Learning Dialogue Dynamics with the Method of Moments

Merwan Barlier, Romain Laroche, Olivier Pietquin

LEARNING DIALOGUE DYNAMICS WITH THE METHOD OF MOMENTS

Merwan Barlier[1,2]*, Romain Laroche*[1]*, Olivier Pietquin*[2,3]

[1] NaDia Team, Orange Labs - Châtillon, France
[2] Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 CRIStAL - Lille, France
[3] Institut Universitaire de France

merwan.barlier@orange.com, romain.laroche@orange.com, olivier.pietquin@univ-lille1.fr

ABSTRACT

In this paper, we introduce a novel framework to encode the dynamics of dialogues into a probabilistic graphical model. Traditionally, Hidden Markov Models (HMMs) would be used to address this problem, involving a first step of handcrafting to build a dialogue model (e.g. defining potential hidden states) followed by applying expectation-maximisation (EM) algorithms to refine it. Recently, an alternative class of algorithms based on the Method of Moments (MoM) has proven successful in avoiding issues of the EM-like algorithms such as convergence towards local optima, tractability issues, initialization issues or the lack of theoretical guarantees. In this work, we show that dialogues may be modeled by SP-RFA, a class of graphical models efficiently learnable within the MoM and directly usable in planning algorithms (such as reinforcement learning). Experiments are led on the Ubuntu corpus and dialogues are considered as sequences of dialogue acts, represented by their Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA). We show that a MoM-based algorithm can learn a compact model of sequences of such acts.

1. INTRODUCTION

The ability to provide models of spontaneous human-human dialogue is an important step in understanding dialogue, leading to applications such as goal-oriented human-machine dialogue [1], chatbots [2] computer-aided human-human dialogue [3]. Hidden Markov Models [4] are the traditional framework addressing this problem [5]. According to this framework, dialogue is seen as a stochastic process transiting through non-observable states, called *dialogue states*, and emitting an observable symbol, an utterance, while transiting. Since dialogue states are, by definition, not directly observable, monitoring the dialogue progress is made by maintaining a probability distribution over those states [6].

Previous work addressing the problem of learning the latent structure of dialogues [7] adopt the following strategy: a first model is handcrafted, its parameters are then adjusted using data by maximizing the likelihood of the model with algorithms such as Markov Chain Monte-Carlo or Baum-Welch/Expectation-Maximization (EM) [8, 9, 10]. Those algorithms rely on a alternation between optimization and update steps. However, three problems arise naturally from this strategy. First, it requires to know what may be the different hidden states in order to build the first model. This state design is generally handcrafted. Models are therefore strongly domain dependent and may suffer from human modeling errors. Second, the EM and MCMC algorithms are iterative procedures converging towards local optima. Finally, those algorithms do not scale well with the number of samples and the number of hidden states.

In this work, we show how these problems can be efficiently handled by modeling dialogue as a Multiplicity Automaton and using the Method-of-Moments.

Multiplicity Automata (MA), introduced in [11], are a general class of graphical models, including HMMs, able to compute a large variety of functions, including probability distributions. Modeling probability distributions as MA allow inference within spectral algorithms [12, 13], which are a class of methods based on the Method-of-Moments which learns the representation of a distribution. Their first interest is that the representation learned by the algorithm is expressed in terms of observable variables, avoiding directly modeling errors. More precisely, in this case, hidden states may be understood as possible futures. Those algorithms also distinguish from EM algorithms by being extremely fast and often coming with theoretical guarantees. They have also shown their efficiency on a wide variety of Natural Language Processing tasks such as learning Latent Dirichlet Allocation (LDA) [14], Probabilistic Context Free Grammars [15], distributed representations [16] or parsing [17].

However, a well-known problem of these algorithms is that in the case of HMMs, they are not able to provide valid parameter estimates. This problem is of major importance since in that case, the learned model is not directly usable for planning algorithms, which rely at the heart of dialogue systems since [18] . In this work, we show that it is possible to bypass this difficulty by casting dialogue as a particular type of multiplicity automaton, called Probabilistic Residual

Finite Automaton (PRFA), which is efficiently learnable [19].

