Building Document Treatment Chains Using Reinforcement Learning and Intuitive Feedback

Esther Nicart, Bruno Zanuttini, Hugo Gilbert, Bruno Grilhères, Fredéric Praca

Building document treatment chains using reinforcement learning and intuitive feedback

Esther Nicart*†, Bruno Zanuttini†, Hugo Gilbert‡, Bruno Grilhères§, Fredéric Praca†
*Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France; *first.last-name@unicaen.fr*
†Cordon Electronics DS2i, France; *first.last-name@cordonweb.com*
‡LIP6, Paris, France; *first.last-name@lip6.fr*
§Airbus Defence and Space, Élancourt, France; *first.last-name@airbus.com*

***Abstract*—We model a document treatment chain as a Markov Decision Process, and use reinforcement learning to allow the agent to learn to construct and continuously improve custom-made chains "on the fly". We build a platform which enables us to measure the impact on the learning of various models, web services, algorithms, parameters, *etc.* We apply this in an industrial setting, specifically to an open source document treatment chain which extracts events from massive volumes of web pages and other open-source documents. Our emphasis is on minimising the burden of the human analysts, from whom the agent learns to improve guided by their feedback on the events extracted. For this, we investigate different types of feedback, from numerical feedback, which requires a lot of tuning, to partially and even fully qualitative feedback, which is much more intuitive, and demands little to no user calibration. We carry out experiments, first with numerical feedback, then demonstrate that intuitive feedback still allows the agent to learn effectively.**

I. Introduction

In the past, the *Open Source INTelligence* (OSINT) analyst's task was finding hidden information. Now, faced with ever-increasing volumes of multi-lingual open source documents, their challenge is to find pertinent information. Many specialised treatment chains have been developed to ease this task [1]. We tackle the generic problem of the improvement of such a chain. Documents are passed continuously through a sequence of web services to extract events (*e.g.*, terrorist attacks) and their characteristics (date, place, agents, *etc.*).

Because of the huge diversity of documents, it is clear that these extractions cannot be perfect, and there cannot be a single optimal chain. Dictionaries are of variable quality. The system is error-prone; *e.g.*, from a school blog recounting "the *bombardment* of a gold *target* by ions during an *atomic* physics demonstration", an atomic bomb attack could erroneously be extracted. The analysts are thus typically forced to correct *a posteriori* the events extracted, an onerous and repetitive task.

The specific application which we examine uses a chain, defined by experts, which consists of a fixed (but potentially conditional) series of individual treatments, such as the detection of the document format, language recognition, translation, extraction of events using nouns and verbs as triggers, *etc.* There is currently no way to improve the chain without the intervention of an expert in consultation with the analysts.

Our objective is to remedy this by providing a mechanism which learns to modify its behaviour in real time, continuously improving the chain, and reducing human effort by decreasing the error rate. This mechanism receives feedback (see below) automatically obtained as the analysts consult the system output. We thus allow the chain to "learn" from its mistakes, *e.g.*, a well-written French article should be translated to English, as there is a wealth of English extraction rules available, but for a *tweet*, a direct extraction is preferable as the dictionaries are insufficient, and translating only introduces noise.

Feedback must be collected non-intrusively, invisible to the analyst. The treatments must remain "black boxes", allowing the analyst to be distanced completely from the extraction process. User expertise can be modelled by tracing their actions [2]. These action traces can be captured in production through the graphical interface giving a summary of the events extracted. We can then base the feedback on the analyst-system interactions that currently occur, *e.g.*, an explicit feedback can be deduced from the distance between an event as *corrected* and the extraction. Our first set of tests (Section VI) demonstrate this; the agent receives a numerical feedback as an automatically generated judgement on the quality of extractions. However, an event may also be *not extracted*, *extracted too slowly*, *etc.* To interpret these "non-actions" as feedback, we must ask "Is it preferable to extract erroneously, or to miss an extraction?", "Should I sacrifice speed for quality?". These questions cannot be answered naturally with a numerical response. We therefore progress to giving intuitive feedback (Section VII), based on easily gathered and naturally expressed user preferences, such as "I prefer an extraction to no extraction" and "I prefer it to be fast".

To our knowledge, an agent capable of dynamically constructing a chain of services, which constantly learns and self-improves from natural human feedback has not yet received academic attention. Nevertheless, the base elements are there:

Chaining together services is not a new concept, *e.g.* for assistance with form-filling [3], or the construction of adaptive, modular chains for photocopiers in real time [4]. The inputs and outputs of each service are known in advance, and it is a planning task to string them together. User preference is not taken into account, there is no measure of the result's quality, and no automatic improvement.

The user can reconfigure the chain using a dedicated language [5], or by choosing a service from a directory [6], but they must have system expertise to appreciate the consequences of their choices, and the chain will not adapt

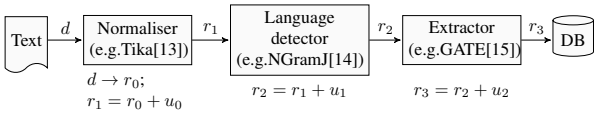


Fig. 1: A simple but typical Weblab chain: a document, d, is converted into an XML resource r_0, annotations u_j are added by each service, finally the results are stored in a database.

without intervention.

Model-based learning from both explicit and implicit human feedback has also been explored. The trainer's positivity can influence the agent [7] or implicit, non-numeric feedback can be provided by inference on the observation of the trainers' actions [8]. Models of the user's behaviour [9] and preferences [10] can be built up through human-system interactions to provide personalised systems. The agent can also learn by demonstrating two policies, which are ordered by the expert [11]. Although these systems react well to changing users and preferences, they require continuous user interaction to confirm or contradict their choice. Our users are not dedicated trainers, and our aim is to reduce human effort.

