Amélioration continue d'une chaîne de traitement de documents avec l'apprentissage par renforcement ⋆

Esther Nicart [1, 2], **Bruno Zanuttini** [2], **Bruno Grilhères** [3], **Patrick Giroux** [3], **Arnaud Saval** [1]

1. *Cordon Electronics DS2i*
   27000 Val de Reuil, France
   {esther.nicart,arnaud.saval}@cordonweb.com

2. *Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC*
   14000 Caen, France
   {bruno.zanuttini,esther.nicart}@unicaen.fr

3. *Airbus Defence and Space*
   Élancourt, France
   {bruno.grilheres,patrick.giroux}@airbus.com

RÉSUMÉ. *Nous modélisons une chaîne de traitement de documents comme un processus de décision markovien, et nous utilisons l'apprentissage par renforcement afin de permettre à l'agent d'apprendre à construire des chaînes adaptées à la volée, et de les améliorer en continu. Nous construisons une plateforme qui nous permet de mesurer l'impact sur l'apprentissage de divers modèles, services* web*, algorithmes, paramètres, etc. Nous l'appliquons dans un contexte industriel, spécifiquement à une chaîne visant à extraire des événements dans des volumes massifs de documents provenant de pages* web *et d'autres sources ouvertes. Nous visons à réduire la charge des analystes humains, l'agent apprenant à améliorer la chaîne, guidé par leurs retours (*feedback*) sur les événements extraits. Pour ceci, nous explorons des types de retours différents, d'un* feedback *numérique requérant un important calibrage, à un* feedback *qualitatif, beaucoup plus intuitif et demandant peu, voire pas du tout, de calibrage. Nous menons des expériences, d'abord avec un* feedback *numérique, puis nous montrons qu'un* feedback *qualitatif permet toujours à l'agent d'apprendre efficacement.*

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 custommade chains "on the fly". We build a platform which enables us to measure the impact on the*

⋆. Cet article est une version étendue d'un article présenté aux 26^{e} Journées francophones d'Ingénierie des Connaissances Nicart *et al*. (2015).

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.*

MOTS-CLÉS : *intelligence artificielle, apprentissage par renforcement, extraction et gestion des connaissances, interaction homme-machine, renseignement d'origine source ouverte (ROSO).*

KEYWORDS: *artificial intelligence, reinforcement learning, extraction and knowledge management, man-machine interaction, open source intelligence (OSINT).*

DOI:10.3166/RIA.31.619-648 © 2017 Lavoisier

1. Introduction

Notre travail est motivé par les besoins réels de la communauté *Renseignement d'Origine Source Ouverte* (ROSO). Par le passé, la tâche des analystes des services de renseignement a été de chercher des informations cachées. Maintenant, ils font face à un volume toujours croissant de pages *web*, ainsi que d'autres documents d'origine source ouverte, et ils doivent extraire de l'information utile et pertinente à partir de cette richesse de données. Des chaînes de traitement spécialisées ont été développées pour faciliter cette tâche (par exemple, Ogrodniczuk, Przepiórkowski (2010) donnent une synthèse de quelques chaînes fournies en tant que services *web*).

Nous nous intéressons au problème générique de l'amélioration continue d'une telle chaîne de traitement de documents, plus précisément d'extraction d'événements d'intérêt pour la veille et le renseignement. Dans cette application, des documents provenant essentiellement du *Web* sont fournis en continu à une chaîne de traitement, qui vise à extraire des événements (par exemple, une attaque terroriste) et leurs caractéristiques (date, lieu, acteurs, *etc.*) pour les intégrer dans une base de données.

Dans de telles applications, l'extraction ne peut pas être parfaite. Tout d'abord, du fait de la grande diversité de documents sur le *Web*, il ne peut pas exister une unique chaîne optimale pour tous les documents. Ensuite, on peut imaginer, par exemple, qu'une dépêche relatant « le *bombardement*, par des ions, d'une *cible* d'or dans une animation autour de la physique *atomique* lors d'une *manifestation* pour la fête de la science », puisse induire une chaîne de traitement en erreur, et lui fasse insérer dans la base un attentat à l'arme atomique. Par ailleurs, la volonté de traiter des documents provenant du monde entier entraîne le besoin de traiter des documents dans des langues très diverses, pour lesquelles des dictionnaires peuvent être de qualité très variable. Pour toutes ces raisons, les analystes perdent du temps dans la tâche lourde et répétitive de la correction *a posteriori* des événements.

L'application qui nous intéresse utilise une chaîne de traitement, définie par des experts, consistant en un enchaînement figé (mais potentiellement conditionnel) de traitements atomiques, tels que la détection de la langue ou du format, la traduction, la détection d'événements en utilisant des mots ou verbes déclencheurs, *etc.* Actuellement, aucun mécanisme ne permet d'améliorer la chaîne au fil du temps sans l'intervention d'un expert, en consultation avec les analystes, pour recalibrer la chaîne.

Notre objectif est de combler ce manque, en fournissant un mécanisme automatique d'amélioration continue de la chaîne de traitement, tirant parti des retours (ou *feedback*) exprimés implicitement par les analystes lorsqu'ils consultent la sortie du système. Il s'agit de faire en sorte que la chaîne « apprenne de ses erreurs ». Par exemple, si le document est un article scientifique en français, il peut être préférable de le traduire d'abord en anglais, pour profiter du plus grand choix de règles d'extraction pour l'anglais, puis d'extraire les événements. En revanche, pour un *blog* sur lequel l'argot est abondamment utilisé, une extraction directe peut être préférable, car les dictionnaires sont insuffisants pour l'argot, et une traduction ne fait qu'ajouter du bruit. Ainsi, les services, ou briques pour la construction de la chaîne, restent des « boîtes noires » permettant à l'analyste de se distancier du processus d'extraction.

Nous devons récupérer le *feedback* de manière invisible pour l'analyste, sans impact sur son travail. Bratko, Suc (2003) montrent que l'expertise de l'utilisateur (son *feedback*) est manifestée par les traces de ses actions. Ces traces peuvent être captées à travers l'interface graphique qui donne le détail des événements en synthèse, et le *feedback* basé sur les interactions analyste-système actuelles. Par exemple, d'un événement *corrigé*, nous pouvons tirer un *feedback* explicite, via une distance entre l'événement idéal (corrigé) et celui qui avait été extrait. Nos premières expériences simulent ceci : l'agent reçoit un jugement sur la qualité de ses extractions sous la forme d'une distance, et donc d'un *feedback* numérique.

Mais l'événement peut aussi être *non extrait*, *correct mais extrait par un traitement trop lent*, *etc.* Prendre en compte ces différents aspects donne lieu à des questions, telles que « Est-il préférable d'extraire un événement incorrect, ou de manquer une extraction ? La qualité est-elle plus importante que la vitesse, ou l'inverse ? ». Il n'est pas évident de répondre à de telles questions avec une réponse numérique, par exemple, de dire qu'une extraction partielle mais rapide vaut 10.5 fois moins qu'une extraction complète mais lente. Il est essentiel, alors, que nous progressions vers un *feedback* intuitif, basé sur des préférences que les utilisateurs puissent exprimer naturellement, telles que « Je préfère une extraction incorrecte à une extraction manquée », ou encore « Je préfère que cela soit aussi rapide que possible ». Les besoins peuvent alors être spécifiés et changés très facilement, de façon intuitive.

1.1. Contribution

Nous formalisons notre problème comme un problème d'apprentissage par renforcement, et la chaîne de traitement elle-même comme un processus de décision markovien (MDP, Définition 4). Nous étudions trois cadres différents. Dans le premier,

l'agent reçoit des retours *numériques* sur la qualité du traitement, ce qui est très informatif, mais difficile à calibrer. Les deux autres cadres utilisent des retours *intuitifs* (pour l'utilisateur), avec une formalisation partiellement qualitative pour le deuxième, et totalement qualitative pour le troisième. De tels retours sont moins informatifs pour l'agent, mais requièrent beaucoup moins, voire pas du tout de calibrage.

Nous rendons compte de la plateforme que nous avons construite, appelée *BIMBO (Benefiting from Intelligent and Measurable Behaviour Optimisation)*. Dans cette plateforme, on peut « brancher » différents algorithmes d'apprentissage par renforcement, différents modèles de la tâche, différents services *web*, différents types de retours de l'utilisateur, *etc.*, ce qui nous permet de mesurer l'impact de ces éléments sur l'apprentissage. Nous rendons ensuite compte de l'utilisation de cette plateforme pour l'application industrielle qui nous intéresse, c'est-à-dire pour l'amélioration continue d'une chaîne de traitement qui extrait des événements à partir de pages *web* et d'autres documents de sources ouvertes. La plateforme sous-tend notre application, mais peut également être utilisée pour d'autres applications, et/ou pour l'étude de différentes méthodes d'amélioration continue de chaînes de traitement. Nous décrivons brièvement, en conclusion, son utilisation pour des tâches de segmentation d'images et de reconnaissance optique de caractères.

Nous présentons ensuite des résultats expérimentaux, tout d'abord avec des retours numériques (simulés) des utilisateurs (section 7). Les résultats montrent qu'apprendre automatiquement à enchaîner et paramétrer les services est totalement faisable en pratique. Nous présentons enfin des résultats expérimentaux avec des retours *intuitifs* (section 8). Là encore, et malgré l'information bien moins précise véhiculée par de tels retours, les résultats montrent que l'apprentissage reste totalement faisable en pratique, et ce, en demandant très peu d'effort de calibrage de la part des analystes.

1.2. Travaux connexes

À notre connaissance, il n'y a pas eu de travaux académiques s'intéressant à des agents capables de construire dynamiquement une chaîne de services, et d'apprendre et de s'améliorer en continu en utilisant des retours intuitifs des utilisateurs. Toutefois, de nombreux ingrédients ont été étudiés et proposés pour de telles applications.

L'enchaînement de services pour réaliser un but précis, par exemple, ne constitue pas une idée nouvelle. Des services peuvent être composés, par exemple, pour aider l'utilisateur à remplir des formulaires (Saïs *et al.*, 2013). Dans une autre application, les caractéristiques physiques de documents sont utilisées avec succès pour construire, en temps réel, des chaînes adaptatives et modulaires pour des photocopieurs Xerox (Fromherz *et al.*, 2003). Toutefois, les entrées et sorties de chaque service sont connues à l'avance, et enchaîner ces services constitue donc un problème de planification, sans apprentissage. Par ailleurs, les préférences de l'utilisateur ne sont pas prises en compte, et aucun retour n'est utilisé pour améliorer le processus d'une fois sur l'autre.

