apprentissage de séquences et extraction de règles de réseaux récurrents : application au traçage de schémas techniques.

Ikram Chraibi Kaadoud

THÈSE

PRÉSENTÉE A

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

par **Ikram Chraibi Kaadoud**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ: INFORMATIQUE

Apprentissage de séquences et extraction de règles de réseaux récurrents : application au traçage de schémas techniques.

Date de soutenance : Mars 2018

Devant la commission d'examen composée de :

Frédéric Alexandre	Directeur de recherche, Inria	Directeur
Nicolas Rougier	Chargé de recherche HDR, Inria	Co-Directeur
Michel Paindavoine	Professeur des Universités, Université de Bourgogne	Rapporteur
Arnaud Revel	Professeur des Universités, Université de La Rochelle	Rapporteur
Anke Brock	Enseignante-chercheuse, ENAC	Examinatrice
Helene Sauzeon	Professeure des Universités, Université de Bordeaux	Examinatrice
Hervé Frezza-Buet	Professeur, Centrale Supelec	Examinateur

– 2018 –

Résumé Les connaissances implicites d'individu s'acquièrent selon deux moyens. Le premier consiste en la répétition de séquences, ce qui permet à l'individu d'extraire implicitement des régularités. Le second moyen est une migration de connaissances explicites en connaissances implicites dut au développement d'une expertise. Il s'agit dans les deux cas d'apprentissage implicite. Dans nos travaux, nous avons souhaité nous pencher sur les séquences de composants électriques et notamment la problématique d'extraction des règles implicites dans ces séquences, aspect important de l'extraction de l'expertise métier à partir des schémas techniques. Nous nous plaçons dans le domaine connexionniste, et nous avons en particulier considéré des modèles neuronaux capables de traiter des séquences. Nous avons implémenté deux réseaux de neurones récurrents : le modèle de Elman, le Simple Recurrent Network, et un modèle doté d'unités LSTM (Long Short Term Memory). Nous avons évalué ces deux modèles sur différentes grammaires artificielles (grammaire de Reber et ses variations) au niveau de l'apprentissage, de leurs capacités de généralisation de celui-ci et leur gestion de dépendances séquentielles. Finalement, nous avons aussi montré qu'il était possible d'extraire les règles encodées (issues des séquences) dans le réseau récurrent doté de LSTM, sous la forme d'automate. Le domaine électrique est particulièrement pertinent pour cette problématique car il est plus contraint (combinatoire plus réduite) que la planification de tâches dans des cas plus généraux comme la navigation par exemple, qui pourrait constituer une perspective de ce travail.

Title Sequence learning and rules extraction from recurrent neural networks: application to the drawing of technical diagrams

Abstract Implicit knowledge is acquired in two ways. The first consists in the repetition of sequences, which allows the individual to extract implicitly regularities. The second way is a migration of explicit knowledge into implicit knowledge during the development of an expertise. In both cases, it is implicit learning. In our work, we endeavor to observe sequences of electrical components and in particular the problem of extracting rules hidden in these sequences, which are an important aspect of the extraction of business expertise from technical drawings. We place ourselves in the connectionist domain, and we have particularly considered neuronal models capable of processing sequences. We implemented two recurrent neural networks: the Elman model and a model with LSTM (Long Short Term Memory) units. We have evaluated these two models on different artificial grammars (Reber's grammar and its variations) in terms of learning, their generalization abilities and their management of sequential dependencies. Finally, we have also shown that it is possible to extract the encoded rules (from the sequences) in the recurrent network with LSTM units, in the form of an automaton. The electrical domain is particularly relevant for this problem. It is more constrained with a limited

combinatorics than the planning of tasks in general cases like navigation for example, which could constitute a perspective of this work.

Keywords Recurrent Neural Networks, LSTM, Sequence Learning, Rules Extraction, Technical diagrams

Mots-clés Réseaux de neurones récurrents, LSTM, Apprentissage de séquences, Extraction règles, schémas techniques

Laboratoire d'accueil Inria Bordeaux Sud-Ouest, 200 Avenue de la Vieille Tour, 33405 Talence

Remerciements

Nous piétinerons éternellement aux frontières de l'inconnu, cherchant à comprendre ce qui restera toujours incompréhensible. Et c'est précisément cela qui fait des nous des hommes

Isaac Asimov

A la fin de la thèse, un exercice difficile survient : celui de rendre hommage aux personnes qui ont contribué au succès de celle-ci, des travaux menés et de la rédaction du manuscrit. C'est donc maladroitement que je m'essaie à cet exercice ici, dans les quelques lignes qui suivent ...