In summary, we formalise the improvement of a treatment chain as a reinforcement learning problem (Section III), and the chain itself as a Markov Decision Process (Section IV). We demonstrate the viability of automatically learning to chain and parametrise services in three different settings. In the first, the agent is rewarded with numerical feedback on the quality of the treatment, which is very informative but difficult to calibrate (Section VI). The second and third use *intuitive* user feedback (a partially qualitative formalisation, and then a fully qualitative one). These are less informative but require much less, or no calibration (Section VII). We build a platform, *BIMBO* (Benefiting from Intelligent and Measurable Behaviour Optimisation) into which we can "plug" different RL algorithms, models, web services, reward mechanisms *etc.*, which enables us to measure their impact on the learning. We apply this in an industrial setting, specifically to an open-source document treatment chain (Section II).

II. THE WEBLAB PLATFORM

Our industrial application uses the open-source platform *WebLab* [12] for economic, strategic and military surveillance. We also use it for our experiments. WebLab integrates web services which can be interchanged or permuted to create a treatment chain for the analysis and extraction of information from web pages and other open-source multimedia documents.

A typical WebLab treatment chain (Figure 1) first converts the source into an XML resource. This resource is then passed from service to service. Each service analyses the contents of the resource it receives and enriches it with annotations. Finally, the results are stored for the analyst to consult.

Example II.1. Consider the following text:

```
4/29/1971: In a series of two incidents that might
have been part of a multiple attack, suspected
members of the Chicano Liberation Front bombed a
Bank of America branch in Los Angeles, California,
US. There were no casualties but the building
sustained $1 600 in damages.
```

```
1  <resource type="Document" uri="weblab:aaa">
2    <annotation uri="weblab:aaa#a0">
3      <wp:originalContent resource="file:weblab.content"/>
4      <wp:originalFileSize>255</wp:originalFileSize>
5      <dc:source>documents/event.txt</dc:source>
6      <wp:originalFileName>event.txt</wp:originalFileName>
7      <dc:modified>2015-02-14T19:52:21+0100</dc:modified>
8      <wp:collected>2015-02-10T00:11:00+0200</wp:collected>
9    </annotation>
10   <annotation uri="weblab:aaa#a1">
11     <wp:isProducedBy resource="weblab:tika"/>
12     <dc:format>text/plain</dc:format>
13   </annotation>
14   <annotation uri="weblab:aaa#a2">
15     <wp:isProducedBy resource="weblab:ngramj"/>
16     <dc:language>en</dc:language>
17   </annotation>
18   <mediaUnit type="wl:Text" uri="weblab:aaa#0">
19     <content>4/29/1971: In a series of two incidents that might have been part of
         a multiple attack, suspected members of the Chicano Liberation Front bombed a
         Bank of America branch in Los Angeles, California, US. There were no
         casualties but the building sustained $1 600 in damages.</content>
20   </mediaUnit>
21 </resource>
```

Fig. 2: Simplified XML of a WebLab resource mid-chain.

The resource in Figure 2 would be produced by passing the document through a normaliser, adding the annotations "text/plain" (line 12) and original content (line 19); and a language detector adding the language "en" (line 16).

As our chain is dedicated to extracting events, we define these formally (here using the *WebLab* definition [16], but any other definition of an event such as [17] could be used).

Definition 1 (Event)**.** An *event* is a quadruplet (C, T, S, A):

- $C \subseteq \mathbb{C}$ is the conceptual (semantic) dimension. A set of elements taken from the domain $\mathbb{C}$ common to all events;
- T is the temporal dimension (when the event occurred). Potentially ambiguous, *e.g.* "last Tuesday", we take $T \subseteq \mathbb{T}$, where $\mathbb{T}$ is the set of all dates;
- $S \subseteq \mathbb{S}$ is the spatial dimension (where the event occurred), also potentially ambiguous;
- $A \subseteq \mathbb{A}$ is the agentive dimension (the participants).

More precisely, in this article: $\mathbb{C}$ is a fixed and finite set of elements in the ontology *WOOKIE* [16]; $\mathbb{T}$ is the set of all relative and absolute "dates", such as "tomorrow", "2001/9/11"; $\mathbb{S}$ is the set of entities defined in Geonames [18]; $\mathbb{A}$ is the infinite set of all extractable participants, as strings.

Example II.2. An event E could be extracted from the text in Example II.1 with $C = \{AttackEvent, BombingEvent\}$, $T = \{4/29/1971\}$, $S = \{\text{Los Angeles, California, US}\}$, and $A = \{\text{Chicano Liberation Front, Bank of America}\}$.

Building an efficient treatment chain depends not only on choosing the services but also on setting their parameters correctly. For instance, the extractor GATE relies on *gazetteers*, lists of nouns and verbs triggering the detection of a specific type of event (*e.g.* the *bombing* verb gazetteer contains "explode", "detonate", *etc.*). GATE is therefore a *parametrised service*; without suitable *gazetteers* it is ineffective.

III. REINFORCEMENT LEARNING

In production, the treatment chain is complex. It is written and calibrated by experts who choose the services making up the chain, their order, and their parameters. The service order is fixed, but can be conditional (*e.g.*, if the document is in English, send to *extractor*, and to *translator* otherwise). Despite their expertise, it is impossible to create the perfect universal chain. The treatment of open source documents and web pages means that: their format and contents are not standard, the source pages themselves are not controlled, URLs change or are pirated, and there is "noise" (*e.g.*, adverts). The right web services have to be called in the correct order and supplied with the best parameters, and even then may or may not extract useful information from a document. Undetected events and partially or falsely extracted events (*e.g.*, unconnected information in the same sentence erroneously associated) provoke extraction errors. For example, we saw in Section I it is impossible to be certain that "bombardment" refers to aerial bombings. Only once the event has been extracted do we see that the page was a school blog. Even from a specialist web page, the word could refer to the incessant questions of the journalists. Maybe a synonym "shelling" was used, and the event was not recognised.

Before starting the treatment chain, we only know the available services, their parameters, and the potential characteristics of the XML resource and system (see Section II). The agent knows neither the form, nor the content of the documents in advance, nor even if an extraction is possible. Its actions are thus taken under uncertainty. Hence we model the problem as one of reinforcement learning (RL) for Markov Decision Processes (MDPs [19]). We give an overview here but recommend reading [20].