This paper is organized as follows: in Section 2, we briefly recall the Hidden Markov Model framework and formally introduce the class of Probabilistic Finite Automata; in Section 3, we show how dialogue model learning may be cast as a PRFA learning problem; in Section 4, we show the efficiency of our approach by learning dialogue models for the Ubuntu Corpus.

2. BACKGROUND

2.1. Hidden Markov Models

From the point of view of an external observer, a dialogue may be seen as a generator of symbols, or observations o, drawn from a set Σ. Depending on the observer, those observations may be utterances, dialogue acts or words. One fundamental task when studying such phenomena is to perform *inference*, *ie.* to predict what may be the next observations given the past. Formally, one assumes that each observation is drawn according to some distribution p:

$$p(o_0, o_1, ...) = p(o_0) \prod_{t>0} p(o_t|h_t),$$

where $h_t = \{o_0, ..., o_{t-1}\}$ is called *history*. Computing this distribution for all possible histories is often intractable, a problem known as the *curse of history*. A compression of those histories called *states* are often introduced. In a dialogue context, those states often contains the last utterance and some relevant information about the past [20]. This relevant information is however not precisely defined and in practice, it is often handcrafted and strongly domain-dependent. Furthermore, when spoken dialogues are listened by a machine, the correspondence between states and observations may be fuzzy, since speech recognition and spoken language understanding may be error-prone. This problem is commonly addressed within the Hidden Markov Models (HMM) paradigm. Formally,

Definition 2.1. An HMM is an uncontrolled Markov Process or a tuple $\{Q, \Sigma, T, O, b_0\}$ where:

- $Q = (s_1, \ldots, s_n)$ is a finite set of states
- Σ is a finite set of observations
- $T = [T_{i,j}]_{i,j \in Q}$ is the transition matrix
- $O = [O_{o,i}]_{o \in \Sigma, i \in Q}$ is the observation matrix
- $b_0 = [b_0(i)]_{i \in Q}$ is the initial *belief vector*, representing the initial distributions over states.

It is often impossible to know exactly what is the current state of the process. But in a wide variety of applications, such as control and thus human-machine dialogue, it is sufficient to maintain a distribution over the different states. This distribution forms a vector called *belief state*. This belief is updated thanks to the *update equation*, which requires to know the parameters of the HMMs. Those parameters may be learned thanks to algorithms such as Baum-Welch, MCMC... However, it is known that those algorithms need a lot of computational resource and slowly converge towards bad local optima (or require strong prior assumptions).

In the remainder of this section, we will present a new paradigm addressing the inference problem, surpassing this traditional model on a large variety of points.

2.2. Stochastic Processes and Hankel Matrices

In the remainder of the paper, we will adopt the following terminology and notations. Let Σ be the set, called *alphabet*, containing all the possible observations. The concatenation between two elements x and y of Σ will be noted xy. A sequence of observations is called a *word*. The set of words over the alphabet Σ is denoted by Σ^*. The empty word will be denoted by ϵ. Words will be written by variables with bars. The cardinal of a set Q will be denoted by $|Q|$.

A function $p : \Sigma^* \to K$, with $K = \mathbb{R}$ or $\mathbb{R}_+$ is called a formal series. The function p characterizing dialogue defined in the last section is a particular type of formal series called *stochastic process*.

Definition 2.2. [21] A *stochastic process* is a function $p : \Sigma^* \to [0, 1]$ that satisfies

- $p(\epsilon) = 1$
- $\forall \overline{x} \in \Sigma^* : p(\overline{x}) = \sum_{x \in \Sigma} f(\overline{x}x)$

A stochastic process p defines then probabilities of initial observation sequences.

Given a stochastic process $p : \Sigma^* \to \mathbb{R}_+$, one defines its Hankel matrix $H \in \mathbb{R}_+^{\Sigma^* \times \Sigma^*}$ as the bi-infinite matrix $H = [p(xy)]_{x,y \in \Sigma^*}$:

$$H = \begin{pmatrix} p(\epsilon) & p(a) & p(b) & p(aa) & \ldots \\ p(a) & p(aa) & p(ab) & p(aaa) & \ldots \\ p(b) & p(ba) & p(bb) & p(baa) & \ldots \\ p(aa) & p(aaa) & p(aab) & p(aaaa) & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Rows (resp. columns) of this matrix correspond thus to *prefixes* (resp. *suffixes*) of the words which will be taken as arguments of the formal series p.

