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[1,2], Bruno Zanuttini[1], Hugo Gilbert[3], Bruno Grilhères[4], Fredéric Praca[2]

[1] Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France
first.last-name@unicaen.fr

[2] Cordon Electronics DS2i, 27000 Val de Reuil, France first.last-name@cordonweb.com

[3] LIP6, Paris, France first.last-name@lip6.fr

[4] 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 custom-made chains "on the fly", and to continuously improve them. We build a platform, BIMBO (Benefiting from Intelligent and Measurable Behaviour Optimisation) which enables us to measure the impact on the learning of various models, algorithms, parameters, *etc.* We apply this in an industrial setting, specifically to a 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 user effort and tuning, to partially and even fully qualitative feedback, which is much more intuitive, and demands little to no user intervention. We carry out experiments, first with numerical feedback, then demonstrate that intuitive feedback still allows the agent to learn effectively.

Keywords : Artificial intelligence, Reinforcement learning, Extraction and knowledge management, Man-machine interaction, Open source intelligence (OSINT)

1 Introduction

Our research is driven by an industrial need, expressed by the *Open Source INTelligence* (OSINT) community. In the past, the OSINT analyst's task was finding hidden information. Now, faced with ever-increasing volumes of web pages and other open source documents, the challenge is to find pertinent information from the huge wealth of data available. Many specialised treatment chains have been developed to ease this task (for example, Ogrodniczuk & Przepiórkowski (2010) give an overview of some processing chains that are provided as web services).

We tackle the generic problem of the improvement of such a chain, specifically the extraction of events for OSINT assessment and surveillance. In this application, documents, sourced for the most part from the web, are inserted continuously into a treatment chain which aims to extract events (*e.g.*, terrorist attacks) and their characteristics (date, place, agents, *etc.*).

In such applications, it is clear that the extraction cannot be perfect. Firstly, because of the huge diversity of documents on the Web, there cannot be a single optimal chain for all documents. Secondly, imagine, for example, a school blog recounting "the *bombardment* of a gold *target* by ions during an *atomic* physics demonstration at the school science fair". The treatment chain could erroneously recognise and extract an atomic bomb attack. Then, the need to treat documents coming from all over the world in diverse languages entails the use of dictionaries whose quality is variable. For these reasons, the analysts are typically forced to correct *a posteriori* the events extracted, a distracting, 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 over time without the intervention of an expert in consultation with the analysts.

Our objective is to remedy this situation 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) obtained as the analysts consult the system output. In other words, we allow the chain to "learn" from its mistakes. For instance, that if the document is a well-written scientific article in French, a translation to English before the extraction works well to make the most of the wealth of English extraction rules available, but for blogs containing slang, a direct extraction is preferable as the dictionaries are insufficient, and a translation only introduces more noise. The services, or building blocks for the chain, thus remain a "black box" for the analyst allowing her to distance herself completely from the extraction process.

We need to collect the feedback non-intrusively, that is, in a way that is invisible to the analyst without impacting on her work. A user's expertise can be modelled by tracing his actions (Bratko & Suc, 2003). In production, these action traces could be captured through the graphical interface which gives a summary of the events extracted. We can then base the feedback on the analyst-system interactions that currently occur. For example, an event might be *corrected*, in which case we can deduce an explicit feedback based on the corrections that the analyst might make, that is, a distance between the event as corrected and that which was extracted. Indeed, our first set of tests in Section 9 simulate this, in that the agent receives a numerical feedback as an automatically generated judgement on the quality of extractions.

The difficulty is that an event may also be *not extracted*, *extracted in too long a time*, *etc.* To interpret these "non-actions" as feedback, we must first answer questions such as "Is it preferable to extract something erroneously, or to miss an extraction ?", "Should I sacrifice speed for quality, or vice versa ?".

These questions cannot be answered naturally with a numerical response ; it is not easy to determine that "A partial, but fast extraction is worth 10.5 times more than a slow but complete extraction". It is therefore essential that we progress to giving intuitive feedback (Section 10), based on naturally expressed user preferences, such as "I prefer an extraction to no extraction" and "I prefer it to be as fast as possible". The requirements can thus be gathered very easily, and can be changed if necessary in an intuitive fashion.

2 Contribution

We formalise the improvement of a document treatment chain as a reinforcement learning (RL) problem, and the treatment chain itself as a Markov Decision Process (MDP) (Section 6). We investigate three different settings. In the first, the agent is rewarded with numerical feedback on the quality of the treatment, which is very informative but very time-consuming for the human user. 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 user interaction.

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 a treatment chain which extracts events from web pages and other open-source documents.

We carry out experiments, first with numeric, potentially sporadic feedback data (Section 9), which show that automatically learning how to chain and parametrise services is perfectly feasible. Then we carry out experiments with intuitive feedback (Section 10), where even though the roughness of feedback makes learning less effective, it remains perfectly feasible, while requiring little or no tuning.

3 Related work

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 to attain a desired result, for instance, is not a new concept. Service compositions can be computed, for example, to help the user to fill in forms (Saïs *et al.*, 2013), and the physical characteristics of documents are used to successfully construct adaptive, modular chains for Xerox photocopiers in real time (Fromherz *et al.*, 2003). However, 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 feedback into the process to improve it next time.