In RL, the learner receives a reward, based on the results of its chosen actions. The closer the results are to the objectives, the higher the reward. The learner tries to maximise these rewards, typically by *exploiting* current knowledge to continue to receive good rewards, or by *exploring* new actions with the hope of obtaining even better ones. RL is generally formalised as an MDP, which models the environment in terms of *states*, in which *actions* are possible, leading to other states stochastically. The fact that the environment is in a given state at a certain instant bestows an immediate reward on the learner (or agent). The agent's objective is to choose actions such that it maximises its expectation of cumulated rewards.

Definition 2 (MDP)**.** A *Markov Decision Process (MDP)* is a quintuplet (S, A, P, R, γ) where:

- S a set of possible states in the environment (here finite);
- A a set of actions available to the agent (here finite);
- P a set of distributions $\{P_a(s,\cdot) \mid s \in S, a \in A\}$; $P_a(s,s')$ is the probability that the environment is in state s' after the agent performs action a in s;
- R a reward function, defined on the states; $R(s)$ is the reward obtained by the agent when it reaches state s;
- $\gamma \in [0,1]$ an attenuation factor, weighting expected future rewards against those currently expected.

In RL, the agent initially only has knowledge of the state / action space $S \times A$, as well as the factor γ. At each instant t, it knows the current state s_t of the environment, and chooses an action a_t. The environment passes into state s_{t+1} according to the probability distribution $P_{a_t}(s_t,\cdot)$, and the agent is informed of the state s_{t+1} and the reward $r_{t+1} = R(s_{t+1})$. The process continues in s_{t+1}. The agent, as it interacts with the environment, learns a series of *policies* $\pi_0, \pi_1, \ldots, \pi_t, \ldots$, where a *policy* $\pi_t : S \to A$ gives, at instant t, the action $\pi_t(s)$ to perform if the current state s_t is s. Its goal at each moment is to maximise the expected accumulated reward, that is, the expected quantity $\sum_{t'=t}^{\infty} \gamma^{t'} R(s_{t'})$. This generic framework encompasses many variations (for a recent summary, see [21]).

Numerous algorithms exist to solve RL problems. Here, we first use standard *Q-learning* [22] with numerical feedback, then for the qualitative setting, which standard approaches cannot handle, we use *SSB Q-learning* [23], but in practise, any RL algorithm could be used. We give overviews of both below to keep this paper self-contained, but encourage the reader to consult the original articles. We also tried RMax [24] and VMax [25], which are conceptually different (learning the underlying MDP instead of a policy directly) but the convergence and calculation time proved prohibitively long for our problem. Note that we present here the general algorithms. User feedback is formalised in Sections VI–VII.

Q-learning is a simple algorithm with easily observable results, and parameters which can be set intuitively. It maintains, for each state / action couple (s,a), a value denoted $\hat{Q}(s,a)$ which represents the agent's current estimate of the expected reward if from s it executes a, then follows an optimal policy. It is controlled using an ϵ-greedy (EG) strategy: when the agent is in state s_t, it chooses a_t which maximises $\hat{Q}(s_t,a_t)$, except with probability ϵ, when it chooses a random action (it *explores*). Observing the new state s_{t+1} and immediate reward r_{t+1}, it updates $\hat{Q}(s_t,a_t)$ as shown in Algorithm 1. The *learning rate* $\alpha \in [0,1]$ fixes the importance of the latest

Algorithm 1: Q-learning

Data: MDP $\mathcal{M}$
while *True* **do**
  Choose a_t using the EG exploration strategy
  Play a_t, observe s_{t+1}, and let $r_{t+1} = \mathcal{R}(s_{t+1})$
  $\hat{Q}_{t+1}(s_t,a_t) \leftarrow \hat{Q}_t(s_t,a_t) + \alpha_t(s_t,a_t)(r_{t+1} + \gamma\, max_b\{\hat{Q}_t(s_{t+1},b)\} - \hat{Q}_t(s_t,a_t))$

experience $(r_{t+1} + \gamma\, max(\ldots))$ with respect to the experience already gained (the previous value of $\hat{Q}(s_t,a_t)$). The *discount factor* $\gamma \in [0,1]$ determines the importance of long-term gain.

Our approach works very well with numerical feedback (Section VI), but very few humans could say "A false extraction is 10.5 times better than a missed extraction, which is 7.9 times worse than a 90 second extraction". More natural is "I prefer a fast, false extraction to a missed extraction". Note that there are two preferences expressed here, which although they impact each other (one might suppose that speed

could result in poor extraction quality), cannot naturally be linked numerically. We therefore want to give feedback based on the decoupled user preferences: "I prefer an extraction to no extraction" and "I prefer it to be fast". This preferential information can be expressed as a partial order over possible results achieved $(f_1, \ldots, f_k)$ (seen as final states) given by the human agent. For instance f_i can stand for "a good extraction in 10 seconds" or "no extraction in 5 seconds", *etc.*

Like *Q-learning*, *SSB Q-learning* (Algorithm 2) uses EG exploration, and it updates the Q-values similarly. However instead of having the numerical values of the rewards given by the environment, they are defined by preference ϕ (see below) and the past experiences of the agent *i.e.* the frequencies $\mathbf{p}_t$ with which it has achieved each possible final result so far; the resulting numerical value is denoted by $\mathcal{R}_{\mathbf{p}_t}(s_{t+1})$.

Probabilistic dominance [26] allows us to maximize the probability of yielding a preferred outcome. Let π_1 and π_2 be two policies and let F_1, F_2 be two random variables on $(f_1, \ldots, f_k)$ where $\mathbb{P}(F_i = f_j)$ is the probability of π_i achieving result f_j. Then $\pi_1 \succeq \pi_2 \Leftrightarrow \mathbb{P}(F_1 \succeq F_2) \geq \mathbb{P}(F_2 \succeq F_1)$. That is, policy π_1 is preferred to π_2 if the expectation of π_1 doing better than π_2 is greater than the converse. Probabilistic dominance is a type of Skew Symmetric Bilinear (SSB) utility function [27]. Given ϕ, an SSB utility function, $\phi(\pi, \pi') > 0$ (resp. $\phi(\pi, \pi') < 0$) means the user prefers π to π' (resp. π' to π) whereas $\phi(\pi, \pi') = 0$ expresses indifference. In probabilistic dominance, ϕ is always 1, 0, or -1, requiring only purely ordinal user feedback.