It is important to notice that the Hankel matrix is not a model of the stochastic process but an equivalent representation. The ability to compute H gives thus the complete knowledge of p. Since both representations are indifferently interchangeable, one may define the rank of the stochastic process as the rank of H: $\text{rank}(p) := \text{rank}(H)$. A remarkable feature of the Hankel matrix is that it contains all the

information concerning the stochastic process but is defined only over observable elements. No hidden state is therefore introduced.

In the next section, we present Multiplicity Automata, a particular class of graphical models closely related to the Hankel matrix, which will allow its computation.

2.3. Multiplicity Automata

Multiplicity Automaton (MA), introduced in [11], are a a class of graphical models realizing formal series. Formally,

Definition 2.3. A K-Multiplicity Automata (K-MA) is a tuple $\{\Sigma, Q, (A_o)_{o\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty\}$ where:

- Σ is an alphabet
- Q a set of states
- $\boldsymbol{\alpha}_0 \in K^{|Q|}$ the initialization vector
- $A_o \in K^{|Q|\times|Q|}$ the transition matrices
- $\boldsymbol{\alpha}_\infty \in K^{|Q|}$ the termination vector

The dimension of the MA is defined by $d = |Q|$. Let A be a K-MA realizing a formal serie $f_A : \Sigma^* \to K$ defined by:

$$\begin{aligned}\forall \overline{u} \in \Sigma^*,\ f_A(\overline{u}) &= \boldsymbol{\alpha}_0^\top A_{\overline{u}} \boldsymbol{\alpha}_\infty \\ &= \boldsymbol{\alpha}_0^\top A_{u_1} \dots A_{u_n} \boldsymbol{\alpha}_\infty\end{aligned}$$

SP-NFA are $\mathbb{R}_+$-MA realizing a stochastic process.

In the general case, no assumption is made on the weights. If $K = \mathbb{R}_+$, ignoring a normalization factor, weights may be interpreted as transition probabilities. But if $K = \mathbb{R}$, they lose this probabilistic interpretation. Furthermore, it has been shown [22, 23] that every HMM with d hidden states may be converted into a K-MA with dimensions at most d while conversely, a d-dimensional SP-NFA may not be represented by a finite-state HMM. A first advantage of both SP-NFA and OOM over HMMs models is that they are thus as expressive as HMM while being at least as compact.

The following theorem gives sufficient constraints on the parameters of an $\mathbb{R}_+$-MA to define a SP-NFA [22]:

Theorem 2.1. *An $\mathbb{R}_+$-MA such that $\boldsymbol{\alpha}_0^T \mathbb{1} = 1$, $\sum_{o\in\Sigma} A_o \mathbb{1} = \mathbb{1}$ and $\boldsymbol{\alpha}_\infty^T \mathbb{1} = 1$ realizes a stochastic process.*

MA are linked with Hankel matrices thanks to the following theorem [24]:

Theorem 2.2. *Let f_A be a formal series realized by an MA with n states, then rank$(H_{f_A}) = n$. Conversely, if the Hankel matrix H_f of a formal series f has rank n, then f can be realized by an MA with n states, but not less.*

This theorem states that every formal series (including stochastic processes) may be realized by an MA with a number of states equals to the rank of the formal series.

In the next section, we show how one can learn such MA (and thus our distribution p) from a batch of data.

3. LEARNING STOCHASTIC PROCESSES WITH THE METHOD-OF-MOMENTS

3.1. The Method-of-Moments

Algorithms based on the Methods-of-Moments (MoM) attempt to learn the parameters of an automata realizing a stochastic process by identifying them with the empirical moments of the stochastic process. Those algorithms mainly rely on the Spectral Value Decomposition (SVD) of the Hankel matrix, they are hence often called Spectral algorithms.

Those algorithms may be decomposed into the following procedure:

Input : Batch trajectories
Output: An K-MA approximating p
1. Find a basis $\mathcal{B}$ of the Hankel matrix H_p
2. Find an MA with $|\mathcal{B}|$ states, verifying constraints of theorem 2.1 and realizing p

However, these algorithms have an important drawback. They are improper, which means that even if the learned automaton may be arbitrarily close from the original one, they do not need to belong to the same class. In fact, applying this method often leads to automata with negative weights. Thus, the automaton will not be usable for planning. This problem is called Negative Probability Problem (NPP).