The user can reconfigure the chain herself using a dedicated language, *e.g.*, ReCooPLa (Rodrigues *et al.*, 2015), or by choosing a service from directories of the same type (Doucy *et al.*, 2008). The chain will not adapt dynamically without user intervention, however, and the user must have system expertise to appreciate the consequences of her choices.

Using human feedback, both explicit and implicit for model-based learning has also been explored. An agent can learn sequential tasks using continuous human evaluation of its actions, for instance, the trainer's positivity can influence the agent to learn trainer-dependent behaviour (Knox & Stone, 2015) or implicit, non-numeric feedback can be provided by inference on the observation of the trainers' actions (Loftin *et al.*, 2016 01). Models of the user's behaviour (Azaria *et al.*, 2012) and preferences (Karami *et al.*, 2014) 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 (Akrour *et al.*, 2011). Although these systems react well to changing users and their preferences, they are not adapted to our problem, as they require continuous user interaction at each step to confirm or contradict their choice. This would be prohibitively expensive in our case as our users are not dedicated trainers and the human effort needed to yield a competent policy can be considerable.

We currently treat the services as "black boxes", aiming to improve the way they are combined. We note that information can be gleaned about the quality of each service (Caron *et al.*, 2014) and potential degradation in the chain (Amann *et al.*, 2013), and that individual services can be automatically updated based on user edits (Formiga *et al.*, 2015). We can imagine using a combination of techniques such as these in parallel with our approach.

4 The WebLab Platform

Our industrial application uses the open-source platform WebLab (2015) 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.

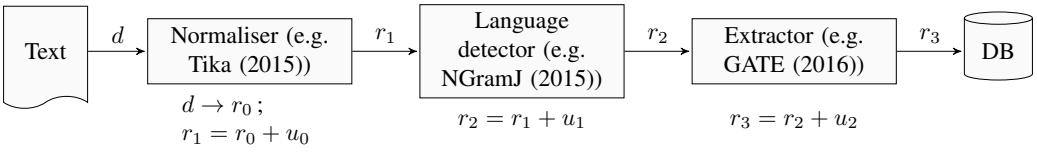


FIGURE 1 – A simple but typical Weblab chain: a document, d, is converted into an XML resource, r_0, and annotations, u_j, are added by each service. Finally the results are stored in a database.

Example 1
Consider the following document:

> *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.*

The simplified XML resource in Figure 2 is produced after 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 first define these formally. We chose a formalisation in the ontology *WOOKIE* (Serrano, 2014), but any other definition of an event (e.g. van Hage *et al.* (2011)) could be used.

```
<resource type="Document" uri="weblab:aaa">
  <annotation uri="weblab:aaa#a0">
    <wp:originalContent resource="file:weblab.content"/>
    <wp:originalFileSize>255</wp:originalFileSize>
    <dc:source>documents/event.txt</dc:source>
    <wp:originalFileName>event.txt</wp:originalFileName>
    <dc:modified>2015-02-14T19:52:21+0100</dc:modified>
    <wp:collected>2015-02-10T00:11:00+0200</wp:collected>
  </annotation>
  <annotation uri="weblab:aaa#a1">
    <wp:isProducedBy resource="weblab:tika"/>
    <dc:format>text/plain</dc:format>
  </annotation>
  <annotation uri="weblab:aaa#a2">
    <wp:isProducedBy resource="weblab:ngramj"/>
    <dc:language>en</dc:language>
  </annotation>
  <mediaUnit type="wl:Text" uri="weblab:aaa#0">
    <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>
  </mediaUnit>
</resource>

```

FIGURE 2 – Simplified XML of a WebLab resource mid-chain.

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

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

Even though the definition is general, in this article we use precise domains: $\mathbb{C}$ is a fixed and finite set of elements in *WOOKIE* ; $\mathbb{T}$ is the set of all relative and absolute "dates", such as "tomorrow", "2001", "2001/9/11" ; $\mathbb{S}$ is the set of entities defined in Geonames (2015) ; finally, $\mathbb{A}$ is the infinite set of all extractable participants, seen as strings.

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

Building an efficient treatment chain depends not only on the services chosen but their parameters. For instance, the extractor GATE is parametrised by *gazetteers*, which are lists of nouns and verbs triggering the detection of a specific type of event ; the verb gazetteer for a *bombing* event would typically contain *explode*, *detonate*, *etc.* Hence we see GATE as a *parametrised service*: choosing the correct gazetteers for it to use is part of building an efficient treatment chain.

5 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 French, send to extractor 1, and to service 2 otherwise).

Despite their expertise, it is impossible to create the perfect universal chain, as the treatment of open source documents gathered from the web 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, imagine that the analyst wants information on aerial bombings. We saw in Section 1 that it is impossible to say with certainty that the word "bombardment" refers to this. The analyst can see that the

page is a school blog only once the event has been extracted. Even from a specialist web page, it is possible that the word refers to the incessant questions of the journalists, and not an attack. Maybe the synonym "shelling" was used, and the event is not recognised. With these uncertainties, the experts who configure the chain try to envisage the most common situations. It is inconceivable that they construct individual chains by examining each source document.

Formalising this treatment chain and improving the event extraction is therefore not a trivial problem. The only information known before starting the treatment chain consists of the available services, their parameters, and the potential characteristics of the XML resource and system (see Section 4). 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). We give an overview here, but recommend reading Sutton & Barto (1998) for more details.