Algorithm 2: SSB Q-learning

Data: MDP $\mathcal{M}$, SSB function φ

while *True* **do**
- Choose a_t using the EG exploration strategy
- Play a_t, observe s_{t+1}, and let $r_{t+1} = \mathcal{R}_{\mathbf{p}_t}(s_{t+1})$
- $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)(r_{t+1} + \max_b\{Q_t(s_{t+1}, b)\} - Q_t(s_t, a_t))$
- **if** s_{t+1} *is a final state* f_i *and exploration is off* **then**
  - $\mathbf{p}_{t+1} = \mathbf{p}_t + \frac{1}{\eta+1}(\mathbf{1}_i - \mathbf{p}_t)$
  - # η is the number of times $\mathbf{p}$ has been updated.

Finally, for both algorithms we used versions with *eligibility traces* [20, Section 7]. In these, a *decay parameter* $\lambda \in [0, 1]$ controls how far new experiences are back-propagated along the trace of decisions taken. $\lambda = 0$ corresponds to the algorithms as described above, while at the other extreme, $\lambda = 1$ corresponds to Monte-Carlo like RL algorithms.

IV. CHAIN IMPROVEMENT AS RL

Given that it is impossible to create a unique chain capable of perfectly treating every type of document, the ideal is to have a tailor-made chain for each one. We do this by modelling the treatment of a document as an MDP, and its improvement as an RL problem. This is not as trivial as it may seem, as the treatment process and its input are heterogeneous, and yet a way must be found to standardise them into the states and actions of a decision problem. The rewards cannot be specified easily, our users are not dedicated trainers, and we are trying to collect feedback in a non-intrusive fashion.

We define the states using the characteristics of the resource and system at a given moment (Section V). The system thus perceives the task as the *states* of a process, and each passage through a service modifies the current state.

The system also has a certain number of *actions* available which it can apply in the current state. The repetition of the same action on the same document is technically authorised, but will penalise the system, as the treatment will be longer, and due to the massive volumes being treated in production, faster results are preferred. The actions take the system from one state to another, *e.g.*, if 70% of the source documents are in English, the agent will perceive that taking the action "detect language" in a state s_t leads with probability 0.7 to a state s_{t+1} similar to s_t except that it contains the information "language detected" and the annotation "en".

More precisely, the states that the system perceives are combinatorial states, formed from the values of a certain number of *descriptors* of the documents. These states allow a generalisation of the learning; we have already seen that the *type* of source web page (school blog vs. specialist) could greatly influence the utility of the word "bombardment" for the extraction of information. We might hope that the system would learn from the user interactions that the best action is:
– to stop the treatment if the current state descriptor "typeExtracted" is "true" and the descriptor "type" is "school blog";
– to pass the document to an extraction service using the word "bombardment" among the trigger words, if the type is extracted, but not "school blog";
– to pass the document to a type-recognition service otherwise.

Finally, the analysts consult the summaries produced by the system (which show the events extracted and link to the original source documents). The corrections made by the analyst to the extracted information indirectly furnish a feedback on the treatment the document received. The rewards $r(s)$ are thus given to the system only for the final states of a treatment, and are defined in three ways: *a quantitative value* based on the corrections made to the extraction (if any) by the analyst and the document treatment time (Section VI); *a qualitative reward* based on the user's purely ordinal preference on the results; and *a weighted ordinal preference* giving a middle-ground between the first two (Section VII).

V. EXPERIMENTAL FRAMEWORK

To assess the validity of our approach, we ran experiments using a simple, but typical chain, as described in Section II, to which we added the possibility of choosing a "useless" service *Geo*. The chain is written as a *Camel* [28] route in XML. Each service is defined as an endpoint, and we use the *Dynamic Router* to give *BIMBO* control over the services called, their order and their parameters, specifically the choice of *gazetteers* (trigger words) of GATE [15] for event detection in a text. The available actions are therefore to choose the next service from {*Tika, NGramJ, GATE, Geo*}, to choose a GATE gazetteer, or

to *STOP* the treatment and return any extracted events. These services are "black boxes" to the algorithms and *BIMBO*, so that knowledge of, *e.g.*, their WSDLs is unnecessary.

A state is represented by the characteristic attributions:

- information already extracted: *language* $\in$ {"en"," "}; *format* $\in$ {"*text/plain*"," "}; *interesting* $\in$ {*true, false*} (*true* if a given type of event has been extracted); *any* $\in$ {*true, false*} (*true* if any events have been extracted);
- parameters chosen: *nounGazetteer* (the noun gazetteer currently chosen, if any), similarly *verbGazetteer*;
- system state: *timeTakenSoFar* (seconds since the current document treatment started in $timeInterval$ (parametrizable) steps; *nbServices* $\in \{0-5, 6-20, 21+\}$ (the number of services through which the document has passed).

We use an open-source corpus in which the events are already known: the *Global Terrorism Database* (GTD) [29], consisting of details of over 125 000 worldwide terrorist events from 1970 to 2014. We mapped the event types from the GTD to WOOKIE, to construct a set of documents $\{d_1 \dots d_N\}$, from which a perfect chain could extract perfect events $E_1, \dots, E_N$, respectively (as in Examples II.1,II.2), if it had infinite time, and access to an infinite number of perfectly parametrised web services. Such a chain does not exist in real life, so we use the results from an expert chain (constructed by hand) as a reference. We expect our AI to learn to construct chains that eventually perform as well as this expert chain, learning not only the correct order of the services and the best parameters, but also the fact that certain actions (service *Geo*, some GATE gazetteers) are not useful.

As explained in Section III, we test the AI's learning capacity with no *a priori* knowledge. Such an AI is "untrained". After treating a certain number of documents, it learns what it considers to be an optimal policy, becoming "trained". In production, to avoid learning from scratch, we would capitalise on existing expertise by initialising *BIMBO* with a policy (here the Q-Values) of an expert chain.