To avoid the NPP, constraints have to be inserted into the second step of the previous procedure. In the case of HMMs however, it has been shown [25] that the number of constraints necessary to return an SP-NFA was infinite. This is why in this work, we consider SP-RFA, a smaller class of automata, which is efficiently learnable within the MoM and adapted to model dialogues.

3.2. SP-RFA

SP-RFA are a class of automata realizing stochastic processes slightly less general than SP-NFA but efficiently learnable. Formally,

Definition 3.1. An SP-RFA is an SP-NFA $\{\Sigma, Q, \boldsymbol{\alpha}_0, (A_o)_{o\in\Sigma}, \boldsymbol{\alpha}_\infty\}$ realizing a distribution p such that for all states $q \in Q$, there exists a word $\overline{u} \in \Sigma^*$ such that $\boldsymbol{\alpha}_0 A_{\overline{u}} = \mathbb{1}_q$.

SP-RFA may be understood as stochastic processes whose generality lies between n-th order Markov chains and HMMs. In a dialogue context, if we assume that SP-RFA models a conversation, it means that whatever the dialogue state may be, there will be a sequence of utterances which leads to this dialogue state with probability one. This is a reasonable assumption since one often considers that the dialogue state depends only on the noiseless history.

As [19] show, SP-RFA may be characterized by a finite set of constraints. It is thus possible to design an algorithm based on the MoM which will output an SP-NFA.

To characterize those constraints, one first needs to define the following operators. Let p be a formal series and $o \in \Sigma$ a symbol. The function $\dot{o}p$ is the formal series such that $\forall u \in \Sigma$, $\dot{o}p(u) = p(ou)$. Then, for all $u \in \Sigma$ such that $p(u\Sigma^*) > 0$, one defines p_u as the formal series $p_u = \frac{\dot{u}p}{p(u\Sigma^*)}$, where $u\Sigma^*$ is the set of all words beginning by u. One may thus understand $p_u(x)$ as the conditional probability of x given the prefix u.

The first property gives sufficient conditions on the coefficients of an MA to realize a SP-RFA.

Property 3.1. *Let p be a distribution, if there exists a set of words $\mathcal{R}$ and two associated sets of non-negative reals $\{a^v_{u,o}\}_{u,v\in\mathcal{R},o\in\Sigma}$ and $\{a^v_\epsilon\}_{v\in\mathcal{R}}$ such that $\forall u \in \mathcal{R}, \overline{p}(u) > 0$ and*

$$\forall u \in \mathcal{R}, o \in \Sigma, \dot{o}p_u = \sum_{v\in\mathcal{R}} a^v_{u,o} p_v$$
$$\text{and } p = \sum_{v\in\mathcal{R}} a^v_\epsilon p_v$$

then, $\{\Sigma, S, (A_o)_{o\in\Sigma}, \boldsymbol{\alpha}_0, \alpha_\infty\}$ defines a SP-RFA realizing p, where $Q = \mathcal{R}$,

$$\boldsymbol{\alpha}_0^\top = (a^u_\epsilon)^\top_{u\in\mathcal{R}} \forall u, v \in \mathcal{R}, A_o[u,v] = a^v_{u,o}.$$

Conversely, the second property assures the existence of these coefficients for SP-RFA:

Property 3.2. *Let $\{\Sigma, Q, (A_o)_{o\in\Sigma}, \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty\}$ be a SP-RFA and p the distribution it realizes, then there exists a set of words $\mathcal{R}$ such that:*

$$\forall u \in \mathcal{R}, \overline{p}(u) > 0$$
$$p \in \left\{ \sum_{u\in\mathcal{R}} \alpha_u p_u | \alpha_u \in \mathbb{R}_+ \right\}$$
$$\forall u \in \mathcal{R}, o \in \Sigma, \dot{o}p_u \in \left\{ \sum_{v\in\mathcal{R}} \alpha_v p_v | \alpha_v \in \mathbb{R}_+ \right\}$$

The sets $\{\sum_{e\in E} \alpha_e e | \alpha_e \in \mathbb{R}_+\}$ are called conical hull of the set E. From now on, they will be denoted as $\text{coni}(E)$.