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 a Markov Decision Process (MDP, Puterman (1994)), 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 *is a (here finite) set of possible states in the environment ;*
- A *is a (here finite) set of actions (available to the agent) ;*
- P *is 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 *is a reward function, defined on the states ;* $R(s)$ *is the reward obtained by the agent when it reaches state* s *;*
- $\gamma \in [0, 1]$ *is an discount 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 Szepesvári (2010)).

Numerous algorithms exist to solve RL problems. In our experiments we first use standard *Q-learning* (Watkins, 1989) with numerical feedback. Tests were also carried out using RMax (Brafman & Tennenholtz, 2003) and VMax (Rao & Whiteson, 2011), 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. Then for the qualitative setting, which standard approaches cannot handle, we used a recently proposed algorithm, called *SSB Q-learning* (Gilbert *et al.*, 2015). We give brief overviews of both below to keep this paper self-contained, but encourage the reader to read the original articles. Nonetheless, our contribution includes modelling the improvement of a treatment chain as an RL problem, and in practise, any RL algorithm could be used. Note that we present here the general algorithms. Our formalisation of user feedback for our application is detailed in Sections 9–10.

5.1 Q-learning

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*). Then, upon reception of 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 experience $(r_{t+1} + \gamma\, max(\dots))$ 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.

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 $r_{t+1} = \mathcal{R}(s_{t+1})$

  $\hat{Q}_{t+1}(s_t, a_t) \leftarrow \hat{Q}_t(s_t, a_t) + \alpha_t(s_t, a_t)(r_{t+1} + \gamma\, max_b\{\hat{Q}_t(s_{t+1}, b)\} - \hat{Q}_t(s_t, a_t))$

5.2 SSB Q-learning

Our approach works very well with numerical feedback (Section 9), but very few humans could say "A false extraction is worth 10.5 times more than a missed extraction, which is worth 7.9 times more than an extraction taking 90 seconds". They would more likely reply "I prefer a false extraction to a missed extraction, as I don't want to lose information, and I'd rather it were as fast as possible". 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 as fast as possible". This preferential information can be expressed as a partial order over possible results achieved $(f_1, \dots, f_k)$ (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.*

Solving RL problems based on this kind of ordinal/preferential information is the problem raised by preference-based RL (Akrour *et al.*, 2012; Fürnkranz *et al.*, 2012; Wilson *et al.*, 2012; Wirth & Fürnkranz, 2013; Wirth & Neumann, 2015). In this context, maximizing the expectancy of cumulated rewards cannot be done directly as the numerical values of those rewards are not available. One approach would try to recover a consistent reward function from the given preferential information. This approach has the disadvantage of introducing some preferential information that was not expressed by the decision maker. Thus a second approach aims at using decision criteria adapted to ordinal information. One criterion of this kind is called *probabilistic dominance* (Busa-Fekete *et al.*, 2014) and aims to maximize the probability of yielding a preferred outcome. More formally, let π_1 and π_2 be two policies and let F_1, F_2 be two random variables on $(f_1, \dots, f_k)$ where $\mathbb{P}(F_i = f_j)$ is given by the probability of π_i achieving result f_j, and $F_x \succeq F_y$ indicates that the result achieved by π_x is preferred or equal to that achieved by π_y. Then according to the probabilistic dominance criterion:

$$\pi_1 \succeq \pi_2 \Leftrightarrow \mathbb{P}(F_1 \succeq F_2) \geq \mathbb{P}(F_2 \succeq F_1)$$

In words, policy π_1 is preferred to π_2 if the probability of π_1 doing better than π_2 is greater than the converse.

In fact probabilistic dominance is a particular instance of a wide class of criteria called Skew Symmetric Bilinear (SSB) utility functions (Fishburn, 1984). Given ϕ, an SSB utility function, $\phi(\pi, \pi') > 0$ (resp. $\phi(\pi, \pi') < 0$) expresses the fact that the user prefers π to π' (resp. π' to π) whereas $\phi(\pi, \pi') = 0$ expresses an indifference. Probabilistic dominance is the case when ϕ is always 1, 0, or -1, hence requires only purely ordinal feedback from the user.

To achieve reinforcement learning with this optimisation criterion, we use the recently proposed algorithm called *SSB Q-learning* (Gilbert *et al.*, 2015), which proposes a solution in the general setting of SSB optimisation criteria. Like Q-learning, *SSB Q-learning* 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 ϕ 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 $\mathcal{R}_{\mathbf{p}_t}(s_{t+1})$). Intuitively, the algorithm continuously tries to act better, according to ϕ, than it has so far (Algorithm 2).

Algorithm 2: SSB Q-learning (Gilbert *et al.*, 2015)

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* (Sutton & Barto, 1998, 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.

