contain dialogs in which the machines was driven by one out of three dialog managers in two acoustic conditions and the data of the third challenge were collected with one out of four dialog managers, all in the same acoustic conditions.

1. the count of values for each slot takes into account the special value `"dontcare"`

2.2 Notations

The elements handeled by the belief tracking process are strings. Let us denote the set of strings by $\texttt{string}$. For a set A, let us also define $A_{\subset}$ as the set of all the *finite* subsets of A (this is a way to represent lists[2] with distinct elements) and B^A the set of the functions from A to B. As the context is the filling of the values for the slots in a form, let us denote by $\mathcal{S} = \left\{s_1, \cdots, s_2, \cdots, s_{|\mathcal{S}|}\right\} \in \texttt{string}_{\subset}$ the different slots to be filled. The definition of the slots is part of the ontology related to the dialog. The ontology also defines the acceptable values for each slot. Let us model the slot definition domain thanks to a function $\text{val} \in \texttt{string}_{\subset}{}^{\mathcal{S}}$ which defines $\text{val}(s) = \{v_1, v_2, \cdots, v_{n_s}\}$ as the set of acceptable values for the slot s. Let us consider two extra slot values $\star$ and $\texttt{?}$, respectively meaning that the user does not care about the value of the slot and that the system does not know what the user wants for this slot.

Let us call a *goal* the status of a form, where a slot can be informed or not, depending on what has been said during the interaction with the user. For each slot s, a goal specifies a value in $\text{val}(s) \cup \{\star\}$ if the slot is informed in the goal, or $\texttt{?}$ if it has not been informed yet. Using the notations $A_\star = A \cup \{\star\}$, $A_\texttt{?} = A \cup \{\texttt{?}\}$, $A_{\star\texttt{?}} = A \cup \{\star, \texttt{?}\}$, a goal can be defined as $g \in \mathcal{G}$, with

$$\mathcal{G} = \left\{ g \in (\texttt{string}_{\star\texttt{?}})^{\mathcal{S}} \;\middle|\; \forall s \in \mathcal{S},\ g(s) \in \text{val}(s)_{\star\texttt{?}} \right\} \tag{1}$$

Let us now formalize utterances. Utterances are made of dialog acts, that may differ according to the speaker (user or machine). Acts are coded as a label and a set of slot-value pairs. The labels for machine acts are denoted by $\mathbb{M} = \{\texttt{"affirm"}, \texttt{"bye"}, \texttt{"canthear"}, \cdots\}$ and the ones for user acts by $\mathbb{U} = \{\texttt{"ack"}, \texttt{"affirm"}, \texttt{"bye"}, \cdots\}$. For example ACK (), INFORM ($\mathbf{food}$=$\star$) and CANTHELP ($\mathbf{food}$=$british$, $\mathbf{area}$=$south$) are dialog acts[3]. Let the machine acts be the elements in

$$\mathcal{M} = \left\{ (a, args) \in \mathbb{M} \times (\mathcal{S} \times \texttt{string}_\star)_{\subset} \;\middle|\; \forall (s, v) \in args,\ v \in \text{val}(s)_\star \right\} \tag{2}$$

Let us define user acts $\mathcal{U}$ similarly, using $\mathbb{U}$ instead of $\mathbb{M}$.

$$\mathcal{U} = \left\{ (a, args) \in \mathbb{U} \times (\mathcal{S} \times \texttt{string}_\star)_{\subset} \;\middle|\; \forall (s, v) \in args,\ v \in \text{val}(s)_\star \right\} \tag{3}$$

We can thus denote a machine utterance by $m \in \mathcal{M}_{\subset}$ and a user utterance by $u \in \mathcal{U}_{\subset}$. The SLU hypotheses are a set of user utterances with a probability for each one. Let us use the notation[4] $\widetilde{A} = \left\{ f \in [0,1]^A \;\middle|\; \sum_{a \in A} f(a) = 1 \right\}$ for the definition of the SLU hypotheses space $\mathcal{H} = \widetilde{\mathcal{U}_{\subset}}$. An SLU hypothesis is thus denoted by $h \in \mathcal{H}$.

The principle of the rule based belief tracker presented in this paper is to handle a distribution probability $b \in \mathcal{B} = \widetilde{\mathcal{G}}$ that is updated at each dialog turn t. The update (or transition) function $\tau \in \mathcal{B}^{\mathcal{B} \times \mathcal{M}_{\subset} \times \mathcal{H}}$ is used as

$$b_t = \tau(b_{t-1}, m_t, h_t) \tag{4}$$

As detailed further, the update τ is based on a process that extracts informations from the current SLU hypotheses h_t and the current machine utterance m_t. This process consists formally in extracting informations from every user utterances $u \in \mathcal{U}_{\subset}$. However, in the implementation, this

2. E.g. $\{a, b, c\}_{\subset} = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \cdots, \{a, b, c\}\}$
3. For the sake of clarity, a slot-value pair $(\texttt{"a"}, \texttt{"b"})$ is denoted by $\mathbf{a}$=b and a dialog act $(\texttt{"act"}, \{(\texttt{"slot"}, \texttt{"value"}), (\texttt{"foo"}, \texttt{"bar"})\})$ by ACT ($\mathbf{slot}$=$value$, $\mathbf{foo}$=bar)
4. The definition stands for finite sets.

combinatorial extraction is avoided by taking the probability $h(u)$ into account and therefore ignoring the huge amounts of null ones. Let us define here what an information about a dialog turn is. Let us use $\overline{A} = \{\neg a \mid a \in A\}$ the set of the negated values of A. An information $i \in \mathcal{I}$ is a function that associates for each slot a value that can be a string, $\star$ if the user does not care about the slot value or ? if nothing is known about what the user wants for that slot. An information can also be a set of negated values ($\star$ can be negated as well), telling that it is known that the user does not want any of the values negated in that set. This leads to the following definition for $\mathcal{I}$.

$$\begin{aligned} V &= (\texttt{string}_{\star\,?})_{\{1\}} \cup \left(\overline{\texttt{string}_{\star}}\right)_{\subset} \\ \mathcal{I} &= \left\{ i \in V^{\mathcal{S}} \;\middle|\; \forall s \in \mathcal{S},\ i(s) \in \left(\mathrm{val}(s)_{\star\,?}\right)_{\{1\}} \cup \left(\overline{\mathrm{val}(s)_{\star}}\right)_{\subset} \right\} \end{aligned} \tag{5}$$

where $A_{\{1\}}$ denotes $\{\{a\} \mid a \in A\}$. For example, $i(\mathbf{area}) = \{east\}$, $i(\mathbf{area}) = \{\star\}$, $i(\mathbf{area}) = \{?\}$, $i(\mathbf{area}) = \{\neg east, \neg\star\}$ are possible information values for the slot $\mathbf{area}$. As previously introduced, our rule-based belief tracking process is based on an information extraction from a machine utterance m and some consecutive user utterance u, the probabilities of the SLU hypotheses being handled afterwards. Let us denote this process by a function $\xi \in {\mathcal{I}_{\subset}}^{\mathcal{M}_{\subset} \times \mathcal{U}_{\subset}}$ such that $i \in \xi(m, u)$ is one instance of all the information extracted by ξ from m and u.