De façon générale, un utilisateur peut reconfigurer une chaîne en utilisant un langage dédié, comme ReCooPLa (Rodrigues *et al.*, 2015), ou en choisissant un service dans un répertoire de services du même type (Doucy *et al.*, 2008). Néanmoins, une chaîne ainsi reconfigurée ne s'adapte pas dynamiquement sans intervention de l'utilisateur, et ce dernier doit avoir une certaine expertise sur le système pour apprécier les conséquences de ses choix.

L'utilisation de retours de l'utilisateur, implicites ou explicites, pour l'apprentissage par renforcement, a également été explorée. C'est notamment l'objet d'un thème très actuel en intelligence artificielle : l'*apprentissage par renforcement inverse*. Par exemple, Knox, Stone (2015) étudient, sur des tâches synthétiques mais avec des utilisateurs réels, l'influence de la polarité (« encourageant », « punisseur », *etc.*) d'un utilisateur entraînant un agent, via des retours continuels, pour des tâches séquentielles. Loftin *et al.* (2016-01) étudient l'inférence de *feedback* implicite, non numérique, par des agents à partir de l'observation des actions prises par des utilisateurs sur les mêmes tâches. Un autre exemple de retour implicite consiste, pour l'utilisateur, à indiquer sa préférence parmi deux politiques montrées par l'agent (Akrour *et al.*, 2011). Par ailleurs, des modèles du comportement (Azaria *et al.*, 2012) et des préférences (Karami *et al.*, 2014) d'un utilisateur peuvent être construits en utilisant des interactions utilisateur-système, pour fournir des systèmes personnalisés. Toutefois, bien que ces systèmes soient capables de bien réagir, en particulier à des changements d'utilisateurs et de préférences, ils ne sont pas adaptés à notre cas d'utilisation, car ils requièrent une interaction continue avec l'utilisateur, qui doit constamment critiquer les choix de l'agent. Ceci aurait un coût prohibitif dans notre application, car les utilisateurs principaux du système sont les analystes, dont l'amélioration du système d'extraction n'est pas la tâche première, et car la chaîne est appelée à traiter des volumes extrêmement importants de documents (de l'ordre d'un document toutes les 20 secondes, en continu, pour une instance de la chaîne).

Nous considérons actuellement les services individuels constituant la chaîne comme des « boîtes noires », avec pour objectif que l'agent améliore la façon de les enchaîner et de les paramétrer. Nous notons toutefois que des travaux s'intéressent à l'estimation de la qualité de tels services (Caron *et al.*, 2014) et aux dégradations potentielles dans la chaîne (Amann *et al.*, 2013), ainsi qu'à leur mise à jour automatique en utilisant les retours de l'utilisateur (Formiga *et al.*, 2015). On pourrait tout à fait utiliser une combinaison de ces techniques en parallèle de notre approche.

2. La plateforme WebLab

La chaîne de traitement qui motive notre travail s'appuie sur la plateforme *open-source* (WebLab, 2015). WebLab intègre des services *web* qui encapsulent les processus unitaires, et qui peuvent être interchangés ou permutés afin de créer une chaîne de traitement. Cette chaîne peut ensuite être utilisée pour analyser des documents multimédia *open-source*, et en extraire l'information.

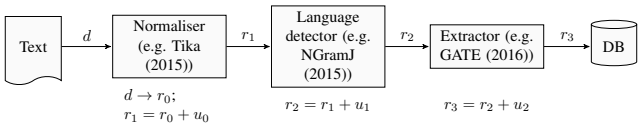


Figure 1. Une chaîne WebLab minimale : un document d est converti en une ressource XML r_0 et les ensembles d'annotations u_i sont ajoutés par chaque service

```
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="weblab: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>
22
```

Figure 2. Flux XML simplifié d'une ressource WebLab au milieu de la chaîne

Une chaîne de traitement typique de WebLab (figure 1), quand elle traite un nouveau document, commence par convertir le document source en une ressource XML. Cette ressource est ensuite transmise de service en service. Chaque service analyse le contenu de la ressource telle qu'il la reçoit, et l'enrichit avec des annotations. Enfin, les résultats sont stockés pour consultation par l'analyste.

EXEMPLE 1 (annotations). — Considérons le document textuel suivant :

```
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.
```

La ressource XML (simplifiée) de la figure 2 est produite après le passage du document par deux services :

– le *normaliser* a converti le document en ressource XML et a ajouté l'annotation « text/plain » (ligne 12) et le contenu original (ligne 19) ;

– le détecteur de la langue a ajouté la langue « en » (ligne 16). □

Nous considérons l'utilisation d'une telle chaîne pour l'extraction d'événements d'intérêt pour la veille économique, stratégique, ou militaire. En travaillant avec We-

bLab, nous nous situons dans la continuité du travail de Serrano (2014), qui propose une définition d'un événement (Définition 2). Cependant, notre travail est indépendant de WebLab, et nous aurions pu utiliser une autre définition (*e.g.* Hage *et al*. (2011)).

DÉFINITION 2 (Un événement). — *Un* événement E *est un quadruplet* $E = (C,T,S,A)$*, où :*

– $C \subseteq \mathbb{C}$ *est la dimension conceptuelle ou sémantique de* E*, donnée par un ensemble d'atomes pris dans un domaine* $\mathbb{C}$ *commun à tous les événements ;*

– $T \subseteq \mathbb{T}$ *est la dimension temporelle de* E*, c'est-à-dire quand* E *est survenu (potentiellement ambiguë, telle que « mardi dernier ») ;*

– $S \subseteq \mathbb{S}$ *est la dimension spatiale de* E*, c'est-à-dire le lieu où* E *est survenu (potentiellement ambiguë également) ;*

– $A \subseteq \mathbb{A}$ *est la dimension agentive de* E*, c'est-à-dire l'ensemble des participants impliqués.*

Plus précisément, dans le cadre de cet article :

– $\mathbb{C}$ est un ensemble fixé et fini d'atomes formalisés dans l'ontologie *WOOKIE* (Serrano (2014)) ;

– $\mathbb{T}$ est l'ensemble de toutes les dates relatives et absolues, par exemple « mardi dernier », « 2001 », « 2001/9/11 » ;

– $\mathbb{S}$ est l'ensemble des entités utilisées par Geonames (2015) ;

– $\mathbb{A}$ est l'ensemble (infini) de tous les participants pouvant être extraits, par exemple, les entités nommées, vus comme des chaînes de caractères.

EXEMPLE 3 (extraction). — Le document précédent (Exemple 1) donnera lieu à l'extraction d'un événement $E = (C,T,S,A)$ avec :

– $C = \{\text{AttackEvent, BombingEvent}\}$;

– $T = \{\text{4/29/1971}\}$;

– $S = \{\text{Los Angeles, California,United States, US, North America}\}$;

– $A = \{\text{Chicano Liberation Front, Bank of America}\}$. □

L'efficience d'une chaîne de traitement ne dépend pas seulement des services utilisés et de l'ordre dans lequel ils sont appelés, mais elle dépend aussi des paramètres utilisés par ces services. Par exemple, l'extracteur GATE est paramétré par des dictionnaires *(gazetteers)*, qui sont des listes de noms et de verbes déclenchant la détection d'un type spécifique d'événement. Son dictionnaire de verbes pour l'extraction d'événements de type *bombing* contient typiquement les verbes *explode*, *detonate*, etc. Nous voyons donc GATE comme un *service paramétré* : choisir les bons dictionnaires à lui donner en paramètre fait partie des actions à (apprendre à) réaliser pour l'agent.

3. Apprentissage par renforcement

Dans la pratique, la chaîne de traitement utilisée est complexe. Elle est écrite et calibrée par des experts qui choisissent les services constituant la chaîne, leur ordonnancement et leurs paramètres (par exemple, les *gazetteers* de mots déclencheurs pour les services de détection d'événements et d'entités nommées). L'ordonnancement est figé, tout en pouvant être conditionnel (par exemple, si le document est en format PDF, passer au service 1 pour le convertir en XML, et passer au service 2 sinon).

Même avec de l'expertise, il est très difficile d'obtenir une chaîne parfaite. Ceci provient du fait que l'utilisation des documents *open source* du *Web* apporte des difficultés : leurs formats et contenus ne sont pas standardisés, les pages sources elles-mêmes ne sont pas contrôlables, elles peuvent comporter du « bruit » (comme de la publicité), être piratées, et leurs URL peuvent changer. On observe des erreurs d'extraction pouvant consister en des événements d'intérêt entièrement ou partiellement manqués, ou des événements mal extraits (des informations non connexes dans la même phrase associées faussement, par exemple).

Par exemple, imaginons que l'analyste veuille de l'information sur les accords entre pays. Il est impossible de dire avec certitude si le mot « alliance » dans un document y fait référence ou non. Ce n'est qu'après l'extraction de l'événement, déclenchée par le mot « alliance », que l'analyste peut se rendre compte que la page parle de mariages, par exemple. Même si le document provient d'un journal politique, il se peut que l'on parle d'une coalition entre partis politiques, ou que les filtres de publicité n'aient pas réussi à attraper une vente de bagues. Un synonyme tel qu'« union » a pu être utilisé au lieu d'« alliance », et l'événement n'a pas été reconnu. Avec ces incertitudes, les experts qui paramètrent la chaîne essaient d'envisager les situations les plus communes. Il est inconcevable qu'ils puissent construire des chaînes à la main en examinant chaque document source.

Les seules informations connues avant de commencer une chaîne de traitement sont les services disponibles, leurs paramètres, et les états potentiels des documents, des ressources XML et du système (*cf.* section 2). L'apprenant ne connaît ni la forme, ni le contenu des documents à l'avance, et il ne sait pas si une extraction sera possible. Ses décisions sont prises dans l'incertain. Pour atteindre notre objectif, nous appliquons donc les techniques de l'*apprentissage par renforcement* (*Reinforcement Learning*, RL) pour les processus de décision markoviens (*Markov Decision Processes*, MDP (Puterman, 1994)). Nous en rappelons les grands principes dans la suite ; pour une introduction détaillée, nous renvoyons le lecteur à Sutton, Barto (1998).