6 Chain improvement as reinforcement learning

Given that it is impossible to create a unique chain capable of perfectly treating every type of document, the ideal would be to have a tailor-made chain for each one. Part of our contribution is to model 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. Neither can the rewards be easily specified, especially as 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 original document and of the resource and system at a given moment (Section 7). 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 time will be longer, and due to the massive volumes being treated in production (potentially all possible documents from the web) 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:
– if the current state has the value "true" for the descriptor "typeExtracted" and the value "school blog" for the descriptor "type", then the best action to perform consists of stopping the treatment (useless to try to extract bombings from school blogs) ;
– if the type is extracted and is not "school blog", the best action consists of passing the document to an extraction service, using the word "bombardment" among the trigger words ;
– otherwise, the best action consists of passing the document to a type-recognition service.
Finally, the rewards are defined by the user in three ways. The first is a quantitative value based on the corrections made to the extraction (if any) by the analyst and the time spent in treating a document (Section 9). The second is a qualitative reward based on the user's purely ordinal preference on the results, and the third, a middle-ground between the first two, is a weighted ordinal preference (Section 10). The rewards $r(s)$ are thus given to the system only for the final states of a treatment. For this, we can use the fact that the analysts consult the summaries produced by the system, which show the events extracted and link to the original source documents. The actions of the analyst, such as correcting the extracted information or consulting a

document without correcting it, furnish indirectly a feedback on the treatment the document received.

7 Experimental framework

To assess the validity of our approach, we ran experiments with our industrial application, using a simple, but typical chain, as described in Section 4, to which we added the possibility of choosing a "useless" service *Geo*. The chain is written as a Camel (2015) 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 (Cunningham *et al.*, 2014) 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. Note that the services are "black boxes" to the algorithms and *BIMBO*, so that knowledge of, *e.g.*, their WSDLs is unnecessary. [1]
A state is represented by an attribution of the following characteristics:

— the information already extracted (if any): *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) ;

— the parameters chosen: *nounGazetteer* (the noun gazetteer currently chosen, if any), similarly *verbGazetteer* ;

— the system state: *nbServices* $\in \{0-5, 6-20, 21+\}$ (the number of services through which the current document has already passed) ; *timeTakenSoFar* (seconds since the current document treatment started in $timeInterval$ (parametrizable) steps.

This restricted set of states and actions was chosen to demonstrate the validity of our approach. The next step is to take into account more characteristics (*e.g.*, source website type, complete list of languages), services (*e.g.*, web-crawling, translation) and their parameters.

Given the cost of an analyst's time, we simulate the feedback in our experiments. We use an open-source corpus in which the events are already known: the *Global Terrorism Database* (GTD LaFree (2010)), consisting of details of over $125\,000$ worldwide terrorist events from 1970 to 2014. We constructed a set of documents summarising these events $\{d_1 \dots d_N\}$ from which a perfect chain could extract perfect events $E_1, \dots, E_N$, respectively (as in Examples 1 and 2), if it had infinite time, and access to an infinite number of perfectly parametrised web services. Obviously such a chain does not exist in real life, so we use the results from a production chain tuned by hand as a reference, referred to hereafter as the "expert" chain. We expect our AI to learn to construct chains that approach and eventually improve on the "expert" chain, in learning not only the correct order of the services and the best parameters, but also the fact that certain services (in our case, the geographical inference) and certain parameters (some GATE gazetteers) are not useful.

As we explained in Section 5, in our tests, the AI starts learning with no *a priori* knowledge of the documents or the users needs. We say that such an AI is "untrained". Once the AI has treated a certain number of documents, it will have learnt what it considers to be an optimal policy. We say that it is "trained".

Note that here we test the learning capacity of an initially untrained AI. In production, to avoid learning "from scratch", we would capitalise on existing expertise by initialising *BIMBO* with a policy based on that of an "expert" chain.

8 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.

Numerous methods have been proposed for measuring similarity (Pandit *et al.* (2011); Tversky & Gati (1978); Cohen *et al.* (2013); Dutkiewicz *et al.* (2013) to give but a few), but we wanted a measure taking into account the events' four dimensions, and capable of differentiating between different types of event, therefore we define:

1. Of course, any such extra information could be used by *BIMBO* to generalise over actions, for instance, or to build a prior model of the actions and states, but we leave this for future work.

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

We define the semantic (resp. geographical) similarity $\sigma_C(C_1, C_2)$ (resp. $\sigma_S(S_1, S_2)$) 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 $\sigma_C(C_1, C_2) = 1$.

The temporal similarity $\sigma_T(T_1, T_2)$ is 1 for $T_1 \cap T_2 \neq \emptyset$, but if there is no common element, we use partial and derived information (day, month, year, day of the week). For example, for $T_1 = \{7\ October\ 1969\}$, $T_2 = \{October\}$ and $T_3 = \{Tuesday\}$, $\sigma_T(T_1, T_2) = \frac{1}{10}$. $T_1 \cap T_3$ is $\emptyset$, but as 7^{th} October 1969 was a Tuesday, $\sigma_T(T_1, T_3) = \frac{1}{7}$.[2]

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 the 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 (Ginstrom, 2007), and define $\sigma_A(A_1, A_2)$ 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 $A_2 = \{Doolittle\}$, $\sigma_A(A_1, A_2)$ 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) takes into account this similarity, and 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 the fact that the correct extraction of events is primordial, and must be done in a reasonable time, and that the AI should rapidly detect if there is no interesting event to extract.

9 Experiments with numerical feedback

In these experiments, the AI receives numerical feedback as defined in Section 7 on the quality Q of its results. In production, this corresponds to rewarding it based on 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 indeed non-intrusive, as it relies 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 information given to the AI here is the same as that used for assessing the quality of the policy it has learnt, as is standard in RL but not natural in real life. In Section 10, we remove these requirements and report the results with qualitative feedback.