A. Measuring the quality of the results

To measure the quality of the results, we must first define the similarity between an event extracted from the document by the chain $E_1 = (C_1, T_1, S_1, A_1)$, and the corresponding "perfect" event $E_2 = (C_2, T_2, S_2, A_2)$ in the GTD:

$$\sigma(E_1, E_2) = \frac{a\sigma(C_1, C_2) + b\sigma(T_1, T_2) + c\sigma(S_1, S_2) + d\sigma(A_1, A_2)}{(a + b + c + d)}$$

We define the semantic (resp.geographical) similarity $\sigma(C_1, C_2)$ (resp.$\sigma(S_1, S_2)$) to be 1 if there is a common element between the semantic (resp.geographical) dimensions of E_1 and E_2, and 0 otherwise. For example, for $C_1 = \{Attack, Bombing\}$ and $C_2 = \{Bombing\}$, we obtain $C_1 \cap C_2 = \{Bombing\}$, so $\sigma(C_1, C_2) = 1$.

The temporal similarity $\sigma(T_1, T_2)$ is 1 for $T_1 \cap T_2 \neq \emptyset$. Otherwise, we use partial and derived information, *e.g.*, for $T_1 = \{7\ October\ 1969\}$ (a Tuesday), $T_2 = \{October\}$, $T_3 = \{Tuesday\}$: $\sigma(T_1, T_2) = \frac{1}{10}$ and $\sigma(T_1, T_3) = \frac{1}{7}$ (the values were chosen by experiment, highlighting the difficulty of putting a numerical value on a comparison).

Finally, for the agentive similarity, we use the Levenshtein distance (the minimum number of characters to delete, insert or replace to convert one string into the other) on each pair of agents a_1, a_2 in A_1, A_2. As named entity (NE) extractors, such as GATE can make partial matches, or over-match, we make this distance "fuzzy" ($FL(a_1, a_2)$) by comparing the substrings [30], and define $\sigma(A_1, A_2)$ as $1 - \min\{FL(a_1, a_2) \mid \forall a_1 \in A_1, \forall a_2 \in A_2\}$ if over a certain threshold θ, and 0 otherwise (in practise, $\theta = 0.45$ gave the best results). For example, if $A_1 = \{Dr\ Dolittle\ PhD\}$ and $A_2 = \{Doolittle\}$, $\sigma(A_1, A_2)$ is $1 - FL(\cancel{Dr}\ Dolittle\ \cancel{PhD}, Do\cancel{o}little) = 1 - \frac{1}{8} = 0.875$. We do not try here to associate named entities such as *London / capital of England*, but the modularity of the system would allow this.

The quality of the events extracted (if they exist) weighs this similarity against the time taken to treat the document. More precisely, if the extraction of $\hat{E}_i$ from document d_i took time t with target event E_i, we define the quality Q of the extraction to be $\sigma(\hat{E}_i, E_i)/t$ if $\sigma(\hat{E}_i, E_i) \neq 0$, and $-t$ otherwise (no event extracted, or null similarity with the target event).

We thus formalise that the correct extraction of events is primordial, it must be done in a reasonable time, and the AI should rapidly detect if there is no interesting event to extract.

VI. TESTS WITH NUMERICAL FEEDBACK

The AI receives numerical feedback as defined in Section V on the quality of its results. In production, this is the distance between the event which it has extracted (if any), and the extraction as corrected by the human analyst, considered to be the "perfect" event which could be extracted from the document (if any). This is non-intrusive, relying on corrections which the analyst would need to make anyway, but requires fine-tuning of the definition of the similarity and its parameters, a cognitively difficult task. Note that the numerical reward given to the AI here is standard in RL but not natural in real life. In Section VII, we show the qualitative feedback results. Q-learning was run with standard parameters: exploration rate $\epsilon = 0.4$, then divided by 2 every 500 documents until $\epsilon = 0.1$, learning rate $\alpha = 0.2$. λ was 0 (no eligibility traces).

a) Training and production simulation: We first tested the quality of the policy learnt (1) after training repeatedly on a small set of documents, and (2) from scratch. We trained our AI on 100 documents taken from the GTD, 64% of which described "interesting" events (see below), 17% described other events, and 19% contained no extractable events. We treated these documents in the same order 30 times. The AI had the choice of six *gazetteers* (three lists of verbs and three of nouns). Two pairs of these gazetteers (*bombing* and *injure* verbs and nouns) contained lists of words likely to trigger the extraction of events of that type. Another pair of *dummy* gazetteers contained words not present in the GTD. We represented an analyst's preference by defining "interesting" events as $Bombings$ and emphasising semantic similarity (specifically $a = 20, b = c = d = 1$). The AI should thus favour the *bombing* gazetteers over the *injury* gazetteers, and ignore the *dummys* as they never lead to an extraction.

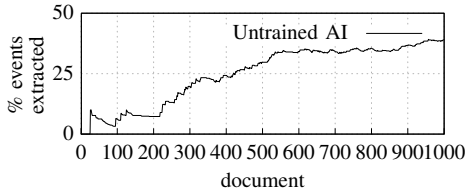


Fig. 3: Percentage events extracted in production simulation

We then simulated a production phase by running the policy learnt by this trained AI as well as an untrained AI on 1 000 documents chosen randomly from the GTD, "seeing" them for the first time. Only 29% of these contained $Bombing$ events, 7% other events, and 64% no extractable events.

The untrained AI (starting "from scratch") managed to extract 39% of the possible events from the 1 000 unknown documents, generalising well, its performance improving the more different documents it encountered (Figure 3).

The trained AI demonstrated an optimal policy, extracting 100% of the events, showing that it also generalises well, applying a learnt policy successfully to unknown documents. Both AIs learnt not only to order the chain, but to optimise it. They don't call the service *Geo*, and only use the *bombing* verb list. However, they unexpectedly preferred *injure* nouns to those of *bombing*. On investigation, we found that the GATE service used relied on verbs for extraction, ignoring the nouns. This shows that the AI was able to discover strategies which were not clear, even to the expert calibrating the chain.

b) Performance testing: We now assess how fast an untrained AI can learn an "expert-like" policy (able to extract 100% of the events), and how well this policy performs (how similar the events extracted are to the targets). We treated a set of 63 documents from the GTD in the same order repeatedly (in *rounds*). We expanded the action space by giving the AI the choice of ten *gazetteers* (five lists of verbs and five of nouns): three single-event pairs (*bombing*, *shooting* and *injure*), one *dummy* pair, and one *mixed* pair containing a mix of words likely to result in the detection of several types of events. We sought to verify that the AI would learn to use the *mixed* gazetteers rather than the single-event or *dummy* gazetteers, and that it was not sensitive to a larger action space.