3.3. Learning SP-RFA

In this section, we present the algorithm CH-SP-RFA (Algorithm 2) of [19] which, given a stochastic process p generated by an SP-RFA, learns an automaton realizing p with proper probabilities as weights, and being as such directly usable to initialize Baum-Welch algorithm or to use planning algorithm.

The algorithm takes as input a basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ and works as follows.

First, one estimates the formal series $\hat{\mathbf{p}}$ and $\dot{o}\hat{\mathbf{p}}$ (where $\mathbf{p}$ denotes the vectorial representation of p). They will be used to learn the coefficients of the Proposition 3.1.

Second, since by assumption data have been generated by a SP-RFA, one looks for the conical hull of Proposition 3.2 in which lies our distribution. This conical hull may be found by a Non-Negative Matrix Factorization algorithm. Following the advice of [26], in our experiments, we used the Successive Projection Algorithm [27].

Third, one regresses the coefficients given by Property 3.1 in the previous conical hull conditioned on the constraints of Theorem 2.1 to return a stochastic process.

One may notice that the MA returned by the algorithm is not necessarily a SP-RFA. It has been assumed that the formal series we try to learn is one but in practice, one only needs to compute a stochastic process.

Input : A target dimension d, a separable complete basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, and a training set.
Output: A SP-NFA $\{\Sigma, S, (A_o)_{o\in\Sigma}, \boldsymbol{\alpha}_0, \alpha_\infty\}$
for $u \in \hat{\mathcal{P}}$ **do**
  Estimate $\hat{\mathbf{p}}_u$ and $\dot{o}\hat{\mathbf{p}}_u$ from the training set
  $\hat{\mathbf{d}}_u \leftarrow (\hat{\mathbf{p}}_u^\top \ \dot{o}_1\hat{\mathbf{p}}_u^\top \ \dots \dot{o}_{|\Sigma|}\hat{\mathbf{p}}_u^\top)$
end
Find $\hat{\mathcal{R}}$ a subset of d prefixes of $\mathcal{P}$ such that $\forall u \in \hat{\mathcal{R}}, \hat{\mathbf{d}}_u > 0$ and $\hat{\mathbf{d}} \in \text{coni}(\hat{\mathbf{d}}_u | u \in \hat{\mathcal{R}})$
for $u \in \hat{\mathcal{R}}$ **do**
  $$\{\hat{a}^v_{u,o}\} \leftarrow \underset{a^v_{u,o}}{\text{argmin}} \sum_{o\in\Sigma} \left|\left| \dot{o}\mathbf{p}_u - \sum_{v\in\hat{\mathcal{R}}} a^v_{u,o}\hat{p}_v \right|\right|_2$$
  $$\text{st} \sum_{v\in\hat{R},o\in\Sigma} a^v_{u,o}\dot{o} = 1,\ \hat{\mathbf{p}}_u \mathbb{1}_\epsilon = 1 \text{ and } a^v_{u,o} \geq 0$$
end
$$\{\hat{a}^u_\epsilon\} \leftarrow \underset{\{a^u_\epsilon\}}{\text{argmin}} \left|\left| \hat{\mathbf{p}} - \sum_{u\in\hat{\mathcal{R}}} a^u_\epsilon \right|\right|_2$$
$$\text{st} \sum_{u\in\hat{\mathcal{R}}} a^u_\epsilon = 1 \text{ and } a^u_\epsilon \geq 0$$
$$\hat{\boldsymbol{\alpha}}_0^\top \leftarrow (\hat{a}^u_\epsilon)^\top_{u\in\mathcal{R}}$$
for $o \in \Sigma$ **do**
  $$\hat{A}_o \leftarrow (a^v_{u,o})_{u,v\in\hat{\mathcal{R}}}$$
end

Algorithm 1: CH-SP-RFA Algorithm

4. EXPERIMENTAL SETTING

4.1. The Ubuntu Dialogue Corpus

Experiments were led on the Ubuntu dialogue corpus [28], which consists of 1 million dyadic chat dialogues of 3 turns or more extracted from the Ubuntu IRC channel logs. Dialogues in this dataset always follow the same structure: a user asks

first the chatroom some question, which is then answered by another user.