9.1 Training then testing with a given analyst preference

We first tested the quality of the policy learnt (1) after training repeatedly on a small set of documents, and (2) from scratch on a larger, randomly chosen set of documents.

(1) We trained our AI on 100 GTD documents treated in the same order 30 times. To avoid the AI running forever, if the treatment time for a document exceeded a threshold of 30 seconds, the chain was stopped after the current service finished, and any events extracted returned. The AI had the choice of the services or *STOP* (as in Section 7), and six *gazetteers* (three lists of verbs and three of nouns). Two pairs of 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 $Bombing$ events and emphasising semantic similarity (specifically $a = 20, b = c = d = 1$). We hoped the AI would take into account the analyst's preference by favouring the *bombing* gazetteers over the *injury* gazetteers, and ignoring the *dummy* ones as they never lead to an extraction. 64% of of the documents described these "interesting" $Bombing$ events, 17% other events, and 19% contained no extractable events.

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 $\alpha = 0.2$. For these experiments λ was 0 (no eligibility traces).

2. The values $\frac{1}{10}$ and $\frac{1}{7}$ were chosen by experiment, and highlight the difficulty of giving an exact numerical value to a comparison.

(2) We tested both an untrained AI and the policy learnt by this trained AI on 1 000 documents chosen randomly from the GTD, "seeing" them for the first time. Only 29% of these contained "interesting" $Bombing$ events, 7% other events, and 64% no extractable events. As a control, we also ran an "expert" chain which was tuned to extract only the $Bombing$ events.

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

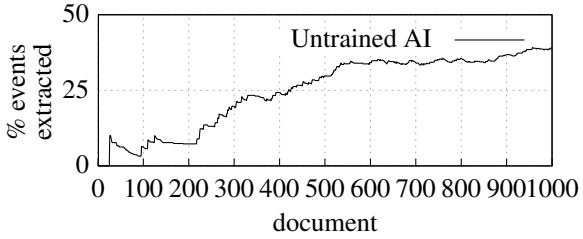


FIGURE 3 – Percentage events extracted by the untrained AI in production simulation

The trained AI demonstrated an "expert"-like policy, extracting 100% of the events, showing that it also generalises well, applying a learnt policy successfully to unknown documents. We also note that the AI learnt to order the chain, *e.g.*, in the state corresponding to a document with neither format nor language detected, and 0 seconds passed, the best action learnt was to pass the document to the service Tika. In the state corresponding to a document where at least one event is extracted, whatever the values of the other attributes, the AI stopped the treatment of that document.

As hoped, the AI also learnt to optimise the chain. It didn't call the service *Geo*, and only used the *bombing* verb list. However, it unexpectedly preferred *injure* nouns to those of *bombing*. On investigation, we found that the GATE service provided only uses the verbs for event 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.

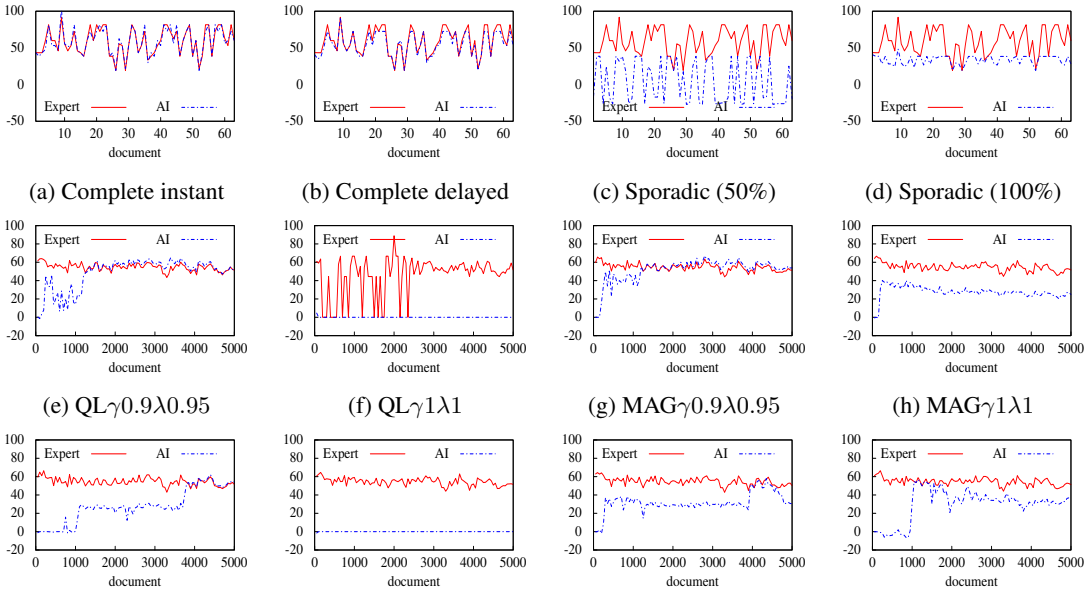
9.2 Performance testing

The previous set of experiments aimed to assess the ability of the AI to extract events from unknown documents. We ran a second set of experiments, to assess how fast an untrained AI could learn an "expert-like" policy (one able to extract 100% of the events), and how well this policy performed, so as to verify that the events extracted were indeed similar to the target ones. For this, we took a set of 63 documents from the GTD, all of which contained events extractable by the "expert" chain. We treated these documents in the same order repeatedly (in *rounds*). To investigate to what extent the AI was able to extract the *correct* events, especially by choosing the best gazetteers for GATE, we expanded the action space by giving the AI the choice of the services or *STOP* plus ten *gazetteers* (five lists of verbs and five of nouns): three pairs (*bombing*, *shooting* and *injure*) containing lists of words likely to lead to the detection of only one type of event ; one pair (*mixed*) containing a mix of words likely to result in the detection of several types of events ; and one *dummy* pair. We sought to verify that the AI would learn to use the *mixed* gazetteers rather than the unique event gazetteers, that it would avoid the *dummy* gazetteers, and that it was not sensitive to a larger action space.