En RL, l'apprenant reçoit une récompense, basée sur les résultats des actions qu'il a choisies. Plus les résultats sont proches des objectifs, plus la récompense est élevée. Le système essaie de maximiser ces récompenses, typiquement en *exploitant* ce qu'il connaît déjà pour continuer à recevoir de bonnes récompenses, et en *explorant* de nouvelles actions avec l'espoir d'obtenir des récompenses encore plus importantes.

Le RL est généralement formalisé pour des MDP. Un tel processus modélise l'environnement en termes d'*états*, dans lesquels des *actions*, qui mènent à d'autres états du système de manière stochastique, sont possibles. Le fait que l'environnement soit dans un état donné à un certain instant apporte une récompense immédiate à l'agent. L'objectif d'un apprenant est de choisir ses actions de façon à maximiser son espérance de récompenses cumulées, sans connaître, initialement, ni les distributions sur les états résultant d'une action, ni les récompenses associées aux états.

DÉFINITION 4 (MDP). — *Un* processus de décision markovien *est un 5-uplet* (S, A, P, R, γ)*, avec*

– S *un ensemble (ici fini) d'états possibles de l'environnement,*

– A *un ensemble (ici fini) d'actions (que l'agent peut effectuer),*

– P *un ensemble de distributions* $\{P_a(s,\cdot) \mid s \in S, a \in A\}$*;* $P_a(s, s')$ *est la probabilité que l'environnement soit dans l'état* s' *après que l'agent a effectué l'action* a *dans l'état* s*,*

– R *une fonction de récompense, que nous supposons définie sur les états ;* $R(s)$ *est la récompense obtenue par l'agent pour se trouver dans l'état* s*,*

– γ *est un facteur d'atténuation dans* $[0,1]$*, qui contrôle l'importance des récompenses espérées dans le futur, relativement aux récompenses espérées dans l'immédiat.*

Dans le cadre du RL, l'agent (apprenant) ne connaît initialement que les espaces d'états S et d'actions A, ainsi que le facteur γ. À tout instant t, il connaît l'état courant s_t de l'environnement, et choisit une action a_t. L'environnement passe dans un état s_{t+1} tiré selon la distribution $P_{a_t}(s_t,\cdot)$, et l'agent est informé de l'état s_{t+1} et de la récompense $r_{t+1} = R(s_{t+1})$. Le processus continue en s_{t+1}. L'agent doit, au fil de ses interactions avec l'environnement, apprendre une série de *politiques* $\pi_0,\pi_1,\ldots,\pi_t,\ldots$, une *politique* $\pi_t : S \to A$ donnant, pour l'instant t, l'action $\pi_t(s)$ à effectuer si l'état courant est s_t. Son objectif est à tout instant de maximiser l'espérance de récompense cumulée, c'est-à-dire l'espérance de la quantité $\sum_{t'=t}^{\infty} \gamma^{t'} R(s_{t'})$. Ce cadre générique admet de nombreuses variantes (pour un aperçu récent, voir Szepesvári (2010)).

De nombreux algorithmes ont été proposés dans la littérature pour les problèmes de RL. Dans cet article, nous utilisons d'abord une approche standard, le *Q-learning* (Watkins, 1989), avec des récompenses numériques, puis, pour le cadre des récompenses qualitatives, que les approches standards ne traitent pas, nous utilisons l'approche *SSB Q-learning* (Gilbert *et al.*, 2016). Notons que notre contribution consiste d'abord à modéliser le problème de l'amélioration continue d'une chaîne de traitement comme un problème de RL, et que d'autres algorithmes pourraient être utilisés.

Pour que cet article soit autosuffisant, nous présentons dans la suite les deux algorithmes de façon générale, mais nous encourageons le lecteur à consulter les articles originaux. Nous avons aussi lancé des tests en utilisant RMax (Brafman, Tennenholtz, 2003) et VMax (Rao, Whiteson, 2011), qui sont conceptuellement différents (ils ap-

prennent le MDP sous-jacent, au lieu d'apprendre directement la politique), mais leurs temps de convergence et de calcul les rendent prohibitifs pour notre problématique.

3.1. Q-learning

Le *Q-learning* est un algorithme simple, facile à mettre en œuvre et à paramétrer. Il maintient, pour chaque couple état/action (s,a), une valeur notée $\hat{Q}(s,a)$, qui représente l'estimation courante, par l'agent, de l'espérance de récompense s'il se trouve dans s, exécute a, puis suit une politique optimale. Lorsque l'agent est dans l'état s_t, choisit l'action a_t, se retrouve en s_{t+1} et reçoit une récompense r_{t+1}, il met à jour son estimation de la valeur $\hat{Q}(s_t,a_t)$ comme indiqué dans l'algorithme 1, où α, le *taux d'apprentissage*, est un coefficient dans $[0,1]$ qui fixe l'importance de la dernière expérience $(r_{t+1} + \gamma\, max(\dots))$ par rapport à l'expérience déjà accumulée (l'ancienne valeur de $\hat{Q}(s_t,a_t)$).

Pour l'exploration, nous utilisons une stratégie ϵ-gloutonne (EG). Le *taux d'exploration* $\epsilon \in [0,1]$ règle le dilemme exploitation/exploration de la manière suivante. À chaque pas de temps t, avec une probabilité ϵ, une action est choisie aléatoirement (l'agent *explore*) ; sinon, l'agent *exploite* et choisit simplement l'action a_t qui maximise $\hat{Q}(s_t,a_t)$.

Algorithme 1 : Q-learning

Données : MDP $\mathcal{M}$

tant que *vrai* **faire**

- Choisir a_t en utilisation la stratégie d'exploration EG
- Exécuter a_t, observer s_{t+1}
- Soit $r_{t+1} = \mathcal{R}(s_{t+1})$
- $\hat{Q}_{t+1}(s_t,a_t) \leftarrow$ $\hat{Q}_t(s_t,a_t) + \alpha_t(s_t,a_t)(r_{t+1} + \gamma\, max_b\{\hat{Q}_t(s_{t+1},b)\} - \hat{Q}_t(s_t,a_t))$

3.2. SSB Q-learning

Notre approche fonctionne très bien avec du *feedback* numérique (section 7), mais très peu d'utilisateurs pourraient dire « l'extraction d'un événement incorrect vaut 10,5 fois mieux qu'une extraction manquée, qui elle-même vaut 7,9 fois mieux qu'une extraction prenant 90 secondes ». Plus vraisemblablement, les utilisateurs peuvent affirmer « je préfère l'extraction d'un événement incorrect à une extraction manquée (car je ne veux pas perdre d'information), et je préférerais que l'extraction soit aussi rapide que possible ». Notons ici l'expression de deux préférences, qui, bien qu'elles ne soient pas indépendantes (on peut supposer qu'une extraction plus rapide soit de moins bonne qualité), ne peuvent pas facilement être reliées numériquement. Nous voulons donc permettre aux utilisateurs de donner des retours à l'agent de façon indépendante sur les deux dimensions : « je préfère qu'un événement soit extrait », et « je

préfère que l'extraction soit aussi rapide que possible ». De telles préférences peuvent être représentées par un ordre partiel sur les résultats possibles de l'extraction tels que perçus par l'utilisateur, notés $f_1, \ldots, f_k$ et vus comme des états finaux du MDP. Par exemple, f_i peut être « extraction correcte en 10 secondes », ou encore « traitement stoppé après 5 secondes, sans événement extrait ».

La résolution de problèmes de RL avec ce type d'information préférentielle/ordinale est le problème étudié par les travaux autour du *preference-based RL* (Akrour *et al.*, 2012; Fürnkranz *et al.*, 2012; Wilson *et al.*, 2012; Wirth, Fürnkranz, 2013; Wirth, Neumann, 2015). Dans ce contexte, on ne peut pas maximiser directement l'espérance de récompense cumulée, car les valeurs numériques des récompenses ne sont pas accessibles (tout du moins directement).

Une approche consiste à calculer une fonction de récompense numérique à partir des retours qualitatifs reçus, mais ceci a typiquement l'inconvénient d'introduire des informations additionnelles, non exprimées par l'utilisateur. Une approche alternative consiste à utiliser des critères de décision adaptés à l'information ordinale. Un critère de ce type est la *dominance probabiliste* (Busa-Fekete *et al.*, 2014), qui cherche à maximiser la probabilité d'obtenir un résultat (un état final f_i) préféré. Plus formellement, soient π_1,π_2 deux politiques, et soient F_1, F_2 deux variables aléatoires sur $(f_1, \ldots, f_k)$, où $\mathbb{P}(F_i = f_j)$ est donné par la probabilité que π_i aboutisse au résultat f_j ; $F_x \succeq F_y$ indique donc que le résultat obtenu par π_x est préféré (ou équivalent) à celui obtenu par π_y. Alors le critère de dominance probabiliste s'écrit :

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

En français, une politique π_1 est préférée à une politique π_2 si la probabilité que π_1 obtienne un meilleur résultat que π_2 est plus grande que l'inverse.

En fait, la dominance probabiliste est un cas particulier d'une large classe de critères de décision, appelés *Skew Symmetric Bilinear — SSB — utility functions* (Fishburn, 1984). Étant donné ϕ une fonction d'utilité SSB, $\phi(\pi, \pi') > 0$ (resp. $\phi(\pi,\pi') < 0$) exprime la préférence de l'utilisateur pour π sur π' (resp. pour π' sur π), tandis que $\phi(\pi,\pi') = 0$ exprime son indifférence. La dominance probabiliste est le cas où ϕ est toujours 1, 0 ou -1, et ne requiert donc de l'utilisateur que des retours purement ordinaux (« préféré », « indifférent ») sur les états finaux.

Pour traiter des problèmes de RL avec ce critère d'optimisation, nous utilisons l'algorithme *SSB Q-learning* récemment proposé (Gilbert *et al.*, 2016). Cet algorithme traite le cas général d'un critère SSB. Comme le Q-learning, il utilise l'exploration EG, et il met à jour les Q-valeurs d'une manière similaire. Toutefois, au lieu d'utiliser du *feedback* numérique, que l'utilisateur n'a pas à lui fournir, il utilise ϕ et ses traitements passés, précisément les fréquences $\mathbf{p}_t$ avec lesquelles il a obtenu chaque résultat possible jusqu'alors (la valeur numérique ainsi définie est notée $\mathcal{R}_{\mathbf{p}_t}(s_{t+1})$). Intuitivement, l'algorithme cherche continuellement à faire mieux, au sens de ϕ, qu'il n'a fait jusqu'alors. Le pseudo-code de *SSB Q-learning* est donné par l'algorithme 2.