Pre-processing steps have then been applied on the corpus. First, utterances were tokenised using the twokenize tokeniser [1]. Then, stemming and lemmatising were applied using the NLTK toolkit [29]. Finally, stopwords from NLTK and words appearing less than 10 times were removed.

We used 2 different representations of utterances which will be described in the following sections: Latent Dirichlet Allocation and Latent Semantic Analysis, all of them computed with the Gensim library [30]. Additionally, we kept the `End of turn` token and add another one `End of dialogue` at the end of each dialogue.

4.2. Representation of utterances

4.2.1. Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [31] represents the utterances as a mixture of topics and that each word in this utterance is generated by one of these topics. More precisely, LDA supposes that each word in an utterance is generated according to the following procedure:

Input : LDA of an utterance - Size of the utterance
Output: Utterance as a Bag-of-Words
1. Draw a topic from the LDA
2. Draw a word from the distribution induced by the topic

In the following experiments, LDA vectors are a mixture over 30 topics.

4.2.2. Latent Semantic Analysis

Latent Semantic Analysis (LSA) [32] projects utterances into a *concept* space where two documents about the same concept are close to each other.

LSA achieves this by performing an SVD on the term-document matrix, which is defined as the matrix containing word counts of each document. To improve the efficiency of the LSA, we did not fill the term-document matrix with word counts but with their tf-idf values which penalizes frequent words and advantages rare ones.

We chose in the experiments to define the LSA over 100 concepts.

4.3. Computing the estimated Hankel matrix

The Hankel matrix was estimated with the suffix-history algorithm [33]. This algorithm considers all n-grams (with n a parameter of the algorithm) as if they were separate training sequences. More precisely, each n-gram is considered as prefix and one then counts its following suffixes. The obtained matrix is finally renormalized in order to obtain probabilities.

In the experiments, we considered 4-grams.

[1] https://github.com/myleott/ark-twokenize-py

Model Size	CH-SP-RFA	Baum-Welch
10	198	879
20	200	3201
30	202	7260
40	207	-

Fig. 1. Computation time for LDA representation (in seconds)

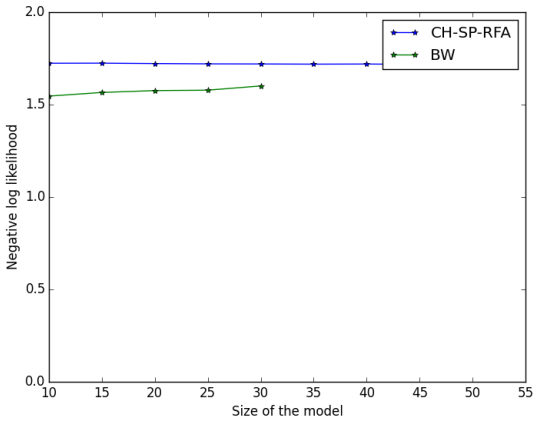

Fig. 2. Performance with the LDA representation

4.4. Results

To compare the performances of the CH-SP-RFA and Baum-Welch algorithms, we chose the negative-log-likelihood metric and execution times.

First, we kept from the Ubuntu dialogue corpus only the dialogues of at least 15 turns, leading to a train (resp. test) set of 257 659 (resp. 4516) dialogues. In order to map continuous LDA and LSA vectors into a set of discrete observations, they were clustered into 30 and 50 categories respectively, leading to alphabets of size 32 and 52 (since `End of turn` and `End of dialogue` tokens have to be learned).

In order to avoid getting stuck in local optima, we used 3 random restarts for the Baum-Welch algorithm. Fig.1 and Fig.2 show the negative-log-likelihood for LDA and LSA respectively. In both of the experiments, the belief is initialized on the first 10 turns of dialogue and the likelihood is then computed on the prediction of the next observations. We did not computed the Baum-Welch algorithm for models with a dimension bigger than 30 due to a long computation time. We observe that in both experiments, results of Baum-Welch algorithms are better than the ones of CH-SP-RFA, even though