Q-learning was run with parameters similar to those of the first experiments: $\epsilon = 0.4$, divided by 2 every 10 rounds until $\epsilon = 0.1$, $\alpha = 0.2$, and $\lambda = 0$. The timeout was set to 18 seconds.

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 was on a par with that of the "expert" chain, as shown in 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 only 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 "expert"-like policy after only 1 260 documents (20 rounds), and again the quality of this policy was on a par with that of the "expert" chain (Figure 4b).



(a) Complete instant (b) Complete delayed (c) Sporadic (50%) (d) Sporadic (100%)

(e) QLγ0.9λ0.95 (f) QLγ1λ1 (g) MAGγ0.9λ0.95 (h) MAGγ1λ1

(i) DOMγ1λ1 (j) DOMγ0.9λ0.95 (k) MAGγ0.9λ1 (l) DOMγ0.9λ0

FIGURE 4 – Quality of the policy learnt with (a) complete instant feedback (policy able to extract 100% events) ; (b) complete delayed feedback (able to extract 100% events) ; (c) sporadic feedback (able to extract 50% events) ; (d) sporadic feedback (able to extract 100% events) ; (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, showing the drop in quality

Finally, the analysts are not dedicated trainers, and will not correct all extractions. We therefore only gave feedback at the end of the round with a probability of 10% on each extraction (otherwise the AI received no feedback at all for that 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 ability to extract events and the ability 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.

10 Experiments with intuitive feedback

We performed a comprehensive set of experiments, with varying parameters γ (discount parameter, see Definition 2), ϵ (exploration rate), and λ (decay parameter for eligibility traces, see Section 5). For each setting of these parameters, 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 9 but with varying parameter settings, and eligibility traces ;
— **DOM:** SSB Q-learning as presented in Section 5, 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 only requires feedback on the relative quality of two extractions. We therefore defined the feedback given to DOM in our experiments as:

— $f \succ_{\text{DOM}} f' \Leftrightarrow \phi_{\text{DOM}}(f, f') = 1$ 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' ;
— $f \sim_{\text{DOM}} f' \Leftrightarrow \phi_{\text{DOM}}(f, f') = 0$ iff both took approximately the same time, *i.e.*, the treatments' total times were within $margin$ seconds (parametrisable) of each other.

We thus encouraged the AI, in an arguably very natural manner, 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 9 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.*

For the MAG type of feedback, we emphasised the importance of extracting events, compared to the importance of extracting the exact target events, in turn compared to the importance of running fast. Intuitively, it 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. Precisely, we define:

— $\phi_{\mathrm{MAG}}(f, f') = 1000$ iff f extracted an event and f' did not ;
— $\phi_{\mathrm{MAG}}(f, f') = 100$ iff both extracted an event but f extracted an event of higher quality (as measured in Section 8) than f' ;
— $\phi_{\mathrm{MAG}}(f, f') = 10$ iff both extracted an event of similar quality or neither extracted an event, 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.

Of course, we expect QL to be more effective than MAG, and MAG to be more effective than DOM, as QL receives much more information than MAG, and MAG receives more than DOM. We however demonstrate in this section that MAG and DOM are perfectly realistic approaches in an industrial setting.

To measure the performance of all approaches, we ran them, initially untrained, on a set of 5 000 GTD documents (presented in the same order, and only seen once in all experiments) from which the "expert" chain can extract events. The available actions were given by choices 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 7, and emphasise that the quality of the extraction was measured in the same manner for all approaches (including the "expert" chain), though the feedback given for the AIs to learn from was different. Obviously, as the AIs learned from scratch, we could not expect that the extraction was good from the start. Rather, we were interested in how fast a "good" policy was learnt (after seeing how many documents), and how good this policy was.

The stopping time $threshold$ was set to 10 seconds and $margin$ to 5. Both Q-learning and SSB Q-learning were run with α set as in Gilbert *et al.* (2015), *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 measured the robustness to parameter choice in the following 36 combinations of tests:

— algorithms QL, DOM, and MAG ;
— EG parameter ϵ divided by 2 after 2500 or after 1000 documents ;
— $\gamma = 0.9$ and $\gamma = 1$;
— $\lambda = 0$, $\lambda = 0.95$, and $\lambda = 1.0$.

10.1 Results

As we cannot show all 36 plots here, we give general comments about the results, showing interesting and representative curves. On each plot, we show the extraction quality of a particular approach (blue dotted line) against that of the "expert" chain (red solid line). We only consider the results for the documents for which the AI "exploited", *i.e.*, followed its "best" policy for the whole treatment (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. For both ϵ reduction strategies, $\gamma = 0.9$ gave excellent results with $\lambda = 0.95$ (Figure 4e) where the AI learnt a "good enough" policy after only 250 documents, and a "expert"-like 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 with $\gamma = 1$ (for example, Figure 4f), the results were very bad regardless of λ: the AI learnt 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 learnt a "good enough" policy after 1 000 documents and stabilises with an "expert"-

like policy after 3 750. Note that we continued this run on a further 5 000 documents (not shown), and it stayed "expert"-like.