2.3 Preprocessing the machine acts and SLU hypotheses

In order to process the utterances of the user in the correct context, any occurrence of the REPEAT machine act is replaced by the machine act of the previous turn. In the formalization of the user acts, there is one ambiguity that must be solved: some acts contain a $\mathbf{this}$ in their slots. In the DSTC challenge, it can occur only for an INFORM act as a INFORM($\mathbf{this}$=$\star$). In Yarbus, the attempt to solve the reference of the slot $\mathbf{this}$ is based on the occurence of machine acts that explicitely require the user to mention a slot, namely the REQUEST, EXPL-CONF and SELECT acts. Therefore, the first step is to build up the set S_m of the slots associated with such acts in the machine utterance :

$$\begin{aligned} A &= \bigcup_{\{(a,args) \in m \mid a \in \{\texttt{"expl-conf"},\texttt{"select"}\}\}} \bigcup_{(s,v) \in args} \{s\} \\ B &= \bigcup_{\{(a,args) \in m \mid a = \texttt{"request"}\}} \bigcup_{(s,v) \in args} \{v\} \\ S_m &= A \cup B \end{aligned}$$

The set S_m can then be used to rewrite a single user act which can be formally defined by equation (6).

$$\rho \in {\mathcal{U}_{\subset}}^{\mathcal{U} \times \mathcal{S}_{\subset}},\ \rho((a, args), S_m) = \begin{cases} \emptyset & \text{if } (\texttt{"this"}, \star) \notin args \text{ or } |S_m| \neq 1 \\ \{(a, (s, \star))\} & \text{otherwise, with } S_m = \{s\} \end{cases} \tag{6}$$

As can be seen by the above definition, the result of rewriting a user act is a set with one or no element. The formal definitions of rewriting the SLU is actually easier is rewriting a single dialog act results as a set. The result is an empty set if there is no such $\mathbf{this}$=$\star$ in the slot value pairs of the user act or if there is more than one candidate for the reference. Processing a user utterance (a collection of acts), defined by equation (7), consists in building up the set of all acts that do not contain a $\mathbf{this}$=$\star$ slot-value pair and then complement it with the rewritten acts.

$$\varphi \in \mathcal{U}_{\subset}^{\mathcal{U}_{\subset} \times \mathcal{M}_{\subset}},\ \varphi(u,m) = \begin{array}{l} u \setminus \{(a,args) \in u \mid (\texttt{"this"}, \star) \in args\} \\ \cup \ \bigcup_{u' \in u} \rho\left(u', S_m\right) \end{array} \tag{7}$$

As dropping acts for the user utterance can result in creating duplicate hypotheses in the SLU, the final step consists in merging these duplicates and summing their probabilities :

$$\text{deref} \in \mathcal{H}^{\mathcal{H} \times \mathcal{M}_{\subset}},\ \text{deref}\,(h,m)\,(u) = \sum_{u' \in \{u'' \in \mathcal{U}_{\subset} \mid \varphi(u'',m)=u\}} h\left(u'\right) \tag{8}$$

The turns in the following table illustrate the SLU hypotheses when trying to solve the reference of `"this"`in two situations[5]. In the turn of the first dialog, the reference is solved with the slot **food** while it cannot be solved in the turn of the second dialog.

System act	**Original SLU**		**Rewritten SLU**	
	Hypothesis	Score	Hypothesis	Score
REQUEST (**slot**=$food$)	INFORM (**this**=$\star$)	0.99	INFORM (**food**=$\star$)	0.99
	AFFIRM () INFORM (**this**=$\star$)	0.01	AFFIRM () INFORM (**food**=$\star$)	0.01
OFFER (**name**=$goldenwok$) INFORM (**price**=$moderate$) INFORM (**area**=$north$)	INFORM (**this**=$\star$)	0.40	$\emptyset$	0.53
	$\emptyset$	0.13		
	REQALTS ()	0.14	REQALTS ()	0.14
	AFFIRM () INFORM (**this**=$\star$)	0.13	AFFIRM ()	0.20
	AFFIRM ()	0.07		
	ACK ()	0.06	ACK ()	0.06
	NEGATE () INFORM (**this**=$\star$)	0.03	NEGATE ()	0.05
	NEGATE ()	0.02		
	INFORM (**this**=$\star$) INFORM (**area**=$north$)	0.02	INFORM (**area**=$north$)	0.02
	THANKYOU ()	0.01	THANKYOU ()	0.01

2.4 Extracting informations from the SLU hypotheses

The rewritten hypotheses resulting from solving the `"this"` reference can now be processed to extract information. Every hypothesis is considered one after the other and a set of basic rules is applied on each. The information extracted from each hypothesis is represented as a tuple with a set of values for each slot. Informally, these rules build up a set for each slot s as :

1. If the hypothesis contains a INFORM (**s**=v), the information v is added to the set,
2. If the hypothesis contains a AFFIRM (), for every EXPL-CONF (**s**=v) in the machine utterance, the information v is added to the set,
3. If the hypothesis contains a DENY (**s**=v), the information $\neg v$ is added to the set,
4. If the hypothesis contains a NEGATE (), for every EXPL-CONF (**s**=v) in the machine utterance, the information $\neg v$ is added to the set,

5. The sentences come from the session-id *voip-db80a9e6df-20130328_234234* of the DSTC2 test set

5. If the hypothesis contains no NEGATE (), for every IMPL-CONF (s=v) in the machine utterance, the information v is added to the set.

Altogether, these rules capture three ideas. The first is that informed slot-value pairs must be captured wheter positively or negatively depending on the act that inform the slot/value pair. The second is that slot-value pairs that are asked by the machine to be explicitly confirmed must be considered only if the user is explicitly accepting or denying them (in which case the values are integrated positively or negatively) and the last rule integrates information implicitly confirmed by the machine only when the user does not negate them. As we shall see in the result section, this set of simple rules is sufficient to get reasonably good results on the challenges.

The five rules for extracting information from the SLU hypothesis can be formally defined by introducing the sets $\mathrm{pos}^s_{m,u}$ and $\mathrm{neg}^s_{m,u}$ in $\left(\texttt{string}_\star \cup \overline{\texttt{string}_\star}\right)_\subset$, $m \in \mathcal{M}$, $u \in \mathcal{U}$ and $s \in \mathcal{S}$ as :

$$\begin{aligned}
\mathrm{pos}^s_{m,u} = \quad & \{v \in \mathrm{val}\,(s)_\star \mid \exists (a, args) \in u,\ a = \texttt{"inform"} \text{ and } (s,v) \in args\} \\
\cup \quad & \left\{ v \in \mathrm{val}\,(s)_\star \;\middle|\; \begin{array}{ll} (\texttt{"affirm"}, \emptyset) \in u & \\ \text{and } \exists (a, args) \in m, & a = \texttt{"expl-conf"} \\ & \text{and } (s,v) \in args \end{array} \right\} \\
\cup \quad & \left\{ v \in \mathrm{val}\,(s)_\star \;\middle|\; \begin{array}{ll} (\texttt{"negate"}, \emptyset) \notin u & \\ \text{and } \exists (a, args) \in m, & a = \texttt{"impl-conf"} \\ & \text{and } (s,v) \in args \end{array} \right\}
\end{aligned} \tag{9}$$

$$\begin{aligned}
\mathrm{neg}^s_{m,u} = \quad & \left\{ \neg v \in \overline{\mathrm{val}\,(s)_\star} \;\middle|\; \exists (a, args) \in u,\ a = \texttt{"deny"} \text{ and } (s,v) \in args \right\} \\
\cup \quad & \left\{ \neg v \in \overline{\mathrm{val}\,(s)_\star} \;\middle|\; \begin{array}{ll} (\texttt{"negate"}, \emptyset) \in u & \\ \text{and } \exists (a, args) \in m, & a = \texttt{"expl-conf"} \\ & \text{and } (s,v) \in args \end{array} \right\}
\end{aligned} \tag{10}$$