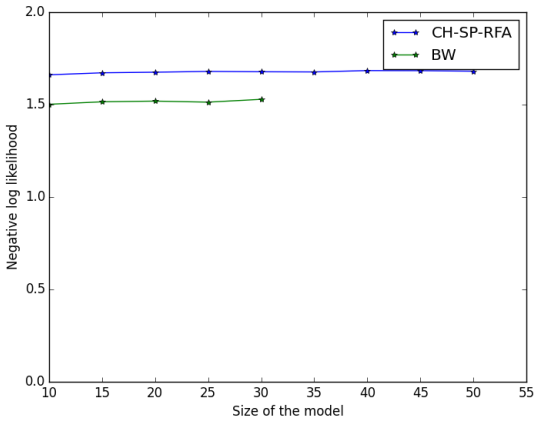

Fig. 3. Performance with the LDA representation

we observe the order of magnitude is the same. On the LDA representation, one may notice that performances of Baum-Welch also decrease with the size of the model, indicating a tendency to overfit.

However for comparables performances, Fig. 3 shows that the running time of CH-SP-RFA is drastically smaller than the one of Baum-Welch. This can be explained by the fact that the sample complexity is linear for CH-SP-RFA since one needs to browse data only once when constructing the Hankel matrix, whereas Baum-Welch has to browse all the data at each iteration of the algorithm. Furthermore, due to local convergence problems, Baum-Welch has to be run several time with different initial models. Finally, the Hankel matrix has to be computed only once for all the experiments, this step being by far the longest of the CH-SP-RFA algorithm and lasting 197 seconds.

Finally, one observes that small models perform almost as well as bigger ones. This may be explained by the fact that the chosen representations are not rich enough to completely catch the meaning of utterances. More sophisticated representations could have been used but it would have lead to bigger alphabet and in that case, Baum-Welch would have been totally intractable.

For these scalability issues, we conclude that spectral algorithms and more especially CH-SP-RFA, by providing almost equivalent results while being order of magnitudes faster, are a good alternative to the Baum-Welch algorithm which is unable to deal with large amounts of data or large alphabets.

5. FURTHER WORK AND CONCLUSION

In this work, we suggest the use of spectral methods for learning linear dialogue models. We justified that modeling dialogue as a SP-RFA was reasonable which allowed us to use the CH-SP-RFA algorithm in order to learn a dialogue model from the Ubuntu dialogue corpus. We experimentally showed that the performance of CH-SP-RFA was in this case comparable to the one of Baum-Welch algorithm while being considerably faster.

This work opens the door to a wide range of extensions. First, we would like to generalize the setting to the continuous observation case [34] in order to work with more expressive features such as distributed representation of utterances [35, 36]. We also want to extend this work for controlled processes [37] in order to apply it to human-machine dialogue and computer-aided human-human dialogue. Finally, we would like to apply those set of techniques for Inverse Reinforcement Learning in partially observable environments [38] in order to learn the preferences of a dialogue system's user.

6. REFERENCES

[1] Oliver Lemon and Olivier Pietquin, *Data-Driven Methods for Adaptive Spoken Dialogue Systems: Computational Learning for Conversational Interfaces* , Springer, 2012.

[2] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Proc. of AAAI*, 2016.

[3] Merwan Barlier, Romain Laroche, and Olivier Pietquin, "A stochastic model for computer-aided human-human dialogue," in *Proc. of Interspeech*, 2016.

[4] Lawrence R Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[5] Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer, "Dialogue act modeling for automatic tagging and recognition of conversational speech," *Computational linguistics*, vol. 26, no. 3, pp. 339–373, 2000.

[6] Jason D. Williams, Antoine Raux, and Matthew Henderson, "The dialog state tracking challenge series: A review," *Dialogue and Discourse*, vol. 7, no. 3, pp. 4–33, 2016.

[7] Jan Alexandersson and Norbert Reithinger, "Learning dialogue structures from a corpus.," in *Proc. of Eurospeech*, 1997.

[8] Srinivas Bangalore, Giuseppe Di Fabbrizio, and Amanda Stent, "Learning the structure of task-driven

human-human dialogs," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 7, pp. 1249–1259, 2008.

[9] Michael J Paul, "Mixed membership markov models for unsupervised conversation modeling," in *Proc. of EMNLP*, 2012.

[10] Ke Zhai and Jason D Williams, "Discovering latent structure in task-oriented dialogues," in *Proc. of ACL*, 2014.

[11] Marcel Paul Schützenberger, "On the definition of a family of automata," *Information and control*, vol. 4, no. 2, pp. 245–270, 1961.