MAG proved robust to changes in parameters, quickly learning a "good enough" policy in every case. $\gamma = 0.9, \lambda = 0.95$ (Figure 4g) is particularly good, learning a "good enough" policy almost straight away, and improving rapidly by speeding up to out-perform the "expert" chain after 3 000 documents.

We also noticed that the AI was capable of improving on a chain which was already good (where the events were extracted correctly but which took a few seconds longer than the "expert" chain). It learnt to speed up, eventually matching the "expert" chain.

Finally, we noted that SSB Q-Learning sometimes learnt an "expert"-like policy but then the performance degrades. Even though the events are still well extracted, it starts taking too long (see Figures 4k and 4l). The logs show that the AI learnt 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.

10.2 Summary

— We noted a slight improvement in the results with a faster reduction in ϵ (every 1 000 documents vs every 2 500), that is, with less exploration overall.
— 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, proving to be a feasible approach.
— MAG offers a good middle ground: it is quite robust to parameter choice, uses mostly intuitive feedback, and still learns a good to "expert"-like policy very quickly.

All in all, the experiments demonstrate that it is feasible 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.

11 Conclusion and future work

We modelled a document treatment chain which extracts events from web pages and other open source documents as a Markov Decision Process, and solved it using reinforcement learning. Our approach is not specific to a document treatment chain. It could be applied to any treatment where the component modules can be varied to change the result, so long as we can define the states in terms of the object being processed, and the actions in terms of those modules.

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 full numerical feedback, and evolved to a more realistic scenario by giving delayed, partial feedback. We then introduced further realism by integrating 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. Hence we 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, from analysts' habitual actions. Our work also evaluates, in an industrial context, the applicability of various algorithms and the impact of various parameters for important RL approaches, which is of independent interest.

The next step is to make the chain more complex, by adding alternative services, expanding the state set, and introducing a wider range of input. We aim to show that the system is capable of continuously improving the chain, by building different chains for different types of input. For example, learning whether a document in French should be translated to make use of the GATE extractors in English, or if the original text should be passed to GATE directly and the French extractors used. Even with a more complex system, the calculation time should not be an obstacle with a *Q-learning* type algorithm, where the calculations are instantaneous at each time step.

References

AKROUR R., SCHOENAUER M. & SEBAG M. (2011). Preference-based policy learning. In *Machine Learning and Knowledge Discovery in Databases*, p. 12–27. Springer.

AKROUR R., SCHOENAUER M. & SEBAG M. (2012). APRIL: Active preference learning-based reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, volume 7524 of *Lecture Notes in Computer Science*, p. 116–131. Springer Berlin Heidelberg.

AMANN B., CONSTANTIN C., CARON C. & GIROUX P. (2013). WebLab PROV: Computing fine-grained provenance links for XML artifacts. In *BIGProv'13 Workshop (in conjunction with EDBT/ICDT)*, p. 298–306, Gênes, Italy: ACM.

AZARIA A., RABINOVICH Z., KRAUS S., GOLDMAN C. V. & GAL Y. (2012). Strategic advice provision in repeated human-agent interactions. *Institute for Advanced Computer Studies University of Maryland*, **1500**, 20742.

BRAFMAN R. I. & TENNENHOLTZ M. (2003). R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, **3**, 213–231.

BRATKO I. & SUC D. (2003). Learning qualitative models. *Artificial Intelligence*, **24**(4), 107.

BUSA-FEKETE R., SZÖRÉNYI B., WENG P., CHENG W. & HÜLLERMEIER E. (2014). Preference-based reinforcement learning: evolutionary direct policy search using a preference-based racing algorithm. *Machine Learning*, **97**(3), 327–351.

CAMEL (2015). Apache Camel. `http://camel.apache.org/`. Accessed: 2015-03-17.

CARON C., AMANN B., CONSTANTIN C., GIROUX P. & SANTANCHÈ A. (2014). Provenance-based quality assessment and inference in data-centric workflow executions. In *On the Move to Meaningful Internet Systems: OTM 2014 Conferences*, p. 130–147: Springer.

COHEN W. W., DALVI B. B. & COHEN B. D. W. W. (2013). Very Fast Similarity Queries on Semi-Structured Data from the Web. In *SDM*, p. 512–520.

CUNNINGHAM H., MAYNARD D., BONTCHEVA K., TABLAN V., ASWANI N., ROBERTS I., GORRELL G., FUNK A., ROBERTS A., DAMLJANOVIC D., THOMAS HEITZ, MARK A. GREENWOOD, HORACIO SAGGION, JOHANN PETRAK, YAOYONG LI & WIM PETERS (2014). Developing Language Processing Components with GATE Version 8 (a User Guide). `https://gate.ac.uk/sale/tao/tao.pdf`. Accessed: 2014-12-17.

DOUCY J., ABDULRAB H., GIROUX P. & KOTOWICZ J.-P. (2008). Méthodologie pour l'orchestration sémantique de services dans le domaine de la fouille de documents multimédia.