Algorithme 2 : SSB Q-learning

Données : MDP $\mathcal{M}$, fonction SSB ϕ

tant que *vrai* **faire**
- Choisir a_t en utilisant la stratégie d'exploration EG
- Exécuter a_t, observer s_{t+1}
- Soit $r_{t+1} = \mathcal{R}_{\mathbf{p}_t}(s_{t+1})$
- $\hat{Q}_{t+1}(s_t,a_t) \leftarrow$ $\hat{Q}_t(s_t,a_t) + \alpha_t(s_t,a_t)(r_{t+1} + \max_b\{\hat{Q}_t(s_{t+1},b)\} - \hat{Q}_t(s_t,a_t))$
- **si** s_{t+1} *est un état final* f_i *et que l'agent n'a pas exploré* **alors**
  - $\mathbf{p}_{t+1} = \mathbf{p}_t + \frac{1}{\eta+1}(\mathbf{1}_i - \mathbf{p}_t)$
  - # η est le nombre de fois où $\mathbf{p}$ a été mis à jour

Enfin, pour les deux algorithmes, nous avons utilisé des versions avec *traces d'éligibilité* (Sutton, Barto, 1998, section 7). Dans ces versions, un paramètre $\lambda \in [0,1]$ permet et contrôle la rétropropagation des nouvelles expériences le long de la trace de décisions prises : $\lambda = 0$ correspond aux algorithmes tels que décrits ci-dessus, tandis qu'à l'autre extrême, $\lambda = 1$ correspond aux algorithmes de type Monte-Carlo.

4. Amélioration continue via l'apprentissage par renforcement

Puisque une chaîne unique pour traiter parfaitement chaque type de document est impossible à construire, l'idéal serait une chaîne faite sur mesure pour chaque document. Notre objectif est donc l'amélioration continue de la chaîne de traitement, de sorte que le système apprenne de ses erreurs. Une partie de notre contribution est de modéliser le traitement d'un document comme un processus décisionnel markovien (MDP), et son amélioration comme un problème d'apprentissage par renforcement. Une telle formalisation n'est pas directe, car le processus de traitement et ses entrées sont hétérogènes, et malgré tout ils doivent être formalisés en états et actions d'un problème de décision. De plus, les récompenses ne peuvent pas être spécifiées aisément, en particulier parce qu'elles doivent être récoltées de façon invisible pour les utilisateurs, sans impact sur leur travail. Finalement, la distribution des documents et les préférences des utilisateurs sont inconnues, et peuvent changer à tout moment.

Nous définissons les états en utilisant les métadonnées du document source, de la ressource et du système à un instant donné, par exemple, les informations déjà extraites, le temps déjà passé sur ce document, etc. (section 5). Le système a ainsi une perception de la tâche sous la forme d'*états* du processus, et chaque passage par un service modifie l'état courant.

Par ailleurs, le système dispose d'un certain nombre d'*actions* qu'il peut appliquer dans l'état courant : ces actions correspondent au service suivant à lancer, à l'arrêt du traitement et l'enregistrement des événements extraits en base de données, ou à la configuration d'un service. La répétition d'une même action sur un même document

est techniquement autorisée, mais sera pénalisée par le système de récompense si elle induit un temps de traitement trop long sans amélioration de l'extraction. La chaîne de traitement n'est plus figée, mais contrôlée par un algorithme de RL.

Les actions font transiter le système d'un état à l'autre. La stochasticité nous permet de prendre en compte le fait que des actions, menées dans un contexte apparemment identique, peuvent ne pas produire le même résultat. À titre d'exemple, choisir de détecter la langue peut résulter en l'extraction de langues différentes, ce qui est pris en compte directement par les actions stochastiques des MDP. Par exemple, supposons que 70 % des documents sources déjà traités étaient en français. L'agent percevra alors une probabilité 0.7 que, lorsqu'il applique l'action consistant à détecter la langue dans un état s_t, il arrive dans l'état s_{t+1}^{fr}, avec s_{t+1}^{fr} égal à s_t mais augmenté de l'information « langue extraite » et de l'annotation « fr ».

Plus précisément, les états que perçoit le système sont des états combinatoires, formés par les valeurs d'un certain nombre de *descripteurs* des documents. Ces états permettent une généralisation en apprentissage, par exemple, nous avons déjà vu que le *sujet* du document (politique vs. mariage) influence de manière forte l'utilité du mot « alliance » pour l'extraction de l'information. Imaginons qu'un utilisateur cherche des accords entre des pays. On peut espérer que le système apprenne à modifier la chaîne ainsi au fil des interactions :

– si l'état courant a la valeur « vrai » pour le descripteur « sujetExtrait » et la valeur « mariage » pour le descripteur « sujetDuDocument », alors la meilleure action à effectuer consiste à arrêter le traitement (inutile de continuer) ;

– si le sujet est extrait, mais n'est pas « mariage », la meilleure action consiste à lancer un service d'extraction qui utilise « alliance » parmi les mots déclencheurs ;

– sinon, la meilleure action consiste à lancer un service de reconnaissance du sujet du document.

Enfin, les récompenses sont construites à partir du *feedback* de l'utilisateur de trois manières. La première consiste en une valeur numérique, basée sur les corrections apportées à l'extraction par l'analyste (le cas échéant) et sur le temps passé à traiter le document (section 7). La seconde consiste en une valeur qualitative, basée sur les préférences purement ordinales de l'utilisateur sur les résultats finaux de l'extraction, et la troisième, qui forme un entre-deux, consiste en une préférence ordinale pondérée (section 8). Les récompenses $r(s)$ sont donc données à l'agent uniquement pour les états finaux des traitements.

En pratique, il s'agit d'utiliser le fait que les analystes consultent les synthèses produites par le système, synthèses qui présentent les événements extraits, avec des liens vers les documents sources. Les actions de l'analyste, consistant par exemple à corriger l'information extraite, ou encore à consulter une synthèse sans rien corriger, fournissent indirectement un retour sur la qualité du traitement opéré, tout en utilisant, de manière non intrusive, un travail effectué par les analystes indépendamment de la problématique d'amélioration du processus de traitement.

5. Cadre expérimental

Nous nous basons sur une chaîne simple mais typique (*cf.* section 2), à laquelle nous avons ajouté la possibilité d'utiliser un service en réalité inutile, *Geo*. La chaîne est écrite comme une route Camel (2015) en XML, où chaque service est défini comme un *endpoint*. Nous utilisons le *Dynamic Router* pour donner à *BIMBO* le contrôle sur les services appelés, leur ordonnancement, et leurs paramètres (ces derniers sont plus spécifiquement le choix des *gazetteers* de GATE (Cunningham *et al.*, 2014) pour la détection des événements dans un texte). Notons que les services sont des « boîtes noires » pour *BIMBO*, et que la connaissance de leurs WSDL, par exemple, n'est pas nécessaire [1].

Les actions disponibles consistent donc à :

– choisir le prochain service parmi {*Tika, NGramJ, GATE, Geo*} ;

– arrêter le traitement du document (« *STOP* ») ;

– choisir la configuration du service GATE, c'est-à-dire, ses *gazetteers*.

Un état est représenté par une affectation de caractéristiques :

– la langue du document, *language* $\in$ {"en"," "} (où " " code « non-extraite ») ;

– le format du document, *format* $\in$ {"*text/plain*"," "} (où " " code « non-extrait ») ;

– le nombre de secondes écoulées depuis le début du traitement du document courant (arrondi à la dizaine de secondes), *durée* $\in \mathbb{R}$;

– le nombre de services déjà appliqués sur ce document (groupés pour éviter l'explosion du nombre des états), *nbServices* $\in \{0-5{,}6-20{,}21+\}$;

– si un événement « intéressant » a déjà été extrait, *interesting* $\in \{$*true, false*$\}$;

– si un événement quelconque a déjà été extrait, *any* $\in \{$*true, false*$\}$;

– le *gazetteer* de noms choisi (si un tel choix a déjà été fait), *nounGazetteer* ;

– le *gazetteer* de verbes choisi (si un tel choix a déjà été fait), *verbGazetteer*.

Dans nos expériences, ces ensembles donnent un espace de $8\,000$ états pour 15 actions, et sont choisis à titre illustratif pour la preuve de notre approche. Un système opérationnel prendrait évidemment en compte de nombreuses autres caractéristiques, services et paramètres (type de documents, liste complète de langues, service de traduction, etc.). Notons que même avec un espace d'états/d'actions ainsi enrichi, le temps de calcul ne serait pas un obstacle avec un algorithme de type *(SSB) Q-learning*, pour lequel les calculs sont instantanés à chaque pas de décision.

Nous avons considéré un corpus de textes, dont les événements d'intérêt sont déjà connus. Le *Global Terrorism Database* (GTD) est une base de données *open source* composée des détails de plus de 125 000 événements terroristes mondiaux de 1970 à 2014 (LaFree (2010)). Nous avons mis les types d'événements du GTD en correspon-

1. Bien sûr, une telle information pourrait être utilisée afin de généraliser sur les actions par exemple, ou de construire un modèle *a priori* des actions et des états, mais nous laissons ceci pour un travail futur.

dance avec ceux de WOOKIE ; une chaîne d'extraction parfaite extrairait des documents $d_1 \ldots d_N$ de notre corpus les événements $E_1, \ldots, E_N \in GTD$, respectivement (*cf.* Exemple 1, Exemple 3). Une telle chaîne n'existe pas en réalité. Nous utilisons donc les résultats d'une chaîne « experte » (construite à la main) comme référence. Nous attendons de notre système qu'il apprenne une chaîne qui s'approche de ce but, en apprenant le bon ordonnancement des services, le bon paramétrage des services, et le fait que certains services (dans notre cas, *Geo*) et certains *gazetteers* de GATE ne sont pas utiles.

Comme expliqué dans la section 3, l'agent (l'IA) n'a initialement connaissance que de l'espace des états et des actions, et donc, dans notre cas, commencera à apprendre sans connaissance *a priori* des documents ou des besoins de l'analyste. Nous dirons qu'une telle IA est « non-formée ». Une fois que l'IA a traité un certain nombre de documents, elle aura appris ce qu'elle considère être une politique optimale. Nous dirons qu'elle est « formée », et nous pourrons vérifier l'efficacité de la politique apprise en obligeant l'IA à la suivre, c'est-à-dire, en mettant $\epsilon = \alpha = 0$, arrêtant ainsi son exploration et son apprentissage.