The set $\mathrm{pos}^s_{m,u}$ (resp. $\mathrm{neg}^s_{m,u}$) contains the positive (resp. negative) information that can be extracted from the machine utterance and a single user utterance. These two sets are then merged and cleaned. Let us take an example to motivate the cleaning process. Suppose that the user has negated a slot-value pair that the machine requests to explicitly confirm ($\mathrm{neg}^{\mathbf{food}}_{m,u} = \{\neg french\}$) and in the same utterance informs s/he has informed that s/he wants a british restaurant ($\mathrm{pos}^{\mathbf{food}}_{m,u} = \{british\}$), then the information about the british food is more informative than the $\neg french$ information for uncovering the user's goal. The second motivation comes from possible conflicts. Suppose the machine utterance is EXPL-CONF (**food**=$british$) and that the SLU recognized the utterance AFFIRM ()INFORM (**food**=$french$). In that case, there is clearly a conflict and there is no *a priori* reason to favor **food**=$british$ over **food**=$french$. In Yarbus, the two extracted positives therefore receive a uniform split of the mass given of the SLU hypothesis from which they are extracted. The step of splitting the mass of an hypothesis over the information extracted from it is made explicit in the next section on the update function. Formally, merging the sets of positives and negatives can be defined as building the set of sets $\mathrm{inf}^s_{m,u}$ as :

$$\mathrm{inf}^s_{m,u} = \begin{cases} \{\texttt{?}\} & \text{if } \left(\mathrm{pos}^s_{m,u}, \mathrm{neg}^s_{m,u}\right) = (\emptyset, \emptyset) \text{ or } u = \emptyset \\ \left\{\mathrm{neg}^s_{m,u}\right\} & \text{if } \mathrm{pos}^s_{m,u} = \emptyset \\ \left\{\{v\} \;\middle|\; v \in \mathrm{pos}^s_{m,u}\right\} & \\ \quad \cup \left\{\left\{\neg v \in \mathrm{neg}^s_{m,u} \;\middle|\; v \in \mathrm{pos}^s_{m,u}\right\}\right\} & \text{otherwise} \end{cases} \tag{11}$$

If there is no positive nor negative or if the utterance of the user is empty[6], the value for the slot is simply unknown. In case there are no positives, all the negatives are kept. In case there are both positives and negatives, all the positives are kept as singletons as well the negatives that conflict with the positives. An example[7] of information extraction and fusion is given in the following:

$$\begin{aligned} m &= \text{EXPL-CONF}\left(\mathbf{food}{=}vietnamese\right) \\ u &= \{\text{NEGATE}\left(\right), \text{INFORM}\left(\mathbf{this}{=}\star\right), \text{INFORM}\left(\mathbf{food}{=}romanian\right)\} \\ \text{pos}_{m,u}^{\mathbf{food}} &= \{\star, romanian\} \\ \text{neg}_{m,u}^{\mathbf{food}} &= \{\neg vietnamese\} \\ \text{inf}_{m,u}^{\mathbf{food}} &= \{\{\star\}, \{romanian\}\} \end{aligned} \tag{12}$$

where both positives and negatives are involved. The positives are retained and the negative is therefore discarded in the fusion.

As Yarbus focuses on joint goals, the cartesian product of the information extracted for each slot is computed and leads to the set of information for all the slots extracted from a single machine utterance and user utterance $\xi\left(m,u\right)$. This can be formally defined as :

$$\xi\left(m,u\right) = \left\{ i \in \mathcal{I} \;\middle|\; \forall s, i\left(s\right) \in \text{inf}_{m,u}^{s} \right\} \tag{13}$$

Let us consider for example the machine utterance EXPL-CONF $\left(\mathbf{pricerange}{=}cheap\right)$. The SLU hypothesis $h \in \mathcal{H}$ as well as their associated information set $\xi\left(m,u\right), u \in h$ are shown in the following, where the tabular definition[8] of function $[x_1 \rightarrow y_1, x_2 \rightarrow y_2, \bullet \rightarrow y_3]$ stands for the function returning y_1 for x_1, y_2 for x_2 and y_3 otherwise. Let us consider

$$\begin{aligned} m &= \{\text{EXPL-CONF}\left(\mathbf{pricerange}{=}cheap\right)\} \\ h &= \begin{bmatrix} \{\text{INFORM}\left(\mathbf{pricerange}{=}\star\right)\} \rightarrow & 0.87 \\ \{\text{AFFIRM}\left(\right), \text{INFORM}\left(\mathbf{pricerange}{=}\star\right)\} \rightarrow & 0.10 \\ \{\text{NEGATE}\left(\right), \text{INFORM}\left(\mathbf{pricerange}{=}\star\right)\} \rightarrow & 0.03 \\ \bullet \rightarrow & 0 \end{bmatrix} \end{aligned}$$

For each hypothesis in h with a non null probability, the extracted informations are

$$\begin{aligned} \xi\left(m, \{\text{INFORM}\left(\mathbf{pricerange}{=}\star\right)\}\right) &= \{[\mathbf{pricerange} \rightarrow \{\star\}, \bullet \rightarrow \{?\}]\} \\ \xi\left(m, \{\text{AFFIRM}\left(\right), \text{INFORM}\left(\mathbf{pricerange}{=}\star\right)\}\right) &= \left\{ \begin{array}{l} [\mathbf{pricerange} \rightarrow \{\star\}, \bullet \rightarrow \{?\}], \\ [\mathbf{pricerange} \rightarrow \{cheap\}, \bullet \rightarrow \{?\}] \end{array} \right\} \\ \xi\left(m, \{\text{NEGATE}\left(\right), \text{INFORM}\left(\mathbf{pricerange}{=}\star\right)\}\right) &= \{[\mathbf{pricerange} \rightarrow \{\star\}, \bullet \rightarrow \{?\}]\} \end{aligned}$$

2.5 Updating the belief from the extracted informations

The goal of a belief tracker is to update a probability distribution over $\mathcal{G}$, i.e to update $b_t \in \mathcal{B}$ at each successive turn t. Before the first turn, we assume no *a priori* on the goal of the user and the

6. this rule prevents the inclusion of information extracted based on the absence of acts in the user utterance such as the third rule of equation (9). Indeed, it is more conservative to suppose that an empty utterance might contain the act we are supposing is missing.
7. the example is the second turn of *voip-5cf59cc660-20130327_143457* in dstc2_test
8. this definition is introduced for the sake of clarity of the examples.

belief b_0 is therefore initialized as :

$$b_0(g) = \begin{cases} 1 & \text{if } g \text{ is the constant function } g(s) = ? \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

The belief update, denoted by τ, relies on an elementary transition function $\mu \in \mathcal{G}^{\mathcal{G}\times\mathcal{I}}$ defined by equation (15).

$$\mu(g,i)(s) = \begin{cases} v & \text{if} \quad g(s) = v \in \text{val}(s)_{\star ?} \quad \text{and} \quad i(s) = \{?\} \\ v & \text{if} \quad g(s) = ? \quad \text{and} \quad i(s) = \{v\},\ v \in \text{val}(s)_{\star} \\ ? & \text{if} \quad g(s) = ? \quad \text{and} \quad i(s) \in \left(\overline{\text{val}(s)_{\star}}\right)_{\subset} \\ v' & \text{if} \quad g(s) = v \in \text{val}(s)_{\star} \quad \text{and} \quad i(s) = \{v'\},\ v' \in \text{val}(s)_{\star} \\ ? & \text{if} \quad g(s) = v \in \text{val}(s)_{\star} \quad \text{and} \quad \neg v \in i(s) \\ v & \text{if} \quad g(s) = v \in \text{val}(s)_{\star} \quad \text{and} \quad i(s) \in \left(\overline{\text{val}(s)_{\star}} \setminus \{\neg v\}\right)_{\subset} \end{cases} \tag{15}$$

For each slot s, the transition function states that the goal v remains the same if the information is unknown, v or a negation of a value different from v. The goal changes to unknown in case the information negates it. And finally, if the information is a positive different from the current goal, the goal switches to this positive.

Given the transition function μ, the belief can be updated by introducing the belief update function $\tau \in \mathcal{B}^{\mathcal{B}\times\mathcal{M}_{\subset}\times\mathcal{H}}$ according to equation (17).

$$P_{m_t,u}^{g'\to g} = \frac{1}{|\xi(m_t,u)|} \sum_{i\in\xi(m_t,u)} \mathbb{1}_{\{g\}}\left(\mu\left(g',i\right)\right) \tag{16}$$

$$b_t = \tau(b_{t-1}, m_t, h_t)(g) = \sum_{g'\in\mathcal{G}} b_{t-1}\left(g'\right) \sum_{u\in\mathcal{U}_{\subset}} h_t(u)\, P_{m_t,u}^{g'\to g} \tag{17}$$

where $P_{m_t,u}^{g'\to g}$ shares equally the probabilities between all the information extracted from m_t and u, and retains only, once the share is affected, the information generating a transition from g' to g.