DUTKIEWICZ J., JĘDRZEJEK C., CYBULKA J. & FALKOWSKI M. (2013). Knowledge-based highly-specialized terrorist event extraction. *RuleML2013 Challenge, Human Language Technology and Doctoral Consortium*, p.1̃.

FISHBURN P. C. (1984). SSB utility theory: an economic perspective. *Mathematical Social Sciences*, **8**(1), 63 – 94.

FORMIGA L., BARRÓN-CEDEÑO A., MÀRQUEZ L., HENRÍQUEZ C. A. & MARIÑO J. B. (2015). Leveraging online user feedback to improve statistical machine translation. *Journal of Artificial Intelligence Research*, **54**, 159–192.

FROMHERZ M. P., BOBROW D. G. & DE KLEER J. (2003). Model-based computing for design and control of reconfigurable systems. *AI magazine*, **24**(4), 120.

FÜRNKRANZ J., HÜLLERMEIER E., CHENG W. & PARK S.-H. (2012). Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine Learning*, **89**(1-2), 123–156.

GATE (2016). GATE Information Extraction. `https://gate.ac.uk/ie/`. Accessed: 2016-06-20.

GEONAMES (2015). Geonames. `http://www.geonames.org/`. Accessed: 2015-03-17.

GILBERT H., ZANUTTINI B., VIAPPIANI P., WENG P. & NICART E. (2015). Model-free reinforcement learning with skew-symmetric bilinear utilities. Accepted at UAI16. Available at `http://zanuttini.users.greyc.fr/research/ssbQLearning.pdf`.

GINSTROM R. (2007). The GITS Blog: Fuzzy substring matching with Levenshtein distance in Python. `http://ginstrom.com/`. Accessed: 2014-08-19.

KARAMI A. B., SEHABA K. & ENCELLE B. (2014). Apprentissage de connaissances d'adaptation à partir des feedbacks des utilisateurs. In *25es Journées francophones d'Ingénierie des Connaissances*, p. 125–136.

KNOX W. B. & STONE P. (2015). Framing reinforcement learning from human reward: Reward positivity, temporal discounting, episodicity, and performance. *Artificial Intelligence*, **225**, 24–50.

LAFREE G. (2010). The Global Terrorism Database: Accomplishments and Challenges | LaFree | Perspectives on Terrorism. *Perspectives on Terror*, **4**(1).

LOFTIN R., PENG B., MACGLASHAN J., LITTMAN M. L., TAYLOR M. E., HUANG J. & ROBERTS D. L. (2016-01). Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous Agents and Multi-Agent Systems*, **30**(1), 30–59.

NGRAMJ (2015). NGramJ, smart scanning for document properties. `http://ngramj.sourceforge.net/`. Accessed: 2015-02-18.

OGRODNICZUK M. & PRZEPIÓRKOWSKI A. (2010). 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)*, p. 1–7.

PANDIT S., GUPTA S. & OTHERS (2011). A comparative study on distance measuring approaches for clustering. *International Journal of Research in Computer Science*, **2**(1), 29–31.

PUTERMAN M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming 1st*. John Wiley & Sons, Inc. New York, NY, USA.

RAO K. & WHITESON S. (2011). *V-MAX: A General Polynomial Time Algorithm for Probably Approximately Correct Reinforcement Learning*. PhD thesis, Amsterdam.

RODRIGUES F., OLIVEIRA N. & BARBOSA L. (2015). Towards an engine for coordination-based architectural reconfigurations. *Computer Science and Information Systems*, **12**(2), 607–634.

SAÏS F., SERRANO L., KHEFIFI R. & SCHARFFE F. (2013). SOS-DLWD 2013.

SERRANO L. (2014). *Vers une capitalisation des connaissances orientée utilisateur: extraction et structuration automatiques de l'information issue de sources ouvertes*. PhD thesis, Universté de Caen.

SUTTON R. J. & BARTO A. G. (1998). *Reinforcement learning: An introduction*. MIT press.

SZEPESVÁRI C. (2010). Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, **4**(1), 1–103.

TIKA (2015). Apache Tika - a content analysis toolkit. `http://tika.apache.org/`. Accessed: 2015-02-18.

TVERSKY A. & GATI I. (1978). Studies of similarity. *Cognition and categorization*, **1**(1978), 79–98.

VAN HAGE W. R., MALAISÉ V., SEGERS R., HOLLINK L. & SCHREIBER G. (2011). Design and use of the Simple Event Model (SEM). *Web Semantics: Science, Services and Agents on the World Wide Web*, **9**(2), 128–136.

WATKINS C. J. C. H. (1989). *Learning From Delayed Rewards*. PhD thesis, Kings College.

WEBLAB (2015). WebLab wiki. `http://weblab-project.org/`. Accessed: 2015-03-17.

WILSON A., FERN A. & TADEPALLI P. (2012). A bayesian approach for policy learning from trajectory preference queries. In *Advances in neural information processing systems*, p. 1133–1141.

WIRTH C. & FÜRNKRANZ J. (2013). EPMC: Every visit preference monte carlo for reinforcement learning. In *Asian Conference on Machine Learning, ACML 2013, Canberra, ACT, Australia, November 13-15, 2013*, p. 483–497.

WIRTH C. & NEUMANN G. (2015). Model free preference-based reinforcement learning. In *EWRL*.