Ici, nous testons la capacité d'apprentissage à partir de rien d'une IA sur des documents complètement inconnus. Bien sûr, en production, nous pourrions capitaliser sur l'expertise existante en initialisant l'IA avec une politique basée sur celle d'une chaîne standard.

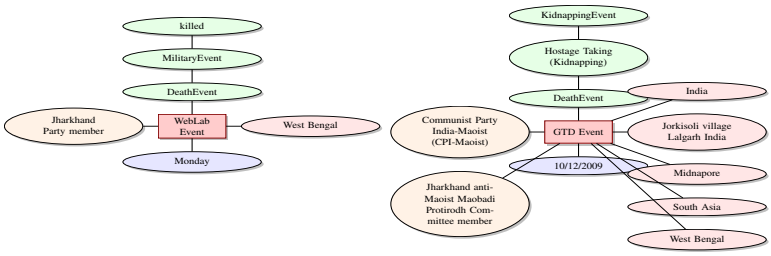
6. Mesure de la qualité des résultats

Pour mesurer la qualité des résultats de nos tests, il faut d'abord définir la similarité entre un événement extrait par la chaîne sur le document $E_1 = (C_1,T_1,S_1,A_1)$ et l'événement « parfait » du GTD (l'oracle) $E_2 = (C_2,T_2,S_2,A_2)$ qui correspond à ce document. De nombreuses méthodes ont été proposées pour mesurer la similarité (Pandit *et al.* (2011); Tversky, Gati (1978); Cohen *et al.* (2013); Dutkiewicz *et al.* (2013), pour n'en citer que quelques-unes), mais nous avons besoin d'une mesure qui prenne en compte les quatre dimensions des événements, et sensible par ailleurs à la différence entre différents types d'événements. Nous définissons donc :

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

La similarité conceptuelle $\sigma_C(C_1, C_2)$ est de 1 s'il y a un atome commun entre les dimensions conceptuelles de E_1 et E_2, et de 0 sinon. Par exemple, pour $C_1 = \{BombingEvent, AttackEvent\}$ et $C_2 = \{BombingEvent\}$, on obtient $C_1 \cap C_2 = \{BombingEvent\}$, et donc $\sigma_C(C_1, C_2) = 1$. Nous procédons de même pour la similarité géographique $\sigma_S(S_1, S_2)$.

La similarité temporelle $\sigma_T(T_1, T_2)$ est de 1 s'il y a au moins un atome en commun entre les dimensions temporelles de E_1 et E_2. Sinon, nous utilisons l'information dérivée (jour, mois, année, jour de la semaine). Par exemple, pour $T_1 = \{7\ October\ 1969\}$ et $T_2 = \{October\}$, $\sigma_T(T_1, T_2) = \frac{1}{10}$. Pour $T_3 = \{Tuesday\}$, l'intersection avec



(a) L'événement extrait (b) L'événement extrait « parfait » du GTD

Figure 3. Comparaison de deux événements

T_1 est vide, mais en notant que le 7 octobre 1969 a été un mardi, nous obtenons $\sigma_T(T_1, T_2) = \frac{1}{7}$. Ces valeurs ont été choisies par expérience, et soulignent la difficulté d'associer une valeur numérique à une comparaison. Cette difficulté sera levée par l'approche basée sur du *feedback* qualitatif.

Enfin, pour la similarité entre les dimensions agentives de E_1 et E_2, nous utilisons la distance de Levenshtein sur chaque paire d'agents a_1,a_2 pris dans A_1,A_2 (nombre minimal de caractères à supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre), distance rendue « floue » en considérant les sous-séquences de la chaîne principale (Ginstrom, 2007), et notée $\tau_A(a_i, a_j)$. Nous définissons $\sigma_A(A_1, A_2)$ comme $(1 - \max\{\tau_A(a_1, a_2) \mid a_1 \in A_1, a_2 \in A_2\})$, si elle est au-dessus d'un certain seuil θ, et 0 sinon (après expérimentation, $\theta = 0.45$ donne de bons résultats). Par exemple, pour $A_1 = \{$*Jharkhand ~~anti-Maoist Maobadi Protirodh Committee~~ member*$\}$ et $A_2 = \{$*Jharkhand ~~Party~~ member*$\}$, on obtient une distance Levenshtein floue de 11, donc $\sigma_A(A_1, A_2) = 1 - \frac{11}{22} = 0{,}5$.

Nous ne nous intéressons pas ici à l'association d'entités telles que *François Hollande / le président*, mais la modularité du système permettrait de prendre en compte cette similarité facilement, en s'appuyant sur des ressources adéquates.

Les volumes de documents traités en production (idéalement, tous les documents du *web*) étant très importants, le temps passé à traiter un document doit être minimisé. La qualité des extractions prend donc en compte la similarité entre l'événement cible et l'événement extrait (le cas échéant), mais aussi le temps de traitement. Précisément, en notant $\tilde{\sigma}(E_e, E_g)$ la similarité moyenne entre les événements extraits (s'ils existent) et l'événement du GTD, et t le temps passé par la chaîne sur le document, nous définissons la qualité de l'extraction Q par $\tilde{\sigma}(E_e, E_g)/t$ pour $\tilde{\sigma}(E_e, E_g) \neq 0$, et par $Q = -t$ sinon (c'est-à-dire s'il n'y pas d'extraction, ou une similarité nulle avec l'événement cible). Nous formalisons ainsi le fait que l'extraction d'événements corrects est primordiale, doit être effectuée en un temps raisonnable, et que l'agent doit détecter rapidement, le cas échéant, qu'il n'y a pas d'événement intéressant à extraire.

EXEMPLE 5 (Similarité). — Prenons les deux événements qui sont décrits dans la figure 3. Nous voyons un atome commun (*Death Event*) dans la dimension conceptuelle, qui donne une similarité de 1. La similarité spatiale est de 1 grâce à l'élément *West Bengal*. Nous déduisons le jour de la semaine, pour obtenir une similarité temporelle de 0,143. Enfin, la similarité agentive, comme nous avons déjà vu, est de 0,5. Avec des poids égaux sur les dimensions ($a = b = c = d = 1$), nous avons une similarité globale entre l'événement extrait, et celui du GTD de

$$\sigma(\textit{WebLab Event}, \textit{GTD Event}) = \frac{1{,}0 + 0{,}143 + 1{,}0 + 0{,}5}{4} = 0{,}66075$$

□

7. Tests avec un *feedback* numérique

Durant ces tests, l'IA reçoit une récompense numérique sur la qualité Q de ses résultats, telle que définie dans la section 6. En production, ceci équivaut à la récompenser en fonction de l'événement qu'elle a extrait (si elle en a extrait un) et de l'événement tel que corrigé par l'analyste humain, ce dernier étant considéré comme l'événement « parfait » pouvant être extrait du document (si un événement peut être extrait). Ceci est réalisé de manière non intrusive, dans la mesure où cela repose sur des corrections que l'analyste a besoin d'effectuer de toute façon, mais nécessite un réglage fin de la définition de la similarité et de ses paramètres, une tâche cognitive difficile. Notons que l'information donnée à l'IA, ici, est la même que celle utilisée pour l'évaluation de la qualité de la politique qu'elle a apprise. Cette méthode est classique en RL, mais pas naturelle dans la vie réelle. Dans la section 8, nous retirons cette contrainte et produisons les résultats obtenus avec du *feedback* qualitatif.

Nous avons constitué trois groupes de tests avec une récompense numérique :

1. Une IA a été formée et sa performance a ensuite été comparée avec celle d'une IA non-formée, et d'une chaîne « experte » sur $1\,000$ documents inconnus, biaisés vers un type d'événement « intéressant ».

2. La performance d'une IA non-formée a été testée sur $5\,000$ documents inconnus, choisis aléatoirement, avec un très faible pourcentage d'événements extractibles.

3. Finalement, la capacité d'apprentissage d'une IA non-formée a été examinée avec une récompense sporadique.

7.1. Formation et test sur des documents biaisés vers un type d'événement « intéressant »

Nous avons d'abord testé la qualité de la politique apprise (1) après un entraînement répété sur un petit ensemble de documents, et (2) à partir de zéro sur un ensemble plus large de documents choisis au hasard.

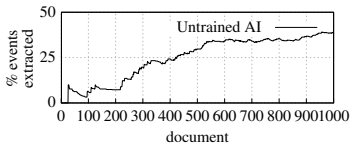


Figure 4. Pourcentage d'événements extraits par l'IA non-formée

(1) Nous avons entrainé notre AI sur 100 documents GTD, traités dans le même ordre 30 fois. L'IA a eu le choix entre les services et ***STOP*** (comme dans la section 5), et six *gazetteers* (trois listes de verbes et trois de noms). Deux paires de *gazetteers* (les noms et verbes *bombing* et *injure*) contiennent des listes de mots susceptibles de déclencher l'extraction d'événements de ce type. Une autre paire de *gazetteers* factices contenait des mots non présents dans la GTD. Nous avons représenté la préférence d'un analyste en définissant les événements $Bombing$ comme des événements « intéressants » en accentuant la similarité sémantique (spécifiquement $a = 20, b = c = d = 1$). Il s'agissait de vérifier que l'IA tiendrait compte de la préférence de l'analyste en favorisant les *gazetteers bombing* par rapport aux *gazetteers injury*, et en ignorant les factices, qui ne doivent donner lieu à aucune extraction. $64\,\%$ des documents décrivaient ces événements $Bombing$ « intéressants », $17\,\%$ d'autres événements, et $19\,\%$ ne contenaient aucun événement extractible.

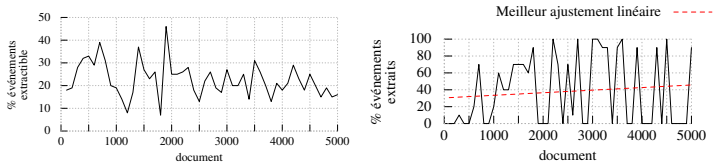
Le Q-learning a été lancé avec des paramètres standards : $\epsilon = 0{,}4$, qui sera divisé par 2 tous les 500 documents jusqu'à $\epsilon = 0{,}1$, $\alpha = 0{,}2$. Pour ces tests λ était fixé à 0 (pas de traces d'éligibilité).