We first trained the AI with feedback given for each document. It turned out that after only 1 008 documents (that is, 16 rounds), the learnt policy was able to extract 100% of the events. More importantly, the quality of the extraction with this policy is on a par with that of the expert chain (Figure 4a).

Naturally the analyst is not available *24×7* to consult each document as it is treated. We therefore ran a similar experiment, but giving feedback on all extractions once all 63 documents had been treated, hence preventing the AI from learning during a round. The AI was understandably slightly slower to learn, but learnt an optimal policy after only 1 260 documents (20 rounds), and again the quality of this policy is on a par with the expert chain (Figure 4b).

Finally, the analysts are not dedicated trainers, and will not correct all extractions. To simulate this, we only gave feedback after each extraction with a probability of 10% (otherwise the AI received no feedback at all for this extraction). Even with such sporadic feedback, the AI managed to extract 50% of the possible events after only 1 890 documents (30 rounds), and 100% after 5 103 documents (81 rounds). The quality of these two policies is depicted in Figures 4c,4d, and suggests that the abilities to extract events and to extract *correct* events increase together. Note that the lower quality (compared to the expert chain) that we observe in the latter plot is due to the processing time, and not the similarity with the target event.

VII. TESTS WITH INTUITIVE FEEDBACK

We performed a comprehensive set of experiments, with varying parameters γ (attenuation parameter, see Definition 2), ϵ (exploration rate), and λ (decay parameter for eligibility traces, see Section III). For each of these parameter settings, we compared the performance of the expert chain and those of three AIs learning "from scratch":

QL: Q-learning with numerical feedback as in Section VI but with varying parameter settings;

DOM: SSB Q-learning (Section III) with probabilistic dominance, that is, purely ordinal feedback, $\phi \in \{-1, 0, 1\}$;

MAG: SSB Q-learning expressing preferences of different magnitudes, $\phi \in \{-1000, -100, -10, 0, 10, 100, 1000\}$.

Recall that SSB Q-learning requires feedback only about the relative quality of two extractions. We therefore defined the feedback given to DOM in our experiments as:

$f \succ_{\mathrm{DOM}} f' \Leftrightarrow \phi_{\mathrm{DOM}}(f, f') = 1$ iff (i) treatment f extracted an event and f' did not, or (ii) neither or both extracted an event, and f was faster than f';

$f \sim_{\mathrm{DOM}} f' \Leftrightarrow \phi_{\mathrm{DOM}}(f, f') = 0$ iff both took approximately the same time, *i.e.*, the treatments' total times were within $margin$ (set to 5) seconds of each other.

We thus encouraged the AI to extract events first, and to do so fast, or to recognise quickly that there are no events to extract. This relies on the correlation, demonstrated in Section VI for Q-learning, between the ability to extract any event, and to extract the correct events. DOM also demonstrates this correlation. Obviously, this preference relation could be completed with additional information over the obtained results, such as the perceived quality of the extraction, *etc.*

Intuitively, MAG feedback is a middle ground between QL and DOM, which can be seen as a weighted form of probabilistic dominance. Yet such feedback remains quite natural. We emphasised the importance of extracting events, as compared to the importance of extracting the exact target events, in turn compared to the importance of running fast:

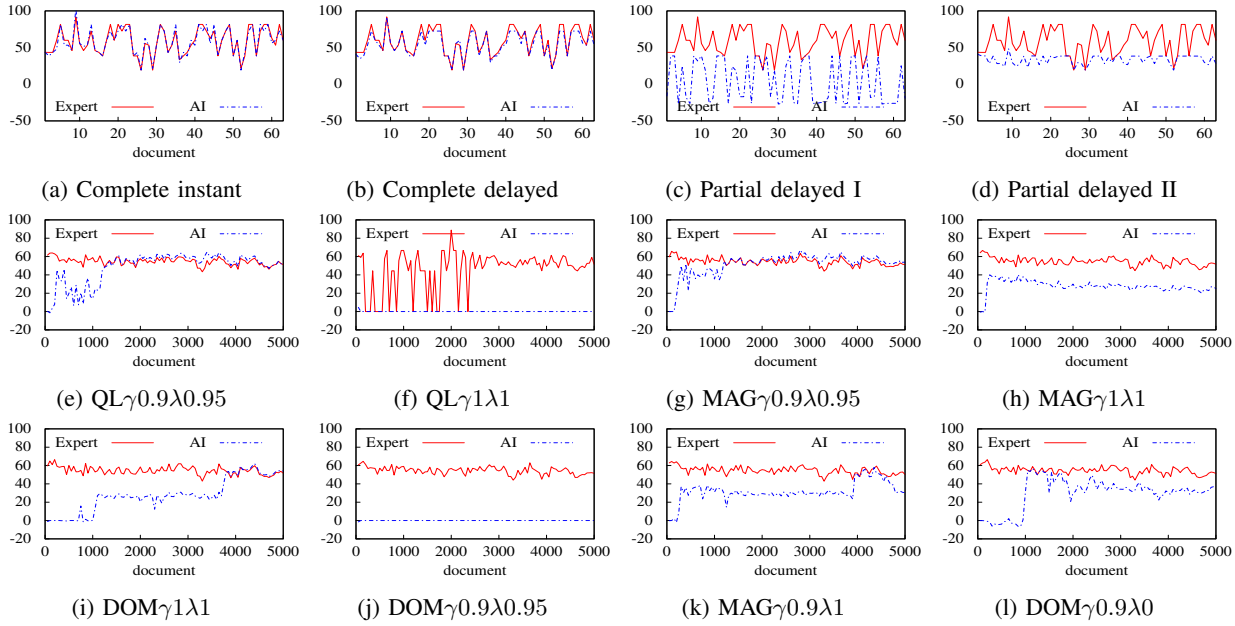
$\phi_{\mathrm{MAG}}(f, f') = 1000$ iff f extracted an event, f' did not;

$\phi_{\mathrm{MAG}}(f, f') = 100$ iff both extracted an event but f extracted an event of higher quality (Section V-A) than f';

$\phi_{\mathrm{MAG}}(f, f') = 10$ iff extractions were of similar quality or neither extracted, but f was faster by $margin$ seconds;

$\phi_{\mathrm{MAG}}(f, f') = 0$ iff f and f' were incomparable with respect to the previous conditions.