3. Results

3.1 Running the tracker on the noise-free SLU from the labels

The datasets of the challenges have been labeled using Amazon Mechanical Turk. These labels contain the joint goals that the belief tracker has to identify and also the *semantics*, i.e. what the labelers understood from the audio recordings of the dialogs and written in the dialog acts formalism. Therefore, one can make use of this semantics to test the belief tracker in a noise free SLU condition. In theory, a good belief tracker should have the best scores on the metrics in this ideal condition. It turns out that Yarbus does perform almost perfectly according to the metrics in this ideal condition by performing the following number of mistakes :

- 5 mistakes on the joint goals for dstc2_train
- 1 mistake on the joint goals for dstc2_dev

- 183 mistakes on the joint goals for dstc2_test
- 0 mistake on the joint goals for dstc3_test

All the mistakes are actually produced by the IMPL-CONF () rule and discarding this rule leads to 100% accuracy. Indeed, there are some slot/value pairs that get integrated in the belief by Yarbus because they have been implicitly confirmed by the machine and not denied by the user. This rule is actually not beneficial when the SLU is un-noisy. Indeed, it generates information that is not explicitly given by the user but generated by the machine based on its belief. As we shall see in the next sections, with the outputs from the SLUs, there is a slight improvement in performances if we consider this rule. Actually, we go back on this issue by discussing the performances with respect to the set of rules being considered in section 3.4.

Live SLU								
dstc2_train			dstc2_dev			dstc2_test		
Accuracy	L2	ROC	Accuracy	L2	ROC	Accuracy	L2	ROC
0.719	0.464	0	0.630	0.602	0	0.725	0.440	0
SJTU 1best SLU								
dstc2_train			dstc2_dev			dstc2_test		
Accuracy	L2	ROC	Accuracy	L2	ROC	Accuracy	L2	ROC
0.835	0.265	0.232	0.801	0.330	0.254	0.752	0.392	0.271
SJTU 1best+sys SLU								
dstc2_train			dstc2_dev			dstc2_test		
Accuracy	L2	ROC	Accuracy	L2	ROC	Accuracy	L2	ROC
0.871	0.213	0.281	0.841	0.257	0.208	0.759	0.358	0.329

Table 1: Results of Yarbus on the DSTC-2 datasets for the three SLU (live, and the two from SJTU(Sun et al., 2014b)). For each dataset, the reported scores are the featured metrics of the challenges, namely: "accuracy", "L2 norm" and "ROC performance Correct Accept 5%".

Live SLU			SJTU asr-tied SLU		
Acc.	L2	ROC ca5	Acc.	L2	ROC ca5
0.582	0.702	0	0.597	0.624	0.226
SJTU errgen SLU			SJTU errgen+rescore SLU		
Acc.	L2	ROC ca5	Acc.	L2	ROC ca5
0.594	0.624	0.150	0.595	0.607	0.151

Table 2: Results of Yarbus on the DSTC-3 datasets with four SLU (live, and the three SLU from (Zhu et al., 2014)). For each dataset, the reported scores are the featured metrics of the challenges, namely: "accuracy", "L2 norm" and "ROC performance Correct Accept 5%".

3.2 Performances on the challenges and comparison to the previous submissions

The featured metrics of the challenges of YARBUS on the DSTC2 and DSTC3 datasets are shown in Table 1 and 2. These are reported with the previous submissions to the challenges on the figures 1 and 3. The baselines (Top-hyp, Focus, HWU and HWU original) were run on the same SLUs than Yarbus and the complete set of results for these trackers is reported in Appendix A. It turns out that with its few rules and the SLU of Sun et al. (2014b), Yarbus ranks reasonably well compared to the other approaches in the DSTC2 challenge and always better than the other baselines (top-hyp, focus, HWU and HWU+) with one exception for the ROC metric compared to HWU. For the DSTC3 challenge, Yarbus is best performing almost on the three metrics by using the error-tied SLU from (Zhu et al., 2014). It usually performs better than the other baselines.

Interestingly, the baselines (not only Yarbus) perform much better than the other approaches when we compare the trackers on the dstc2_dev dataset and running the baselines on the SLU of (Sun et al., 2014b). The metrics of the trackers on the dstc2_dev dataset are reported on Fig. 2.

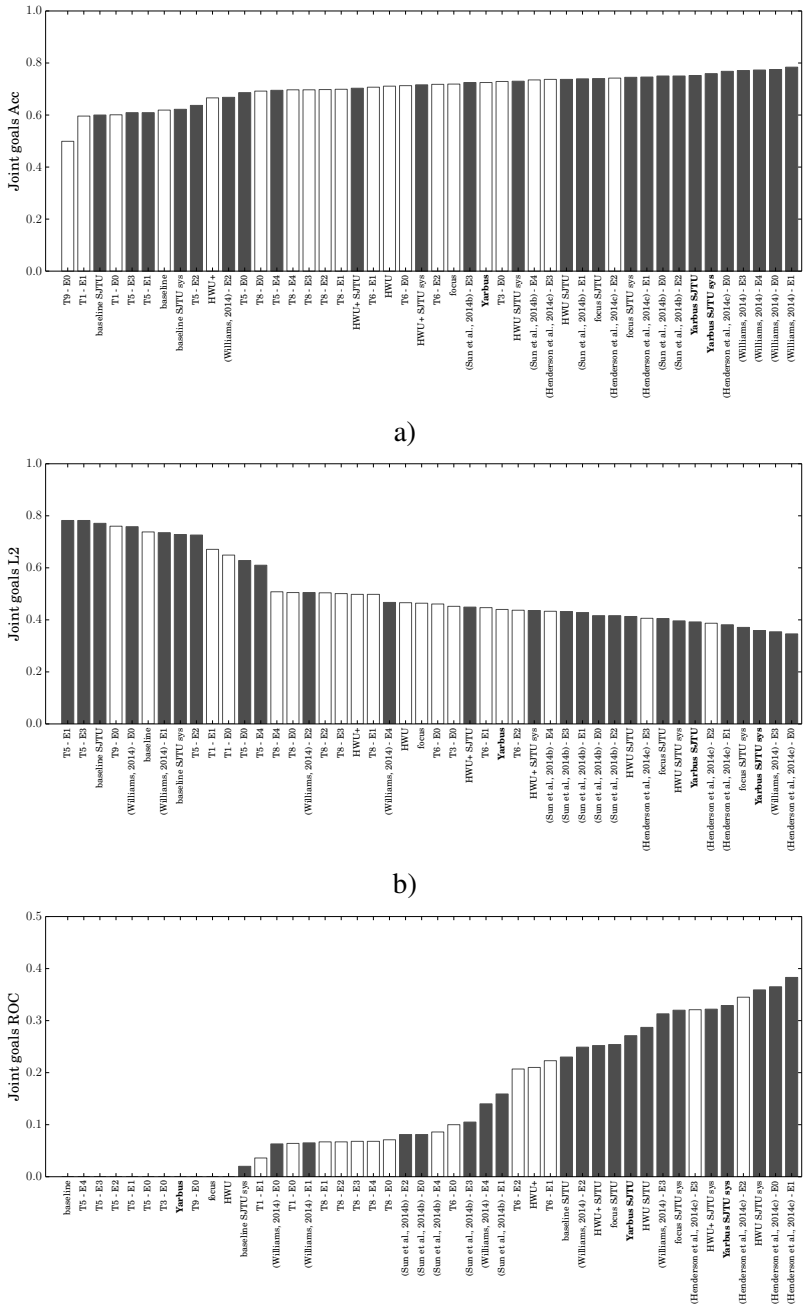
The results of the trackers on the dstc2_train dataset are not available except for the baselines for which the results on the three datasets of DSTC2 are provided in Appendix A. There is clear tendency for Yarbus to perform better than the other baselines on Accuracy and L2 norm but not for the ROC for which the HWU baseline is clearly better.