(2) Nous avons testé à la fois la politique apprise par cette IA formée, et une IA non-formée sur $1\,000$ nouveaux documents choisis aléatoirement depuis la GTD. Seulement $29\,\%$ d'entre eux contenait des événements $Bombing$ « intéressants », $7\,\%$ d'autres événements, et $64\,\%$ aucun événement extractible. Nous avons réglé la chaîne « experte » (*c.f.* la section 5) pour n'extraire que des événements $Bombing$.

L'IA non-formée (partant de zéro) est parvenue à extraire $39\,\%$ des événements possible des $1\,000$ documents inconnus, et nous voyons sur la figure 4 qu'elle généralise bien, sa performance s'améliorant au fur et à mesure des documents rencontrés.

L'IA formée a démontré une politique similaire à celle de la chaîne experte, en extrayant $100\,\%$ des événements. Ainsi, elle aussi généralise bien, appliquant avec succès la politique apprise à des documents inconnus. Notons que les IAs ont également appris à ordonner la chaîne, *e.g.*, dans l'état correspondant à un document dont ni le format ni la langue ont été détectées, et dont le temps de traitement est quasi-nul, la meilleure action apprise était de passer le document au service Tika. Dans l'état correspondant à un document où au moins un événement est extrait, quelle que soit la valeur des autres attributs, l'IA arrête le traitement de ce document.

Comme espéré, l'IA a aussi appris à optimiser la chaîne. Elle n'a pas utilisé le service inutile (*Geo*), et a seulement utilisé la liste de verbes *bombing*. Cependant, elle



(a) Le pourcentage de documents contenant des événements extractibles par 100 documents

(b) Le pourcentage des événements extraits en utilisant les valeurs apprises à la fin de chaque 100 documents

Figure 5. Résultats expérimentaux sur des documents

a de manière surprenante préféré les noms *injure* à ceux de *bombing*. Après enquête, nous avons trouvé que le service GATE fourni n'utilise que les verbes pour l'extraction d'événements, ignorant les noms. Ceci montre que l'IA est capable de découvrir des stratégies qui ne sont pas évidentes pour l'expert ayant calibré la chaîne.

7.2. Tests de performance sur des documents avec un très faible pourcentage d'événements extractibles

Pour ces tests, nous avons augmenté l'espace des actions en donnant à l'IA le choix parmi dix *gazetteers* (cinq listes de verbes et cinq de noms) : six (les verbes et noms de *bombing*, *shooting* et *injure*) contiennent des mots susceptibles de déclencher la détection d'un événement de ce type, deux (les verbes et noms de *mixte*) contiennent un mélange de mots susceptibles de déclencher la détection de plusieurs types d'événements, et deux sont factices. Nous nous attendions à ce que l'IA apprenne à utiliser de préférence les *gazetteers mixte*, ensuite les *gazetteers* qui déclenchent la détection d'un événement unique, et qu'elle évite les *gazetteers factices*. Nous nous attendions également à ce que l'IA ne soit pas sensible à un espace d'actions plus grand.

Nous avons testé une IA non-formée cette fois-ci sur $5\,000$ documents inconnus, aléatoirement choisis avec en moyenne $22{,}56\,\%$ d'événements extractibles (figure 5a).

Parallèlement, après le traitement de chaque 100 documents, nous avons testé une IA initialisée avec la politique apprise à ce point, et les valeurs de ϵ et α à 0, sur 10 documents inconnus contenant tous des événements extractibles.

Nous voyons dans les figures 6– 7 que la première extraction n'a lieu qu'à 173 documents, et que les extractions sur les premiers 700 documents ont une qualité inférieure à celle de la chaîne « experte ». Au document 701, en revanche, l'IA commence à recevoir des récompenses similaires à celles de la chaîne « experte », ce qui indique que la qualité de ses extractions s'améliore.

En testant les politiques apprises, nous constatons à la Figure 5b une amélioration de performance de l'IA, même si les politiques ne sont pas constamment capables

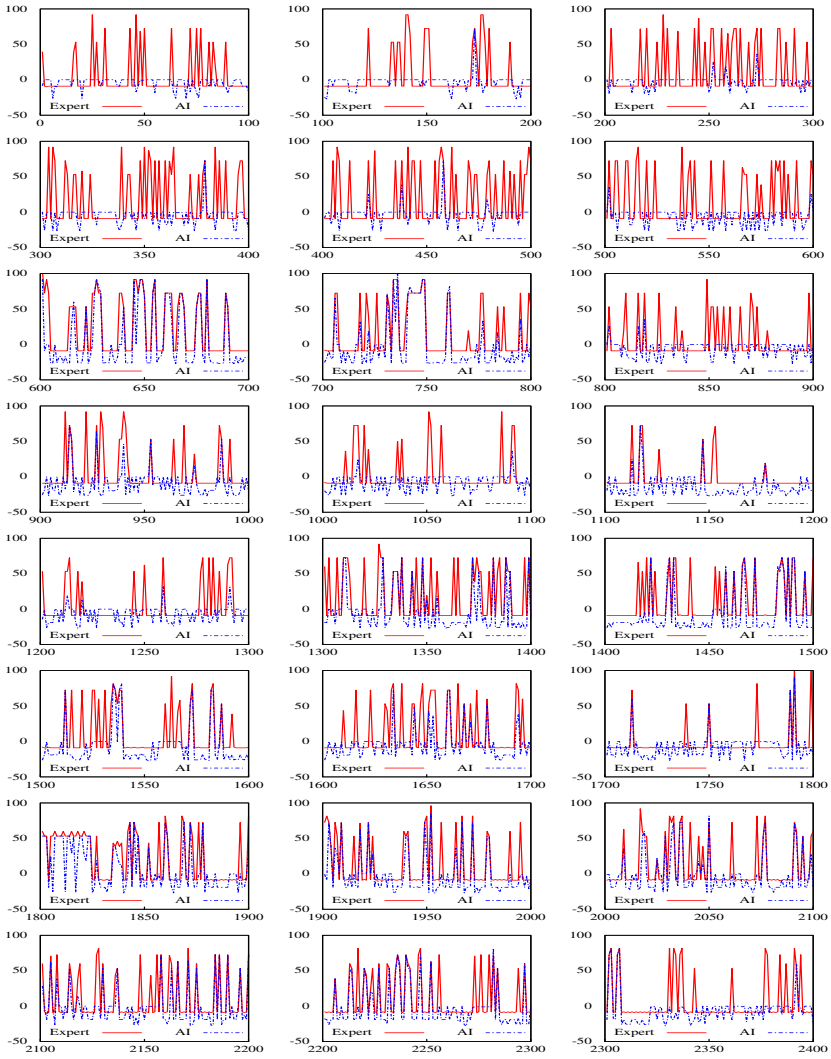


Figure 6. La qualité des extractions de la chaîne « experte », comparées à celles de l'IA non-formée pour les documents 1 - 2 400

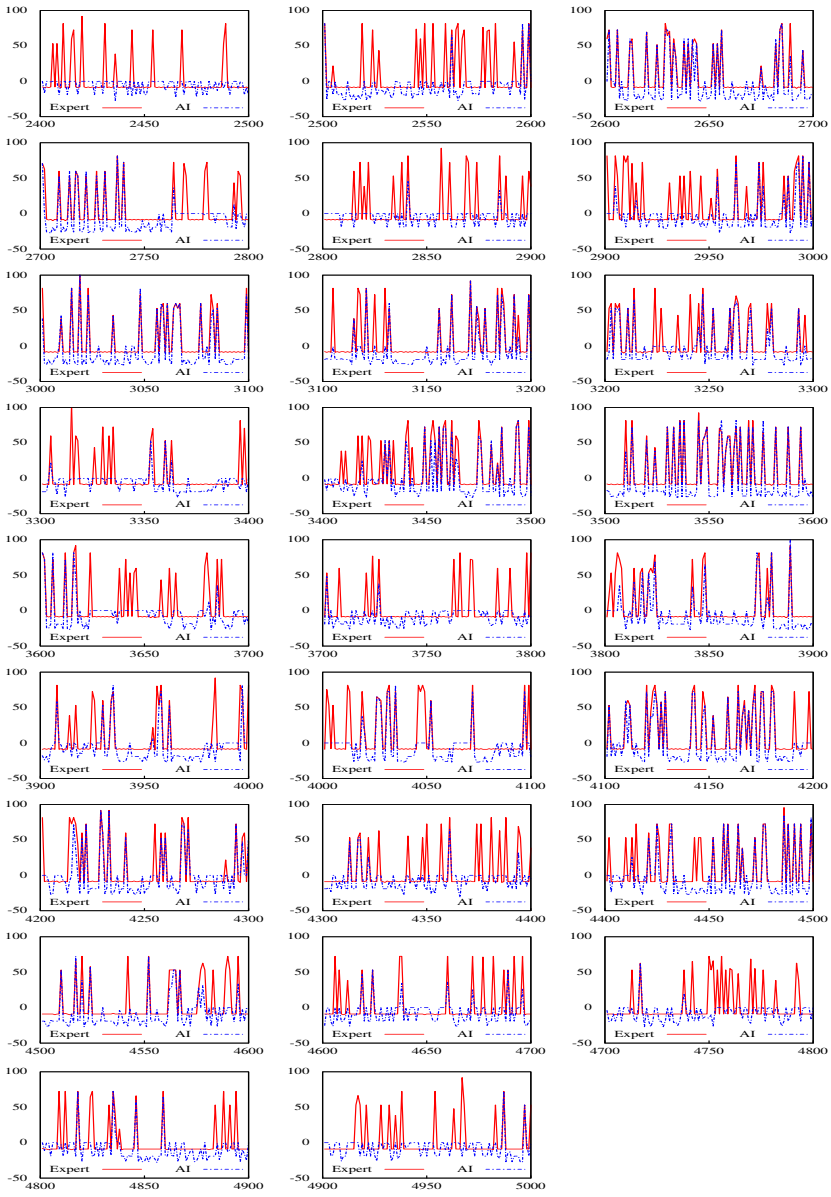


Figure 7. La qualité des extractions de la chaîne « experte », comparées à celles de l'IA non-formée pour les documents 2 401 - 5 000

d'extraire $100\,\%$ des événements. Par exemple, au document 900, l'IA traite la quasi-totalité des documents le plus rapidement possible. Ceci peut être dû à la baisse en pourcentage d'événements extractibles dans les 100 documents précédents. Elle préfère recevoir une récompense de zéro pour passer un document tout de suite, au lieu de recevoir une récompense négative en essayant d'extraire un événement qui n'existe pas. En comparant les documents 1 100–1 300 de la figure 6 avec la figure 5a, nous constatons que le pourcentage d'événements extractibles est très faible, et que l'IA a des problèmes de qualité et de quantité de ses extractions. Nous pouvons déduire de ces résultats que l'IA est sensible au pourcentage d'événements extractibles, mais qu'elle n'a besoin que de documents ayant environ $20\,\%$ d'événements extractibles pour réussir les extractions.