We expect QL to be more effective than MAG, and MAG to



(a) Complete instant (b) Complete delayed (c) Partial delayed I (d) Partial delayed II

(e) QL$\gamma 0.9\lambda 0.95$ (f) QL$\gamma 1\lambda 1$ (g) MAG$\gamma 0.9\lambda 0.95$ (h) MAG$\gamma 1\lambda 1$

(i) DOM$\gamma 1\lambda 1$ (j) DOM$\gamma 0.9\lambda 0.95$ (k) MAG$\gamma 0.9\lambda 1$ (l) DOM$\gamma 0.9\lambda 0$

Fig. 4: Quality of the policy learnt with (a) complete instant feedback; (b) complete delayed feedback; (c) partial delayed feedback (at 50% extraction); (d) partial delayed feedback; (e) QL best results; (f) QL worst results; (g) MAG best results; (h) MAG worst results; (i) DOM best results; (j) DOM worst results; (k) and (l) MAG and DOM respectively quality drop

be more effective than DOM, given the amount of information they receive. We demonstrate, however, that MAG and DOM are still perfectly realistic approaches in an industrial setting.

We ran the AIs, initially untrained, on a set of 5 000 GTD documents (presented in the same order, and only seen once in all experiments). The actions were chosen from ten gazetteers, the services *Tika, NGramJ, GATE, Geo*, and *STOP*. We measured the quality of the treatment of each document as described in Section V for all approaches. Obviously, as the AIs learned from scratch, the policy will initially be poor. We are interested in how fast a "good" policy is learnt (after seeing how many documents), and how good this policy is.

Both Q-learning and SSB Q-learning were run with α set as in [23], *i.e.* decreasing as the number of visits to the current state/action pair grows. We varied the other parameters to get a comprehensive set of results and measure the robustness to parameter choice.

We ran the following 36 combinations of tests:
– algorithms QL, DOM, and MAG,
– EG parameter ϵ divided by 2 after 2500 or 1000 documents,
– $\gamma = 0.9$ and $\gamma = 1$,
– $\lambda = 0$, $\lambda = 0.95$, and $\lambda = 1.0$.
As we cannot show all the results here, we've chosen interesting and representative plots. We plot only the extraction quality where the AI "exploited" *i.e.*, followed its "best" policy for the whole treatment (blue dotted line), against that of the expert chain (red solid line), which is why the red line varies between plots. For readability, we smooth the curve, taking averages on sets of 50 documents in the same order as they are treated.

QL gave excellent results, but proved quite sensitive to changing parameters. $\gamma = 0.9$ gave excellent results with $\lambda = 0.95$ (Figure 4e) where the AI learns a "good enough" policy after only 250 documents (the events are extracted correctly but it takes a few seconds longer than the expert chain), and a optimal one after 1 200 documents. $\gamma = 0.9$ also gave very good results with $\lambda = 0$ (not shown). With $\gamma = 0.9$ and $\lambda = 1$, however, the results were mediocre, and $\gamma = 1$ (Figure 4f) gave very bad results regardless of λ: the AI learns to STOP very early, suggesting a risk-adverse behaviour.

DOM, like QL, was sensitive to the choice of γ and λ. With $\gamma = 0.9$, the results were very bad for $\lambda = 1$ (not shown) and $\lambda = 0.95$ (Figure 4j). However, reasonably good results (not shown) were achieved with $\gamma = 0.9, \lambda = 0$ for the longer ϵ reduction strategy, and with $\gamma = 0.9, \lambda = 0$ and $\gamma = 0.9, \lambda = 0.95$ for the shorter strategy. With $\gamma = 1$, the results were good to excellent, and Figure 4i shows the best results for $\gamma = 1, \lambda = 1$, which learns a "good enough" policy after 1 000 documents and stabilises with an optimal policy after 3 750 documents (verified on a further 5 000 (not shown)).

MAG proved robust to parameter change, quickly learning a "good enough" policy in every case (*e.g.* Figure 4g).

We see a slight improvement in all results with a faster reduction in ϵ (every 1 000 documents vs every 2 500), *i.e.*, with less exploration overall. Finally, SSB Q-Learning sometimes degrades after learning an optimal policy. The events are still extracted, but it starts taking too long (Figures 4k, 4l). The AI learns that passing through GATE from a given state gives a good reward, and if γ and λ are not correctly set, although it takes longer, it starts to prefer this action to stopping.

In summary QL can give excellent results, but is sensitive to parameter variation and depends on numerical feedback. At the other extreme, DOM only requires purely ordinal feedback,

and yet with the correct parameters is able to learn expert-like policies. MAG offers a good middle ground: it is robust to parameter choice, uses mostly intuitive feedback, yet still learns a good to optimal policy very quickly. It is therefore viable to automatically improve a document treatment chain, and even to learn one from scratch, in settings where very little or no numerical information is given.

VIII. CONCLUSION AND FUTURE WORK

We modelled a document treatment chain as a Markov Decision Process, and solved it using reinforcement learning. Our approach could be applied to any treatment, for instance, to the process of object recognition in an image. Many detection devices, and numerous algorithms exist for this, and it is not yet clear how best to combine them.

We developed an application *BIMBO* (Benefiting from Intelligent and Measurable Behaviour Optimisation) into which we can "plug" different algorithms, web services and models to measure their impact on the learning. We established that our approach gave good results with sporadic numerical feedback. We then integrated a reward function formalising naturally expressed user preferences, demonstrating that this still gives good results while requiring much less cognitive effort to define the feedback. We thus showed that it is possible to have a self-improving treatment chain, which does not require intervention or tuning by a human user, and which collects its feedback in a non-intrusive manner. Our work also evaluates, in an industrial context, the applicability of various algorithms and the impact of changing parameters for important RL approaches, which is of independent interest.