A mes parents, qui m'ont accompagné dans cette aventure, à mes sœurs, mes modèles de persévérance, à mes tantes, qui se sont déplacées pour ma soutenance malgré la peur de l'avion, à ma famille bordelaise, aux êtres qui nous ont quitté, et à qui je penserais toujours...Et à tous ceux encore là, je dis un grand merci ! Il va de soi que je ne serais pas là où j'en suis sans vous, sans les valeurs que vous m'avez transmises et le courage que vous m'avez insufflé.

A ma force tranquille qui était tout simplement là et qui m'a soutenu tout au long de mes péripéties, merci.

Je remercie également les membres passés de Mnemosyne, parmi lesquels Charlotte, Fabien et Maxime qui m'ont accueilli et donné envie de faire de la recherche.
Je remercie également les membres actuels de Mnemosyne pour leur présence et leurs accompagnements, Thalita, Bhargav, Silvia, Antony, Remya, Aurélien, Basile et les autres.
Je remercie spécialement, André Garenne et Chrystel Plumejeau, pour m'avoir si souvent écouté et conseillé ! A chaque question, respectivement, sur le devenir des chercheurs ou les démarches administratives, vous avez été là !
Aux membres de l'INRIA BSO et de l'IMN avec qui j'ai eu le plaisir de discuter et d'échanger, je dis merci ! Les petites rencontres du quotidien font la beauté de la vie !

A tous les doctorants de l'association Aquidoc, grâce à qui je me suis sentie moins seule dans les moments d'incertitude, je vous remercie pour le soutien psychologique et les échanges formidables sur la recherche et la vie des doctorants que nous avons pu avoir.

A tous ceux qui me font le bonheur de leur amitié, je ne vous oublie pas et à ceux-là je dis merci !
A tout ceux que j'ai oublié de citer, mais qui ont été là, je m'excuse et je vous remercie.

A Alexandra, Maud, Brice, Meryl et Laure avec qui j'ai traversé le master de sciences cognitives et ces trois années, je vous dois énormément. Merci !

J'adresse mes remerciements les plus sincères à l'entièreté de l'équipe Algo'Tech de Bidart et de Paris, passé et présent : Anne-Cécile, France, Lize, Andoni, Jean-philipe, Maher (et Neila), Sylvain, et tous les autres ! Mais surtout merci à Didier et M. Jean-Michel Petolat, pour leur aide dans la compréhension des problématiques métier et la qualité des échanges que nous avons eu ! Votre accueil à tous, vos explications, votre patience, votre accompagnement dans cette aventure, et surtout votre confiance ont grandement contribué au succès de ma thèse !

Enfin, il va de soi que je n'en serais pas là sans mes 3 Senseis ! Que dis-je mes mentors : Frédéric Alexandre, Nicolas Rougier, mes directeurs de thèse, et Thierry Vieville, mon maitre Yoda ! Je pense que des remerciements ne sauraient exprimer ma gratitude pour l'opportunité que vous m'avez offert de réaliser ma thèse auprès de vous dans le domaine de l'informatique cognitive ! Votre confiance, votre ouverture d'esprit, et votre encadrement et votre patience, ont fait de moi la scientifique et l'humaine accomplie que je suis aujourd'hui !

Si l'on était dans un jeu vidéo, je m'assurerais de sauvegarder ma partie à ce stade, afin de ne jamais oublier les personnes qui ont fait ces trois années ...

A vous tous je dis tout simplement merci.

Saving game ...

Loading next level...

Table des matières

Introduction générale

Table des matières

La gestion des connaissances (KM pour Knowledge Management) pour une organisation se définit comme "le management des activités et des processus destinés à amplifier l'utilisation et la création des connaissances au sein d'une organisation selon deux finalités complémentaires fortement imbriquées : une finalité patrimoniale (préservation, réutilisation et actualisation des connaissances) et une finalité d'innovation durable (création active des connaissances individuelles et de leur intégration au niveau collectif dans une organisation)" [Grundstein, 1995]. Axe de développement stratégique pour les entreprises, l'essor du KM ces dernières années est dû majoritairement à celui des techniques de Machine Learning allié à celui du Big Data, faisant de la collecte des données un réel enjeu économique et sociétal [Serrour *et al.*, 2006; Wirtz *et al.*, 2014; Fischer *et al.*, 2015; Spiekermann *et al.*, 2015; Jin *et al.*, 2015; Zaoui et Mebarki, 2017].

Toutefois, il faut souligner deux points. Premièrement, la bonne récolte des données, ainsi que le fait d'avoir un volume suffisant, n'implique pas une bonne exploitation ou un résultat pertinent pour une entreprise. En effet, l'enjeu majeur actuel des entreprises est celui du choix de la bonne donnée parmi toutes celles accumulées, à savoir celle qui permet de synthétiser l'information pertinente ou de prendre la bonne décision. Deuxièmement, les systèmes d'information actuels (ensemble organisé de ressources qui permet de collecter, stocker,

traiter et distribuer de l'information) ignorent souvent la dimension humaine du processus de création des données [Crié, 2003]. Or, la connaissance n'est pas dans les données mais dans l'utilisateur [Malhotra, 2000] puisque quelles que soient les conclusions des algorithmes de fouille de données, c'est l'opérateur humain qui les valide ou non [Alvarez-Macias *et al.*, 2004].