On the third challenge (fig. 3), the difference in the results of the various trackers is much less clear than in the second challenge, at least in terms of accuracy. Yarbus is slightly better than the other baselines. The best ranking approaches are the recurrent neural network approach of (Henderson et al., 2014a) and the polynomial belief tracker of (Zhu et al., 2014).

3.3 The size of the tracker

As noted in introduction the number of possible joint goals is much larger in the third challenge than in the second. Indeed, in this case, there are around 10^9 possible joint goals. Therefore, when estimating the belief in the joint space, it might be that the representation becomes critically large. Such a situation is much less critical when the belief is defined in the marginal space, as in (Wang and Lemon, 2013) where the space to be represented is the sum and not the product of the number of values of each slot. In Yarbus, after being updated, the belief is pruned by removing all the joint goals having a probability lower than a given threshold θ_b and scaling after-while the remaining probabilities so that they sum to one. In case all the joint goals have a probability lower than θ_b, the pruning is not applied as it would result in removing all the elements of the belief. If the pruning is not applied, the size of the belief might be especially large when the SLU is producing a lot of hypothesis. For example, if we measure the size of the belief on the DSTC2 challenge with the SJTU+sys SLU, the belief can contain up to 700 entries (fig.4a) and to more than 10000 for the DSTC3 challenge with the SJTU err-tied SLU (fig.4b). If the pruning is applied with $\theta_b = 10^{-2}$, the number of entries in the belief does not exceed around 30 while still keeping pretty much the same performances than the un-pruned belief. For the DSTC2 with the SJTU+sys SLU, the performances of the un-pruned belief are (Acc:0.759, L2:0.359, ROC:0.329) and the performances of the pruned belief with $\theta_b = 10^{-2}$ are (Acc:0.759, L2:0.361, ROC:0.320). For the DSTC3 with the SJTU err-tied SLU, the performances of the un-pruned belief[9] are (Acc:0.597, L2:0.615, ROC:0.239) and the performances of the pruned belief with $\theta_b = 10^{-2}$ are (Acc:0.597, L2:0.624, ROC:0.226).

9. The reported performances are actually for $\theta_b = 10^{-10}$ because setting $\theta_b = 0$ produced a tracker output much too large to be evaluated by the scoring scripts of the challenge.



a)

b)

c)

Figure 1: Performances of the trackers on the dstc2_test dataset. The reported measures are the features metrics of the challenge: a) Accuracy, b) L2 norm, c) ROC CA5%. The trackers using the live ASR are represented with black bars and the trackers not using the live ASR (i.e. only the live SLU) in white. The y-ranges are adjusted to better appreciate the differences of the scores.

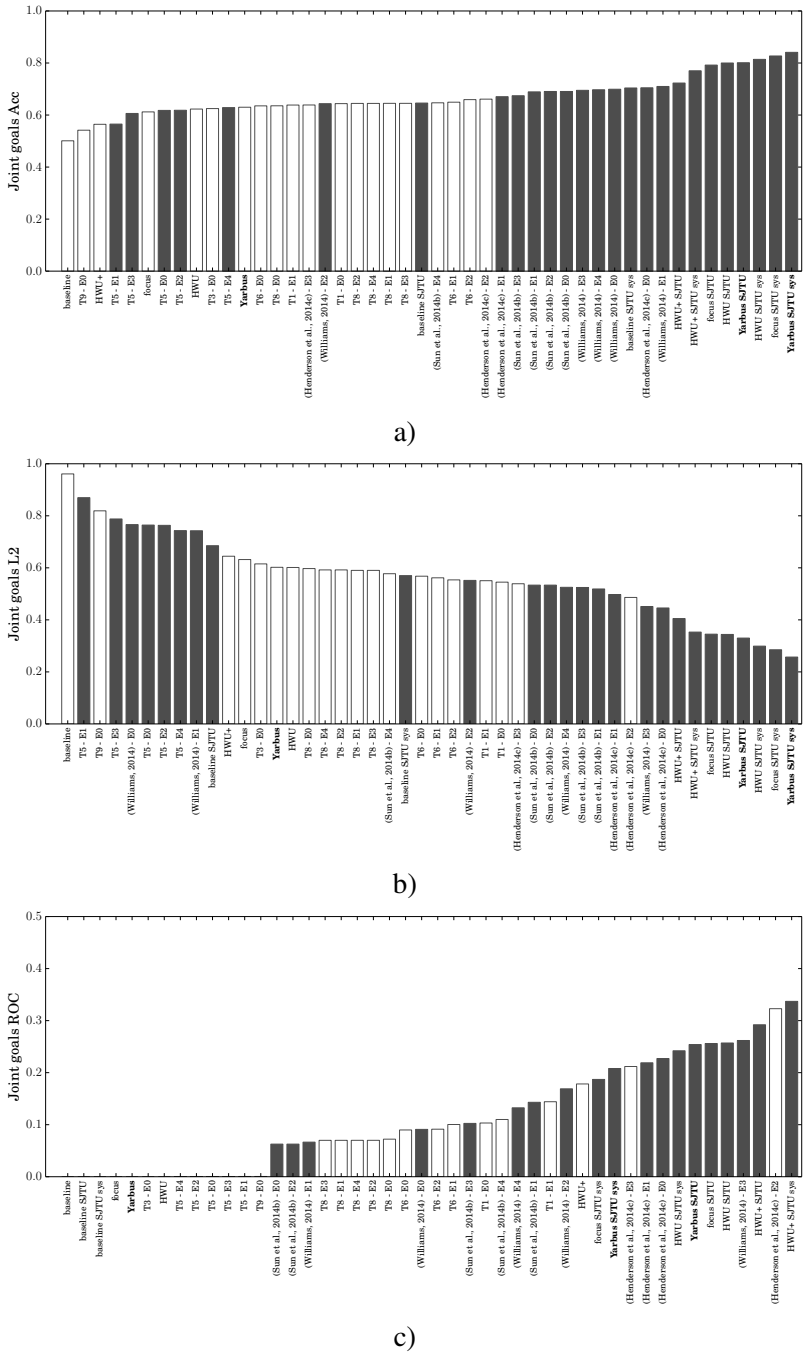


Figure 2: Performances of the trackers on the dstc2_dev dataset. The reported measures are the features metrics of the challenge: a) Accuracy, b) L2 norm, c) ROC CA5%. The trackers using the live ASR are represented with black bars and the trackers not using the live ASR (i.e. only the live SLU) in white. The y-ranges are adjusted to better appreciate the differences of the scores.