[12] Daniel Hsu, Sham M Kakade, and Tong Zhang, "A spectral algorithm for learning hidden markov models," in *Proc. of COLT*, 2012.

[13] Raphaël Bailly, François Denis, and Liva Ralaivola, "Grammatical inference as a principal component analysis problem," in *Proc. of ICML*, 2009.

[14] Anima Anandkumar, Yi-kai Liu, Daniel J Hsu, Dean P Foster, and Sham M Kakade, "A spectral algorithm for latent dirichlet allocation," in *Proc. of NIPS*, 2012, pp. 917–925.

[15] Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle H Ungar, "Experiments with spectral learning of latent-variable pcfgs.," in *Proc. of HLT-NAACL*, 2013, pp. 148–157.

[16] David Belanger and Sham Kakade, "A linear dynamical system model for text," in *Proc. of ICML*, 2015, pp. 833–842.

[17] Franco M Luque, Ariadna Quattoni, Borja Balle, and Xavier Carreras, "Spectral learning for non-deterministic dependency parsing," in *Proc. of EACL*, 2012, pp. 409–419.

[18] Esther Levin and Roberto Pieraccini, "A stochastic model of computer-human interaction for learning dialogue strategies.," in *Proc. of Eurospeech*, 1997.

[19] Hadrien Glaude and Olivier Pietquin, "Pac learning of probabilistic automaton based on the method of moments," in *Proc. of ICML*, 2016.

[20] Staffan Larsson and David R Traum, "Information state and dialogue management in the trindi dialogue move engine toolkit," *Natural language engineering*, vol. 6, no. 3&4, pp. 323–340, 2000.

[21] Michael Thon and Herbert Jaeger, "Links between multiplicity automata, observable operator models and predictive state representations: a unified learning framework," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 103–147, 2015.

[22] Pierre Dupont, François Denis, and Yann Esposito, "Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms," *Pattern recognition*, vol. 38, no. 9, pp. 1349–1371, 2005.

[23] Herbert Jaeger, *Discrete Time, Discrete Valued Observable Operator Models: A Tutorial*, GMD-Forschungszentrum Informationstechnik, 1998.

[24] Jack W. Carlyle and Azaria Paz, "Realizations by stochastic finite automata," *Journal of Computer and System Sciences*, vol. 5, no. 1, pp. 26–40, 1971.

[25] Yann Esposito, *Contribution à linférence dautomates probabilistes*, Ph.D. thesis, Université Aix-Marseille, 2004.

[26] Hadrien Glaude, Cyrille Enderli, and Olivier Pietquin, "Spectral learning with proper probabilities for finite state automaton," in *Proc. of ASRU*, 2015.

[27] Nicolas Gillis and Stephen A Vavasis, "Semidefinite programming based preconditioning for more robust near-separable nonnegative matrix factorization," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 677–698, 2015.

[28] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau, "The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems," *Proc. of SigDial*, 2015.

[29] Steven Bird, "Nltk: the natural language toolkit," in *Proc. of COLING*, 2006.

[30] Radim Rehurek and Petr Sojka, "Software framework for topic modelling with large corpora," in *Proc. of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010.

[31] David M Blei, Andrew Y Ng, and Michael I Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[32] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391, 1990.

[33] Britton Wolfe, Michael R James, and Satinder Singh, "Learning predictive state representations in dynamical systems without reset," in *Proc. of ICML*, 2005.

[34] Byron Boots, Geoffrey Gordon, and Arthur Gretton, "Hilbert space embeddings of predictive state representations," in *Proc. of ICML*, 2013.

[35] Nal Kalchbrenner and Phil Blunsom, "Recurrent convolutional neural networks for discourse compositionality," *Proc. of the 2013 Workshop on Continuous Vector Space Models and their Compositionality*, 2013.

[36] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu, "Towards universal paraphrastic sentence embeddings," *Proc. of ICLR*, 2016.

[37] Kamyar Azizzadenesheli, Alessandro Lazaric, and Animashree Anandkumar, "Reinforcement learning of pomdps using spectral methods," *Proc. of COLT*, 2016.

[38] Jaedeug Choi and Kee-Eung Kim, "Inverse reinforcement learning in partially observable environments," *The Journal of Machine Learning Research*, vol. 12, pp. 691–730, 2011.