Parmi les industriels confrontés à cette problématique de gestion des connaissances se trouve l'entreprise Algo'Tech, éditrice de logiciel de conception et de dessin assisté par ordinateur dans le domaine de la schématique électrique. Analyser et comprendre les données afin d'extraire le métier (comprendre les règles, les missions et les contraintes du métier) de ses clients sont des tâches auxquelles cette entreprise est confrontée quotidiennement. Dans ce domaine très technique où les méthodes de Big Data ne peuvent pas être mise en place, pour des raisons que nous expliquerons ultérieurement, Algo'Tech comme beaucoup d'autres entreprises, se doit de trouver des solutions innovantes pour rester en course. Dans nos travaux ci dessous, et à travers le cas de l'entreprise Algo'Tech, nous proposons une approche centrée sur la dimension humaine et l'organisation cognitive des connaissances, afin d'extraire les connaissances métier de documents techniques.

Mémoire et connaissances dans la fonction de raisonnement

Le raisonnement fait partie des fonctions exécutives chez l'humain. Il s'agit d'un processus cognitif de haut niveau qui permet à un individu de s'adapter à son environnement en fonction des informations perçues, de son expérience et de son objectif [Dubuc, 2002]. Cela permet par exemple de faire face à des situations complexes, d'interagir avec son environnement, ou encore d'établir des plans. Lors du processus de raisonnement, deux types de connaissances sont sollicitées : les connaissances explicites et implicites [Dienes et Perner, 1999].

Les connaissances explicites [Smith, 2001; Dienes *et al.*, 1991] sont l'ensemble des connaissances qu'un individu peut mémoriser et récupérer consciemment puis exprimer par le langage. Elles sont souvent apprises sous forme verbale. Elles sont associées à la mémoire explicite (figure 1) qui concerne l'ensemble des événements biographiques (mémoire épisodique) et la mémoire des concepts et des objets (mémoire sémantique) [Squire, 1992; Squire et Zola, 1996]. Ces connaissances, formalisables et transmissibles, sont exprimables par l'individu et peuvent donc être stockées sur un format extérieur (document, rapports, etc.). Elles sont considérées facilement extractibles [Nonaka et Takeuchi, 1995].

Les connaissances implicites sont non exprimables et sont celles dont l'individu n'est pas conscient [Schacter, 1987]. Elles sont liées à la mémoire implicite (figure 1). L'apprentissage de telles connaissances est implicite : c'est par l'imitation, la pratique, le vécu et l'expérience, qu'un individu acquiert ce type de connaissances. Elles regroupent les connaissances procédurales (liées par exemple aux capacités motrices comme faire du vélo), celles liées à la mémoire perceptive (mémoires des formes, des couleurs, des odeurs, des sons), celles des réflexes et des apprentissages non-associatifs comme l'habituation et la sensibilisation [Squire, 1992; Squire et Zola, 1996]. Ces connaissances sont importantes car elles sont associées au vécu d'une personne et ont une incidence directe sur le raisonnement. Dienes et Berry [1997] présentent la connaissance implicite comme inflexible, inaccessible et liée aux caractéristiques du matériau (i.e. la nature de l'information) utilisé lors de l'apprentissage.

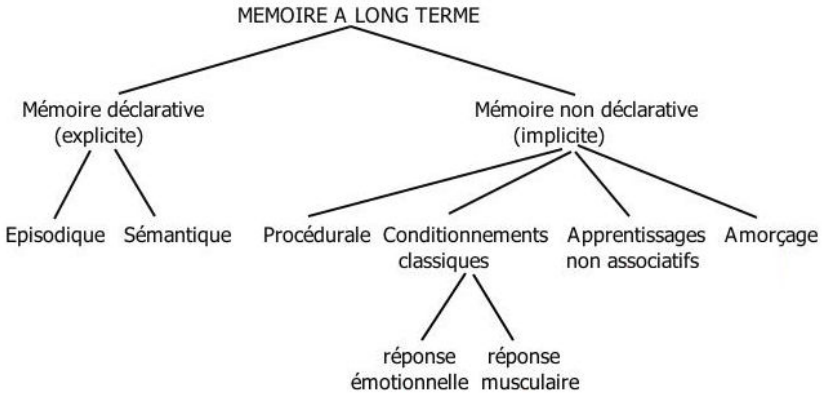


FIGURE 1 – Une taxonomie des différentes mémoires. Image adaptée de Squire et Zola [1996]