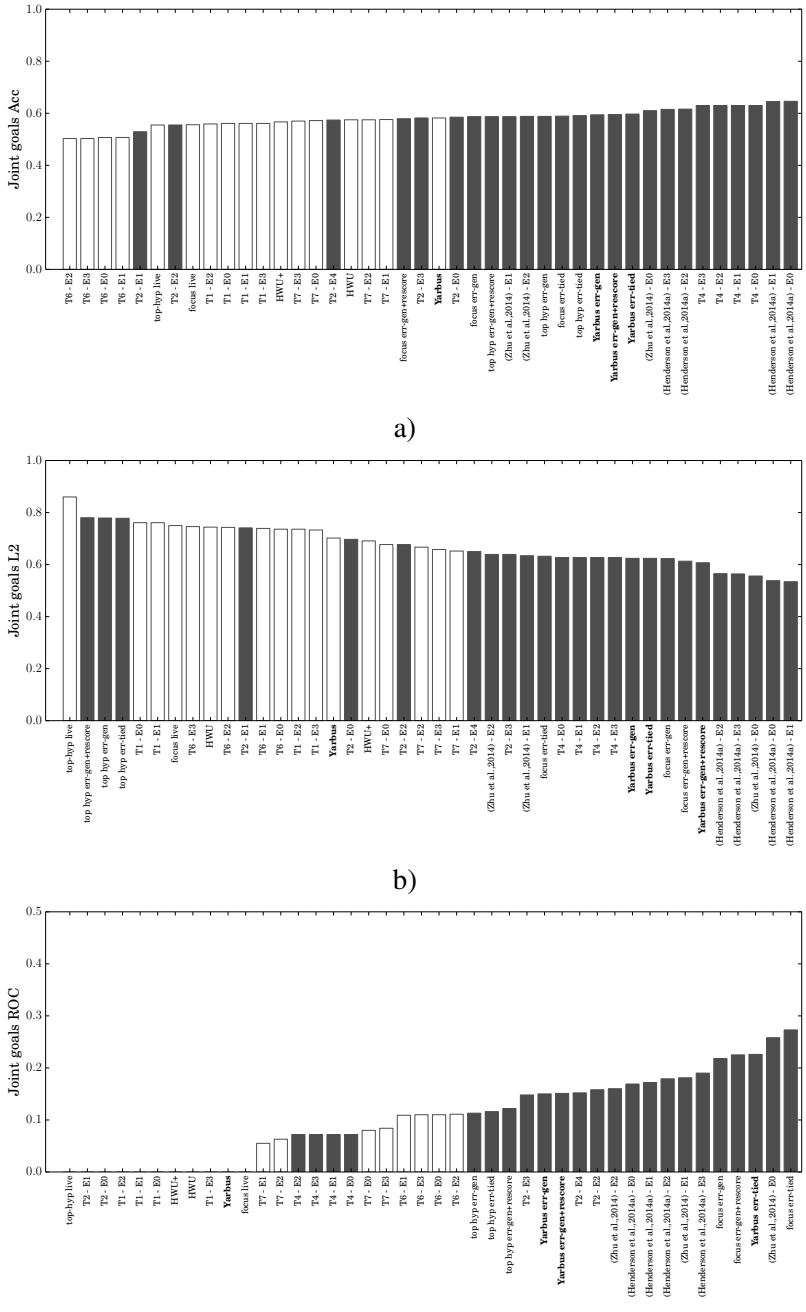
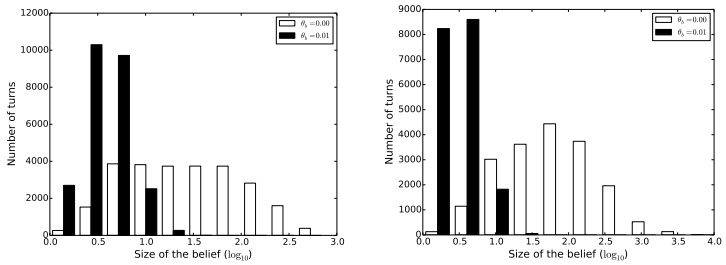


Figure 3: Performances of the trackers on the dstc3_test dataset. The reported measures are the features metrics of the challenge: a) Accuracy, b) L2 norm, c) ROC CA5%. The trackers using the live ASR are represented with black bars and the trackers not using the live ASR (i.e. only the live SLU) in white. The y-ranges are adjusted to better appreciate the differences of the scores.



a) b)

Figure 4: a) On the DSTC2 data (all the subsets included), without filtering the belief after its update, its size can grow up to 700 joint goals. By removing the joint goals with a probability lower than $\theta_b = 10^{-2}$ and scaling the resulting belief accordingly, its size gets no more than around 30. The results are measured on the SJTU sys SLU. b) On the dstc3_test dataset, without filtering, the belief can have up to 10000 entries. Filtering the belief with $\theta_b = 1e-2$ significantly decreases the number of elements (no more than 30). The experiment has been run on the SJTU err-tied SLU.

3.4 Varying the set of rules

The motivation behind the rules in Yarbus is to use a reasonably small number of rules which can hopefully extract most of the information from the machine and user acts. This choice is not driven by any dataset *per se* in the sense that it might be that a smaller set of rules, which might not capture all the information from the defined acts, still performs reasonably well. That point can be checked by simply enabling or disabling rules and checking the metrics of the resulting modified Yarbus. The experimental setup is the following. Let us attribute a rule number to the five rules presented in section 2.4 as :

- Rule 0 : the INFORM () rule in equation (9)
- Rule 1 : the EXPL-CONF () rule in equation (9)
- Rule 2 : the IMPL-CONF () rule in equation (9)
- Rule 3 : the NEGATE () rule in equation (10)
- Rule 4 : the DENY () rule in equation (10)

We can now define variations of Yarbus denoted Yarbus-$r_0r_1r_2r_3r_4$ where the sequence $r_0r_1r_2r_3r_4$ identifies which rules are enabled or disabled. The tracker considered so far is therefore denoted Yarbus-11111. Given the 5 rules defined above there are 32 possibile combinations which can all be tested on the challenge datasets. In the experiment, we make use only of the SJTU+sys SLU for the DSTC2 challenge and SJTU+err-tied SLU for the DSTC3 challenge. The full set of results are given in Appendix B (Table 5 and 6). The metrics of the various trackers on the different datasets are plotted on Fig. 5. It turns out that a big step in the metrics is obtained when enabling the Rule 0, i.e. the inform rule which is not much a surprise. However it might be noted that the performances do not grow much by adding additional rules. There is one exception for the dstc2_test dataset for

which the inclusion of the second rule leads to an increase in 5% in accuracy. The conclusion is clearly that even if the additional rules make sense in the information they capture, they do not lead to significant improvements on the metrics and the performances can be obtained by making use of only two rules : the inform and expl-conf rules of equation (9).

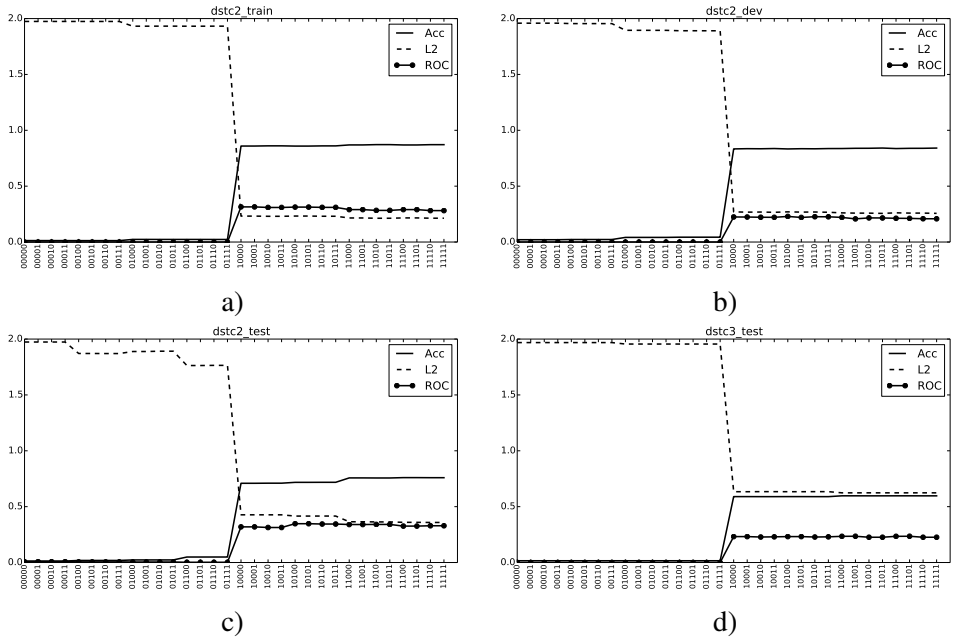


Figure 5: Metrics of YARBUS with different rule sets on a) dstc2_train, b) dstc2_dev, c) dstc2_test and d) dstc3_test.