The sets of states and actions in this article were chosen to demonstrate the validity of our approach. The next step is to make the chain more complex, by adding alternative services (*e.g.* translation), expanding the state set (*e.g.* complete list of languages), and introducing a wider range of input. We aim to show that the system is capable of building different chains for different types of document. Even with this larger state / action space, the calculation time should not be an obstacle with a *Q-learning* type algorithm, where the calculations are instantaneous at each time step.

REFERENCES

[1] M. Ogrodniczuk and A. Przepiórkowski, "Linguistic Processing Chains as Web Services: Initial Linguistic Considerations," in *Proceedings of the Workshop on Web Services and Processing Pipelines in HLT: Tool Evaluation, LR Production and Validation (WSPP 2010) at the Language Resources and Evaluation Conference (LREC 2010)*, 2010, pp. 1–7.

[2] I. Bratko and D. Suc, "Learning qualitative models," *Artificial Intelligence*, vol. 24, no. 4, p. 107, 2003.

[3] F. Saïs, L. Serrano, R. Khefifi, and F. Scharffe, "SOS-DLWD 2013," 2013.

[4] M. P. Fromherz, D. G. Bobrow, and J. De Kleer, "Model-based computing for design and control of reconfigurable systems," *AI magazine*, vol. 24, no. 4, p. 120, 2003.

[5] F. Rodrigues, N. Oliveira, and L. Barbosa, "Towards an engine for coordination-based architectural reconfigurations," *Computer Science and Information Systems*, vol. 12, no. 2, pp. 607–634, 2015.

[6] J. Doucy, H. Abdulrab, P. Giroux, and J.-P. Kotowicz, "Méthodologie pour l'orchestration sémantique de services dans le domaine de la fouille de documents multimédia," 2008.

[7] W. B. Knox and P. Stone, "Framing reinforcement learning from human reward: Reward positivity, temporal discounting, episodicity, and performance," *Artificial Intelligence*, vol. 225, pp. 24–50, 2015.

[8] R. Loftin, B. Peng, J. MacGlashan, M. L. Littman, M. E. Taylor, J. Huang, and D. L. Roberts, "Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning," *Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 1, pp. 30–59, 2016-01.

[9] A. Azaria, Z. Rabinovich, S. Kraus, C. V. Goldman, and Y. Gal, "Strategic advice provision in repeated human-agent interactions," *Institute for Advanced Computer Studies University of Maryland*, vol. 1500, p. 20742, 2012.

[10] A. B. Karami, K. Sehaba, and B. Encelle, "Apprentissage de connaissances d'adaptation à partir des feedbacks des utilisateurs," in *25es Journées francophones d'Ingénierie des Connaissances*, May 2014, pp. 125–136.

[11] R. Akrour, M. Schoenauer, and M. Sebag, "Preference-based policy learning," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 12–27.

[12] WebLab, "WebLab wiki," http://weblab-project.org/, 2015, accessed: 2015-03-17.

[13] Tika, "Apache Tika - a content analysis toolkit," http://tika.apache.org/, 2015, accessed: 2015-02-18.

[14] NGramJ, "NGramJ, smart scanning for document properties," http://ngramj.sourceforge.net/, 2015, accessed: 2015-02-18.

[15] GATE, "GATE Information Extraction," https://gate.ac.uk/ie/, 2016, accessed: 2016-06-20.

[16] L. Serrano, "Vers une capitalisation des connaissances orientée utilisateur: extraction et structuration automatiques de l'information issue de sources ouvertes," Ph.D. dissertation, Universté de Caen, 2014.

[17] W. R. van Hage, V. Malaisé, R. Segers, L. Hollink, and G. Schreiber, "Design and use of the Simple Event Model (SEM)," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, no. 2, pp. 128–136, Jul. 2011.

[18] Geonames, "Geonames," http://www.geonames.org/, 2015, accessed: 2015-03-17.

[19] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming 1st*. John Wiley & Sons, Inc. New York, NY, USA, 1994.

[20] R. J. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[21] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 4, no. 1, pp. 1–103, 2010.

[22] C. J. C. H. Watkins, "Learning From Delayed Rewards," Ph.D. dissertation, Kings College, May 1989.

[23] H. Gilbert, B. Zanuttini, P. Viappiani, P. Weng, and E. Nicart, "Model-free reinforcement learning with skew-symmetric bilinear utilities," 2015, accepted at UAI16. Available at http://zanuttini.users.greyc.fr/research/ssbQLearning.pdf.

[24] R. I. Brafman and M. Tennenholtz, "R-max-a general polynomial time algorithm for near-optimal reinforcement learning," *The Journal of Machine Learning Research*, vol. 3, pp. 213–231, 2003.

[25] K. Rao and S. Whiteson, "V-MAX: A General Polynomial Time Algorithm for Probably Approximately Correct Reinforcement Learning," Ph.D. dissertation, Amsterdam, Sep. 2011.

[26] R. Busa-Fekete, B. Szörényi, P. Weng, W. Cheng, and E. Hüllermeier, "Preference-based reinforcement learning: evolutionary direct policy search using a preference-based racing algorithm," *Machine Learning*, vol. 97, no. 3, pp. 327–351, Dec. 2014.

[27] P. C. Fishburn, "SSB utility theory: an economic perspective," *Mathematical Social Sciences*, vol. 8, no. 1, pp. 63 – 94, 1984. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0165489684900611

[28] Camel, "Apache Camel," http://camel.apache.org/, 2015, accessed: 2015-03-17.

[29] G. LaFree, "The Global Terrorism Database: Accomplishments and Challenges | LaFree | Perspectives on Terrorism," *Perspectives on Terror*, vol. 4, no. 1, 2010.

[30] R. Ginstrom, "The GITS Blog: Fuzzy substring matching with Levenshtein distance in Python," http://ginstrom.com/, 2007, accessed: 2014-08-19.