7.3. Tests de performance avec un **feedback** *sporadique*

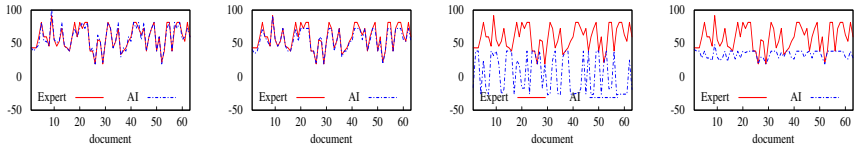
Les deux jeux de tests précédents avaient pour but d'évaluer la capacité de l'IA à extraire des événements depuis des documents inconnus. Nous avons réalisé une troisième série de tests afin d'évaluer la vitesse à laquelle une IA non-formée pourrait apprendre la politique d'une chaîne experte (capable d'extraire $100\,\%$ des événements), et la qualité de cette politique, pour vérifier que les événements extraits sont bien similaires aux résultats attendus. Pour cela, nous avons pris un jeu de 63 documents de la GTD, chacun d'eux contenant un événement extractible. Nous avons traité ces documents dans le même ordre, de manière répétée, par lots. Nous avons utilisé le même espace d'actions qu'en section 7.2, c'est-à-dire, un choix entre les services, *STOP* et dix *gazetteers* (cinq listes de verbes, et cinq listes de noms). Le Q-learning a été lancé avec des paramètres similaires à ceux des premiers tests : $\epsilon = 0{,}4$, divisé par 2 tous les 10 lots jusqu'à $\epsilon = 0{,}1$, $\alpha = 0{,}2$, et $\lambda = 0$.

Nous avons d'abord entraîné l'IA avec une récompense donnée pour chaque document. Il s'est avéré qu'après seulement $1\,008$ documents (c'est-à-dire 16 lots), la politique apprise était capable d'extraire $100\,\%$ des événements. De manière plus importante, cette politique était optimale, c'est-à-dire que la qualité de l'extraction était à égalité avec celle de la chaîne experte, comme l'illustre la figure 8a.

L'analyste n'est pas disponible en permanence pour consulter chaque document dès qu'il est traité. Nous avons alors lancé un test similaire, mais la récompense n'était donnée qu'à la fin du traitement de chaque lot, empêchant l'IA d'apprendre durant un lot. L'IA était légèrement plus lente à apprendre une politique optimale, ayant besoin de $1\,260$ documents (20 lots, figure 8b).

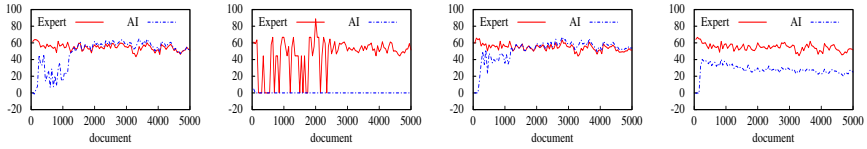
Enfin, les analystes ne sont pas des entraîneurs dédiés, et ne corrigeront pas toutes les extractions. Nous avons donc testé une récompense donnée à la fin de chaque lot, avec une probabilité de $10\,\%$ sur chaque extraction (sinon, l'IA ne recevait aucune récompense pour cette extraction).

Même avec une récompense aussi sporadique, l'IA est parvenue à extraire $50\,\%$ des événements possibles après seulement $1\,890$ documents (30 lots), et $100\,\%$ après

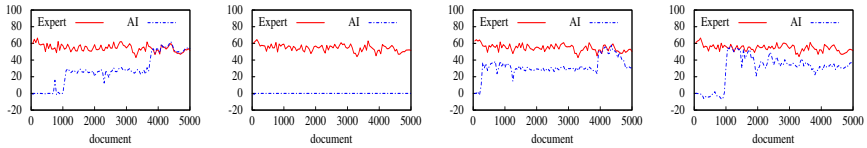


(a) Complète instantanée (b) Complète différée (c) Sporadique 50 % (d) Sporadique 100 %

Figure 8. La qualité de la politique apprise avec une récompense (a) complète instantanée ; (b) complète différée ; (c) sporadique à 50 % *extraction ; (d) sporadique à* 100 % *extraction*



(a) QLγ0.9λ0.95 (b) QL$\gamma 1 \lambda 1$ (c) MAGγ0.9λ0.95 (d) MAG$\gamma 1 \lambda 1$



(e) DOM$\gamma 1 \lambda 1$ (f) DOMγ0.9λ0.95 (g) MAGγ0.9$\lambda 1$ (h) DOMγ0.9λ0

Figure 9. Résultats : (a) QL meilleur ; (b) QL pire ; (c) MAG meilleur ; (d) MAG pire ; (e) DOM meilleur ; (f) DOM pire ; (g) et (h) MAG et DOM dégradation

5 103 documents (81 lots). La qualité de ces deux politiques est représentée sur les figures 8c,8d, et suggère que la capacité à extraire des événements et à extraire des événements corrects croissent conjointement. Notons que la qualité la plus basse (en comparaison avec la chaîne experte) que l'on observe dans la dernière courbe est due au temps de traitement, et non à la similarité avec les événements attendus.

8. Tests avec un *feedback* intuitif

Les tests présentés en section 7 supposent que l'utilisateur est capable de donner un *feedback* numérique à l'agent, ce qui n'est pas réaliste. Nous présentons ici donc une évaluation expérimentale de notre méthode en utilisant un *feedback* qualitatif. Rappelons que l'approche n'a besoin que d'un retour sur les qualités relatives de deux

extractions ; la comparaison entre politiques se fait alors sur le critère de dominance pondérée, relativement à ces comparaisons.

Nous explorons ici deux types de comparaison entre les résultats de deux traitements f,f'. Le protocole expétimental est similaire à celui de la section 7, et nous prenons comme références la chaîne experte, ainsi que les résultats obtenus par le Q-learning avec une récompense numérique (noté QL dans cette section).

Premièrement nous encourageons l'IA à d'abord extraire les événements, et à le faire rapidement, ou à vite reconnaître qu'il n'y a aucun événement à extraire. Ceci repose sur la corrélation entre la capacité à extraire n'importe quel événement et celle à extraire des événements corrects, tel que démontré à la section 7 pour le Q-learning. En l'état, notons que cette récompense peut même être calculé automatiquement, sans intervention de l'analyste. Évidemment, cette relation de préférence pourrait être complétée avec des informations supplémentaires sur les résultats obtenus, tels que la qualité perçue de l'extraction, *etc.* Cette récompense noté DOM est défini comme :
– $f \succ_{\mathrm{DOM}} f' \Leftrightarrow \phi_{\mathrm{DOM}}(f,f') = 1$ ssi (i) le traitement f a extrait un événement mais pas f', ou (ii) aucun ou bien les deux ont extrait un événement, et f a été plus rapide ;
– $f \sim_{\mathrm{DOM}} f' \Leftrightarrow \phi_{\mathrm{DOM}}(f,f') = 0$ ssi les deux ont pris à peu près le même temps, *i.e.*, l'écart entre les deux temps de traitement est inférieur à $margin$ (réglé à 5) secondes.

Nous notons MAG le deuxième type de récompense. Intuitivement, MAG est un intermédiaire entre QL et DOM, qui peut être vu comme une forme pondérée de dominance probabiliste, demeurant toutefois naturel. Nous accordons plus d'importance dans l'ordre à : l'extraction des événements, puis l'extraction des événements exactes, et enfin la vitesse de traitement :
– $\phi_{\mathrm{MAG}}(f,f') = 1000$ ssi f a extrait un événement et pas f' ;
– $\phi_{\mathrm{MAG}}(f,f') = 100$ ssi les deux ont extrait un événement, mais l'extraction de f est de meilleure qualité (section 6) que celle de f' ;
– $\phi_{\mathrm{MAG}}(f,f') = 10$ ssi les extractions étaient de qualité similaire, ou aucune extraction a eu lieu, mais f a été plus rapide par au moins $margin$ secondes ;
– $\phi_{\mathrm{MAG}}(f,f') = 0$ ssi f et f' ne tombent dans aucun des cas précédents. Les valeurs 0,10,100,1000 reflètent une différence de magnitude entre l'importance des différents critères, sans besoin de les régler plus précisément.

Nous avons réalisé un jeu de tests étendu avec QL, DOM et MAG, en faisant varier les paramètres γ, ϵ, et λ (voir la section 3). Nous avons exécuté les IA non-formées sur un ensemble de 5 000 documents GTD (présentés dans le même ordre, et rencontrés une seule fois dans tous les tests). Les actions étaient choisies parmi dix *gazetteers*, les services *Tika, NGramJ, GATE, Geo*, et *STOP*. La qualité du traitement de chaque document a été mesurée comme décrit en section 5 pour toutes les approches. Nous insistons sur le fait que la qualité est donc mesurée avec le même critère (numérique) que celui utilisé pour donner un récompense à QL, et ceci pour les trois approches ; DOM et MAG reçoivent donc un feedback à première vue peu informatif vis-à-vis du critère à optimiser, demandant donc un effort cognitif faible aux experts.

Évidemment, comme les IA apprenaient de zéro, la politique initiale était de mauvaise qualité. Nous nous sommes intéressés au temps requis (en termes de nombre de documents traités) pour atteindre une bonne politique.