4. Discussion

In this paper, we presented a rule-based belief tracker. The tracker, which does not require any learning, performs favourably on the dialog state tracking challenges in comparison with other approaches. It should be noted that there is a significant increase in performances by switching from the live SLU of the challenges to the SLU of (Sun et al., 2014b; Zhu et al., 2014). Yarbus is a very simple tracker. We tried actually to add rules that appeared at first glance to capture more information or to capture the information in a more coherent way (for example by considering alternatives in the way the reference of **this** is solved) but these attempts resulted in degraded performances. Yarbus is in no way a very tricky belief tracker. Most of the *expertise* comes from the design of the rules extracting information from the utterances but otherwise the update of the belief is based on simple Bayesian rules. In light of the results of section 3.4, it turns out that Yarbus could be even simpler by considering only two out of the five rules. However, the point of the paper was clearly not to devise a new belief tracker but the fact that Yarbus uses rules involving the dialog acts is actually quite informative. First, despite its simplicity, if one compares the performances of Yarbus with the best ranking tracker proposed by (Williams, 2014), there are dialogs on which

the first performs better than the second and dialogs on which the second performs better than the first. In that respect, Yarbus might still be a good candidate for ensemble learning (Henderson et al., 2014b). Second, using such a simple rule based tracker informs us on the real performances of more elaborate machine learning based techniques such as recurrent neural networks (Henderson et al., 2014c) or deep neural networks (Henderson et al., 2013). These latter techniques are rather blind to the data being processed. Even if these approaches performs well at first sight, the performances of Yarbus allow to better appreciate what part of the information is really extracted from the data. Last, one natural conclusion from the results of this paper is that there is still work to be done in order to get real breakthroughs in slot filling tasks. Since a simple rule based system performs very well (more than 75% of accuracy) on the second challenge is raising the question of making use of this dataset for evaluating belief trackers. On the third challenge, the conclusion is less straightforward. However, it is clear from the datasets of DSTC2 and DSTC3 that the biggest improvements were achieved thanks to the SLU and this suggests to shift the focus on this element of the dialog loop.

Acknowledgments: This work has been supported by l'Agence Nationale pour la Recherche under projet reference ANR-12-CORD-0021 (MaRDi).

References

Alan W Black, Susanne Burger, Alistair Conkie, Helen Hastie, Simon Keizer, Oliver Lemon, Nicolas Merigaud, Gabriel Parent, Gabriel Schubiner, Blaise Thomson, Jason D. Williams, Kai Yu, Steve Young, and Maxine Eskenazi. Spoken dialog challenge 2010: Comparison of live and control test results. In *Proc SIGDIAL, Portland, Oregon, USA*, 2011. URL `http://research.microsoft.com/apps/pubs/default.aspx?id=160915`.

Jeremy Fix and Hervé Frezza-Buet. Yarbus source code, 2015. URL `https://github.com/jeremyfix/dstc/`.

D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai. A form-based dialogue manager for spoken language applications. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 2, pages 701–704 vol.2, Oct 1996. doi: 10.1109/ICSLP.1996.607458. URL `http://dx.doi.org/10.1109/ICSLP.1996.607458`.

M. Henderson, B. Thomson, and S. J. Young. Deep Neural Network Approach for the Dialog State Tracking Challenge. In *Proceedings of SIGdial*, 2013.

M. Henderson, B. Thomson, and S. J. Young. Robust Dialog State Tracking Using Delexicalised Recurrent Neural Networks and Unsupervised Adaptation. In *Proceedings of IEEE Spoken Language Technology*, 2014a. URL `http://mi.eng.cam.ac.uk/~sjy/papers/htyo14.pdf`.

Matthew Henderson, Blaise Thomson, and Jason Williams. Dialog state tracking challenge 2 & 3, 2013–2014. URL `http://camdial.org/~mh521/dstc/`.

Matthew Henderson, Blaise Thomson, and Jason D Williams. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A., June 2014b. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W14-4337`.

Matthew Henderson, Blaise Thomson, and Steve Young. 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)*, pages 292–299, Philadelphia, PA, U.S.A., June 2014c. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14-4340.

Verena Rieser and Oliver Lemon. *Reinforcement Learning for Adaptive Dialogue Systems: A Data-driven Methodology for Dialogue Management and Natural Language Generation*. Springer-Verlag, 2011.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. A generalized rule based tracker for dialogue state tracking. In *Proceedings 2014 IEEE Spoken Language Technology Workshop*, South Lake Tahoe, USA, December 2014a. URL http://www.aiexp.info/files/slt_1056.pdf.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. The sjtu system for dialog state tracking challenge 2. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 318–326, Philadelphia, PA, U.S.A., June 2014b. Association for Computational Linguistics. URL http://www.sigdial.org/workshops/conference15/proceedings/pdf/W14-4343.pdf.

Zhuoran Wang and Oliver Lemon. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432, Metz, France, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W13/W13-4067.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, Metz, France, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W13/W13-4065.

Jason D Williams. Web-style ranking and slu combination for dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 282–291, Philadelphia, PA, U.S.A., June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14-4339.

Su Zhu, Lu Chen, Kai Sun, Da Zheng, and Kai Yu. Semantic parser enhancement for dialogue domain extension with little data. In *Proceedings 2014 IEEE Spoken Language Technology Workshop*, South Lake Tahoe, USA, 2014. URL http://www.aiexp.info/files/slt_1062.pdf.

Appendix A : Scores of the different baselines with the various SLU

	Live SLU								
	dstc2_train			dstc2_dev			dstc2_test		
	Acc.	L2	ROC ca5	Acc.	L2	ROC ca5	Acc.	L2	ROC ca5
top-hyp	0.582	0.810	0	0.501	0.961	0	0.619	0.738	0
focus	0.715	0.471	0	0.612	0.632	0	0.719	0.464	0
HWU	0.732	0.451	0	0.623	0.601	0	0.711	0.466	0
HWU+	0.646	0.518	0.185	0.564	0.645	0.178	0.666	0.498	0.210
Yarbus	0.719	0.464	0	0.630	0.602	0	0.725	0.440	0
	SJTU 1best SLU								
	dstc2_train			dstc2_dev			dstc2_test		
	Acc.	L2	ROC ca5	Acc.	L2	ROC ca5	Acc.	L2	ROC ca5
top-hyp	0.656	0.669	0.032	0.646	0.685	0	0.600	0.771	0.23
focus	0.831	0.278	0.220	0.792	0.345	0.256	0.740	0.405	0.254
HWU	0.841	0.266	0.206	0.800	0.344	0.257	0.737	0.413	0.287
HWU+	0.755	0.337	0.318	0.723	0.405	0.292	0.703	0.449	0.252
Yarbus	0.835	0.265	0.232	0.801	0.330	0.254	0.752	0.392	0.271
	SJTU 1best+sys SLU								
	dstc2_train			dstc2_dev			dstc2_test		
	Acc.	L2	ROC ca5	Acc.	L2	ROC ca5	Acc.	L2	ROC ca5
top-hyp	0.722	0.512	0.038	0.704	0.570	0	0.622	0.728	0.020
focus	0.862	0.231	0.273	0.827	0.285	0.187	0.745	0.371	0.320
HWU	0.859	0.231	0.299	0.814	0.299	0.242	0.730	0.396	0.359
HWU+	0.803	0.300	0.380	0.770	0.353	0.337	0.716	0.436	0.322
Yarbus	0.871	0.213	0.281	0.841	0.257	0.208	0.759	0.359	0.329

Table 3: Results for the joint goals on the DSCT2 challenge. HWU denotes the Heriot-Watt tracker(Wang and Lemon, 2013), HWU+ is with the original flag enabled.

	Live SLU			SJTU asr-tied SLU		
	Acc.	L2	ROC ca5	Acc.	L2	ROC ca5
top-hyp	0.555	0.860	0	0.591	0.778	0.116
focus	0.556	0.750	0	0.589	0.632	0.274
HWU	0.575	0.744	0	-	-	-
HWU+	0.567	0.691	0	-	-	-
Yarbus $\theta_b = 10^{-2}$	0.582	0.702	0	0.597	0.624	0.226
	SJTU errgen SLU			SJTU errgen+rescore SLU		
	Acc.	L2	ROC ca5	Acc.	L2	ROC ca5
top-hyp	0.588	0.779	0.114	0.587	0.780	0.122
focus	0.587	0.623	0.218	0.579	0.613	0.225
HWU	-	-	-	-	-	-
HWU+	-	-	-	-	-	-
Yarbus $\theta_b = 10^{-2}$	0.594	0.624	0.150	0.595	0.607	0.151

Table 4: Results for the joint goals on the DSCT3 challenge. For HWU and HWU+, using the SJTU SLUs leads to a tracker output too large to hold in memory and the results are therefore not available on these datasets.