Le paramètre α a été réglé comme dans Gilbert *et al.* (2016), *i.e.* diminuant avec le nombre de visites de la paire état / action courante. Nous avons exécuté 36 combinaisons de tests en faisant varier les autres paramètres, afin d'obtenir un ensemble de résultats étendu, et de mesurer la robustesse vis-à-vis du choix des paramètres :
– les algorithmes QL, DOM, et MAG,
– le paramètre EG ϵ divisé par 2 après 2500 ou 1000 documents,
– $\gamma = 0{,}9$ et $\gamma = 1$,
– $\lambda = 0$, $\lambda = 0{,}95$ et $\lambda = 1{,}0$.

La figure 9 montre les courbes les plus intéressantes et représentatives. Nous avons tracé uniquement la qualité d'extraction lorsque l'IA a suivi sa meilleure politique pendant tout le traitement (elle a exploité) avec la ligne bleu en pointillés, et celle de la chaîne « experte » (ligne continue rouge) [2]. Pour une meilleure lisibilité, nous avons lissé les courbes en prenant les moyennes sur les ensembles de 50 documents avec le même ordre de traitement.

QL a donné d'excellents résultats, mais a montré une certain sensibilité au changement de paramètres. $\gamma = 0{,}9$ donne d'excellents résultats avec $\lambda = 0{,}95$ (figure 9a), où l'IA apprend une politique « assez bonne » après seulement 250 documents (les événements sont correctement extraits, mais elle prend quelques secondes de plus que la chaîne « experte »), et une politique optimale après $1\,200$ documents. $\gamma = 0{,}9$ donne aussi de très bons résultats avec $\lambda = 0$ (non montré). Cependant, avec $\gamma = 0{,}9$ et $\lambda = 1$, les résultats sont médiocres, et $\gamma = 1$ (figure 9b) donne de très mauvais résultats indépendamment de λ : l'IA a appris à s'arrêter très tôt, qui suggère une aversion au risque.

DOM, comme QL, a été sensible au choix de γ et λ. Avec $\gamma = 0{,}9$ et $\lambda = 1$ (non montré) et $\lambda = 0{,}95$ (figure 9f) les résultats sont très mauvais. Toutefois des résultats acceptables (non montré) ont été obtenus avec $\gamma = 0{,}9$ et $\lambda = 0$ pour la stratégie de réduction de ϵ la plus longue, et avec $\gamma = 0{,}9$ et $\lambda = 0$ ou $\lambda = 0{,}95$ pour la stratégie la plus courte. Avec $\gamma = 1$, les résultats allaient de bons à excellents, et la figure 9e montre les meilleurs résultats pour $\gamma = 1, \lambda = 1$, où une politique « assez bonne » est apprise après $1\,000$ documents et se stabilise à l'optimum après $3\,750$ documents (vérifié jusqu'à $10\,000$ documents — non montré).

MAG s'est révélé robuste au changement de paramètres, apprenant rapidement une politique au moins « assez bonne » dans tout les cas (*e.g.* la figure 9c).

Nous avons remarqué que le SSB Q-Learning (DOM ou MAG) apprend parfois une politique optimale mais que la performance se dégrade. Même si les événements

2. Ainsi, bien que les courbes rouges représentent toujours la même approche appliquée sur les mêmes documents, elles n'ont pas toujours la même allure puisque seuls des extraits en sont présentés.

sont toujours bien extraits, le traitement devient trop lent (voir les figures 9g, 9h) : l'IA apprend qu'un appel à GATE depuis un état donné produit une bonne récompense, et si γ et λ ne sont pas correctement réglés, bien que cela prenne plus de temps, elle commence à préférer cette action à l'arrêt.

Enfin, nous avons constaté une légère amélioration des résultats avec une réduction plus rapide de ϵ (tous les $1\,000$ documents vs tout les $2\,500$), c'est-à-dire, avec globalement moins d'exploration.

En résumé, nous nous attendions à ce que QL soit plus efficace que MAG, et MAG plus efficace que DOM, étant donné la quantité d'information qu'ils reçoivent. Nous constatons, cependant, que QL peut donner d'excellents résultats, mais est sensible à la variation de paramètres et dépend du *feedback* numérique. DOM ne nécessite qu'un *feedback* purement ordinal, et pourtant, avec les bons paramètres, est capable d'apprendre des politiques optimales, montrant la faisabilité de cette approche. MAG se révèle la meilleure approche dans un contexte industriel, combinant les avantages des deux méthodes : robustesse au choix des paramètres, utilisation du *feedback* essentiellement intuitif, et capacité à apprendre très rapidement une politique optimale.

9. Conclusion et perspectives

Nous avons modélisé une chaîne de traitement de documents sous forme de processus de décision markovien, que nous avons résolu en utilisant l'apprentissage par renforcement. Pour implémenter cela, nous avons développé l'application modulaire *BIMBO (Benefiting from Intelligent and Measurable Behaviour Optimisation)* dans laquelle nous pouvons « brancher » différents algorithmes d'apprentissage par renforcement, services *web* et modèles afin de mesurer leur impact sur l'apprentissage.

Nous avons établi que notre approche donne de bons résultats avec un *feedback* numérique sporadique. Nous avons ensuite intégré une fonction de récompense formalisant des préférences utilisateur exprimées naturellement, permettant d'obtenir d'aussi bons résultats, tout en exigeant moins d'efforts cognitifs pour définir le *feedback*. Nous avons ainsi montré qu'il est possible d'aboutir à une chaîne de traitement capable de s'améliorer sans intervention ou réglage par un utilisateur humain, et qui obtient son *feedback* de façon non intrusive.

Dans cet article, les ensembles d'états et d'actions ont été choisis pour démontrer la validité de notre approche. La prochaine étape consiste à produire des chaînes plus complexes, en ajoutant des services alternatifs (par exemple, la traduction), en étendant l'ensemble d'états (par exemple avec la liste complète de langues), et en introduisant une gamme de documents d'entrée plus large. Notre objectif est de montrer que le système est capable de construire différentes chaînes pour différents types de documents. Même avec cet espace d'états/d'actions plus large, le temps de calcul ne sera pas un obstacle avec un algorithme de type Q-Learning, où les calculs sont instantanés à chaque pas de temps.

La plateforme *BIMBO* sous-tend notre application, mais peut également être utilisée pour l'étude de différentes méthodes d'amélioration continue, et/ou pour l'évaluation des approches principales de RL, de l'efficacité de différents algorithmes, et de l'impact de la variation des paramètres, afin de démontrer l'utilité de ces méthodes en milieu industriel.

Notre approche, aussi, peut être utilisée pour d'autres traitements, par exemple la reconnaissance des objets dans une image. En effet, plusieurs types de capteurs et de nombreux algorithmes existent pour cela, et il n'est pas encore évident de les combiner optimalement. Nous sommes en train d'appliquer notre approche, en utilisant la plateforme *BIMBO* et une variation de Q-Learning(λ), *Dora* (Nicart *et al*., 2016), aux tâches de segmentation (le pré-traitement de l'image) et de la *ROC* (reconnaissance optique de caractères). Notre but est la détection des éléments d'intérêt, pour suggérer automatiquement des titres depuis les pages de garde des documents. Spécifiquement, nous avons utilisé les caractéristiques des pages, leurs images, et les paramètres utilisés par les services pour définir les états du MDP. Les actions consistent à choisir les valeurs de ces paramètres, ou à envoyer l'image au traitement. Nous avons déjà constaté des résultats très encourageants avec un espace de plus d'un million d'états.

Remerciements

Les auteurs veulent remercier Hugo Gilbert pour les fructueuses discussions sur les feedbacks *qualitatifs, ainsi que les reviewers anonymes d'IC2015 et de la RIA pour leurs retours constructifs.*

Bibliographie

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*, vol. 7524, p. 116–131. Springer Berlin Heidelberg.

Amann B., Constantin C., Caron C., Giroux P. (2013, mars). 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*, vol. 1500, p. 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*, vol. 3, p. 213–231.

Bratko I., Suc D. (2003). Learning qualitative models. *Artificial Intelligence*, vol. 24, n° 4, p. 107.

Busa-Fekete R., Szörényi B., Weng P., Cheng W., Hüllermeier E. (2014, décembre). Preference-based reinforcement learning: evolutionary direct policy search using a preference-based racing algorithm. *Machine Learning*, vol. 97, n° 3, p. 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.

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. *et al*. (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. 1.

Fishburn P. C. (1984). SSB utility theory: an economic perspective. *Mathematical Social Sciences*, vol. 8, n° 1, p. 63 - 94. Consulté sur http://www.sciencedirect.com/science/article/pii/0165489684900611

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*, vol. 54, p. 159–192.

Fromherz M. P., Bobrow D. G., De Kleer J. (2003). Model-based computing for design and control of reconfigurable systems. *AI magazine*, vol. 24, n° 4, p. 120.

Fürnkranz J., Hüllermeier E., Cheng W., Park S.-H. (2012, octobre). Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine Learning*, vol. 89, n° 1-2, p. 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. (2016). *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)

Hage W. R. van, Malaisé V., Segers R., Hollink L., Schreiber G. (2011, juillet). Design and use of the Simple Event Model (SEM). *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, n° 2, p. 128–136.

Karami A. B., Sehaba K., Encelle B. (2014, mai). 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*, vol. 225, p. 24–50.

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

Loftin R., Peng B., MacGlashan J., Littman M. L., Taylor M. E., Huang J. *et al*. (2016-01). Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous Agents and Multi-Agent Systems*, vol. 30, n° 1, p. 30–59.

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

Nicart E., Zanuttini B., Grilhères B., Praca F. (2016). Dora Q-learning - making better use of explorations. In D. Pellier (Ed.), *Proc. 11es journées francophones sur la planification, la décision et l'apprentissage pour la conduite de systèmes (jfpda 2016).*

Nicart E., Zanuttini B., Grilhères B., Giroux P. (2015). Amélioration continue d'une chaîne de traitement de documents avec l'apprentissage par renforcement. In *Actes des 26es journées francophones d'Ingénierie des Connaissances (IC 2105).*

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*, vol. 2, n° 1, p. 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*. Thèse de doctorat non publiée, Amsterdam.

Rodrigues F., Oliveira N., Barbosa L. (2015). Towards an engine for coordination-based architectural reconfigurations. *Computer Science and Information Systems*, vol. 12, n° 2, p. 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*. Thèse de doctorat non publiée, 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*, vol. 4, n° 1, p. 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*, vol. 1, n° 1978, p. 79–98.

Watkins C. J. C. H. (1989). *Learning From Delayed Rewards*. Thèse de doctorat non publiée, 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*.