Appendix B: Metrics of YARBUS with various rule sets

Rule set	Accuracy	L2	ROC ca5
00000	0.0128014	1.9743972	0.0000000
00001	0.0128014	1.9743972	0.0000000
00010	0.0128014	1.9743972	0.0000000
00011	0.0128014	1.9743972	0.0000000
00100	0.0125384	1.9742266	0.0000000
00101	0.0125384	1.9742266	0.0000000
00110	0.0125384	1.9742266	0.0000000
00111	0.0125384	1.9742266	0.0000000
01000	0.0233231	1.9318885	0.0000000
01001	0.0233231	1.9318885	0.0000000
01010	0.0230601	1.9323025	0.0000000
01011	0.0230601	1.9323025	0.0000000
01100	0.0230601	1.9317179	0.0000000
01101	0.0230601	1.9317179	0.0000000
01110	0.0227970	1.9321319	0.0000000
01111	0.0227970	1.9321319	0.0000000
10000	0.8590969	0.2320063	0.3144519
10001	0.8590969	0.2320063	0.3144519
10010	0.8608505	0.2303032	0.3093298
10011	0.8608505	0.2303032	0.3093298
10100	0.8587462	0.2324565	0.3126404
10101	0.8587462	0.2324565	0.3126404
10110	0.8604998	0.2307534	0.3101691
10111	0.8604998	0.2307534	0.3101691
11000	0.8690925	0.2153537	0.2902542
11001	0.8690925	0.2153537	0.2902542
11010	0.8718106	0.2127873	0.2831137
11011	0.8718106	0.2127873	0.2831137
11100	0.8687418	0.2158038	0.2902705
11101	0.8687418	0.2158038	0.2902705
11110	0.8714599	0.2132374	0.2813160
11111	0.8714599	0.2132374	0.2813160

a)

Rule set	Accuracy	L2	ROC ca5
00000	0.0205944	1.9588113	0.0000000
00001	0.0205944	1.9588113	0.0000000
00010	0.0205944	1.9588113	0.0000000
00011	0.0205944	1.9588113	0.0000000
00100	0.0224192	1.9549719	0.0000000
00101	0.0224192	1.9549719	0.0000000
00110	0.0224192	1.9549719	0.0000000
00111	0.0224192	1.9549719	0.0000000
01000	0.0414494	1.8949548	0.0000000
01001	0.0414494	1.8949548	0.0000000
01010	0.0414494	1.8946709	0.0000000
01011	0.0414494	1.8946709	0.0000000
01100	0.0432742	1.8911155	0.0000000
01101	0.0432742	1.8911155	0.0000000
01110	0.0432742	1.8908316	0.0000000
01111	0.0432742	1.8908316	0.0000000
10000	0.8334202	0.2697380	0.2242728
10001	0.8355057	0.2673630	0.2237129
10010	0.8347237	0.2683460	0.2217364
10011	0.8368092	0.2659710	0.2211838
10100	0.8331595	0.2701215	0.2290363
10101	0.8352450	0.2677466	0.2200375
10110	0.8344630	0.2687295	0.2264917
10111	0.8365485	0.2663545	0.2259271
11000	0.8368092	0.2609925	0.2202492
11001	0.8388947	0.2586176	0.2072716
11010	0.8391554	0.2585226	0.2159056
11011	0.8412409	0.2561476	0.2153703
11100	0.8365485	0.2613761	0.2125273
11101	0.8386340	0.2590011	0.2116879
11110	0.8388947	0.2589061	0.2085146
11111	0.8409802	0.2565311	0.2079975

b)

Table 5: Metrics of YARBUS with various rule sets on a) the dstc2_train and b) the dstc2_dev datasets. The meaning of the rule set number is defined in section 3.4. The trackers were run on the SJTU + sys SLU.

Rule set	Accuracy	L2	ROC ca5
00000	0.0132109	1.9735783	0.0078125
00001	0.0132109	1.9735783	0.0078125
00010	0.0132109	1.9735783	0.0078125
00011	0.0132109	1.9735783	0.0078125
00100	0.0184746	1.8709733	0.0055866
00101	0.0184746	1.8709733	0.0055866
00110	0.0184746	1.8703227	0.0055866
00111	0.0184746	1.8703227	0.0055866
01000	0.0230158	1.8886356	0.0044843
01001	0.0230158	1.8886356	0.0044843
01010	0.0231190	1.8913796	0.0044643
01011	0.0231190	1.8913796	0.0044643
01100	0.0491279	1.7631903	0.0021008
01101	0.0491279	1.7631903	0.0021008
01110	0.0492311	1.7644893	0.0020964
01111	0.0492311	1.7644893	0.0020964
10000	0.7091547	0.4269933	0.3190220
10001	0.7091547	0.4269933	0.3190220
10010	0.7100836	0.4268610	0.3133721
10011	0.7100836	0.4268610	0.3133721
10100	0.7175147	0.4156691	0.3473820
10101	0.7175147	0.4156691	0.3473820
10110	0.7179275	0.4158861	0.3444508
10111	0.7179275	0.4158861	0.3444508
11000	0.7572505	0.3647205	0.3400572
11001	0.7572505	0.3647205	0.3400572
11010	0.7566312	0.3632813	0.3408812
11011	0.7566312	0.3632813	0.3408812
11100	0.7598307	0.3595799	0.3259984
11101	0.7598307	0.3595799	0.3259984
11110	0.7592115	0.3585286	0.3289831
11111	0.7592115	0.3585286	0.3289831

a)

Rule set	Accuracy	L2	ROC ca5
00000	0.0157356	1.9685289	0.0000000
00001	0.0157356	1.9685289	0.0000000
00010	0.0157356	1.9685289	0.0000000
00011	0.0157356	1.9685289	0.0000000
00100	0.0157356	1.9685289	0.0000000
00101	0.0157356	1.9685289	0.0000000
00110	0.0157356	1.9685289	0.0000000
00111	0.0157356	1.9685289	0.0000000
01000	0.0157356	1.9554551	0.0000000
01001	0.0157356	1.9554551	0.0000000
01010	0.0157356	1.9554583	0.0000000
01011	0.0157356	1.9554583	0.0000000
01100	0.0157356	1.9554551	0.0000000
01101	0.0157356	1.9554551	0.0000000
01110	0.0157356	1.9554583	0.0000000
01111	0.0157356	1.9554583	0.0000000
10000	0.5898568	0.6338138	0.2320315
10001	0.5898568	0.6336678	0.2320315
10010	0.5895172	0.6343928	0.2281325
10011	0.5895172	0.6342467	0.2298608
10100	0.5898568	0.6338138	0.2320315
10101	0.5898568	0.6336678	0.2320315
10110	0.5895172	0.6343928	0.2281325
10111	0.5895172	0.6342467	0.2298608
11000	0.5969887	0.6236647	0.2343794
11001	0.5969887	0.6235338	0.2350431
11010	0.5967623	0.6237590	0.2262164
11011	0.5967623	0.6236281	0.2262164
11100	0.5969887	0.6236647	0.2343794
11101	0.5969887	0.6235338	0.2350431
11110	0.5967623	0.6237590	0.2262164
11111	0.5967623	0.6236281	0.2262164

b)

Table 6: Metrics of YARBUS with various rule sets on a) the dstc2_test and b) the dstc3_test datasets. The meaning of the rule set number is defined in section 3.4. The trackers were run on the SJTU + sys SLU for the DSTC2 dataset and SJTU + err-tied SLU for the DSTC3 dataset.