Comme nous le verrons par la suite, la compréhension de ce type de connaissance, son acquisition et son extraction ont fait et font encore l'objet de nombreux travaux de recherche. Il existe notamment un paradoxe dans cette connaissance implicite, qu'il convient de souligner : c'est à la fois la plus recherchée (liée à l'individu, elle porte plus d'information que des données [Malhotra, 2000; Crié, 2003], mais c'est aussi la moins accessible. Or dans le domaine de l'extraction de l'expertise, qui nous intéresse dans le cadre de nos travaux, c'est la gestion de l'ensemble des connaissances implicite et explicite qui représente un axe stratégique pour les entreprises. L'extraction de l'expertise est d'autant plus complexe qu'il existe trois stades d'acquisition des compétences (que nous détaillerons ultérieurement) en fonction desquels, les experts peuvent ou non verbaliser leurs connaissances [Hoffman, 2014]. Ainsi avant de décrire plus en profondeur cette problématique d'extraction de l'expertise, il faut d'abord s'intéresser à la mémoire implicite et l'apprentissage implicite dans leur ensemble.

Nous les présentons plus en détail dans la section suivante.

Mémoire et apprentissage implicite

La mémoire implicite se distingue de la mémoire explicite d'une part au niveau de l'apprentissage utilisé pour acquérir les connaissances et d'autre part au niveau de l'utilisation de ces connaissances. La majorité des preuves comportementales d'un système de mémoire implicite est basée sur des travaux dans lesquelles l'expérience conduit à une modification des performances sur une tâche donnée sans que les participants soient conscients d'avoir appris quoi que ce soit [Ettlinger *et al.*, 2011].
Le langage est un exemple typique d'apprentissage implicite : c'est par la pratique (répétition et imitation) que le langage s'apprend chez l'enfant (langue maternelle), et non par l'apprentissage explicite des règles grammaticales [Oudeyer, 2009]. Ce phénomène d'apprentissage implicite a été étudié pour différentes tâches appliquées à différents types d'informations (grammaire ou musique) et différentes populations (adulte et enfants)[Reber, 1967; Servan-Schreiber *et al.*, 1988; Cleeremans et McClelland, 1991; Cleeremans, 1993; Gasparini, 2004; Gombert, 2006; Nadeau et Fisher, 2011; Reber, 2013].

Witt [2010b], qui a réalisé une revue des différentes définitions proposées à ce phénomène, suggère ainsi de considérer que le terme d'apprentissage implicite recouvre toutes les formes d'apprentissage qui opèrent à l'insu du sujet, sans que ce dernier soit conscient du fait qu'il est en train de modifier de manière stable son comportement. Il s'appuie pour cela sur la définition de Perruchet et Vinter [1998]. Selon eux, "*l'apprentissage désigne un mode adaptatif par lequel le comportement des sujets se montre sensible aux caractéristiques structurales d'une situation à laquelle ils ont été préalablement exposés, sans que l'adaptation qui en résulte soit due à une exploitation intentionnelle de la connaissance explicite des sujets concernant ces caractéristiques*".
Les circonstances de cet apprentissage sont considérées accidentelles plutôt que intentionnelles [Berry et Dienes, 1991; Dienes et Berry, 1997; Perruchet et Nicolas, 1998]. Considéré robuste et indépendant des mécanismes explicites, l'apprentissage implicite possède un bon maintien dans le temps, mais ne permet pas de transfert des connaissances assimilées à de nouvelles situations [Gasparini, 2004].

Les travaux relatifs à l'apprentissage implicite dérivent de trois domaines de recherche distincts : premièrement, la linguistique et psycholinguistique [Reber, 1967], deuxièmement, l'ergonomie [Broadbent, 1977], et troisiémement, l'étude du rôle de l'attention dans la mémoire et dans l'acquisition d'habiletés sensori-motrices [Nissen et Bullemer, 1987]. Reber [1967] a été le premier à

désigner le processus d'acquisition passif et automatique sous le terme d'apprentissage implicite. Il s'est inspiré pour cela des travaux du linguiste Miller et du psycho-linguiste Chomsky, portant sur les grammaires (ensemble de règles) à états finis pour élaborer une tâche standard de grammaire artificielle nommée désormais, la tâche de Reber. La grammaire représentée sous forme de graphe orienté se compose d'arcs et de nœuds avec la particularité que chaque arc est associé à la production d'un symbole. La figure 2 est un exemple de grammaire artificielle. Ainsi, effectuer un parcours de la grammaire artificielle selon les arcs dans l'ordre des flèches produit une séquence dite grammaticale (les séquences non grammaticales ne respectant pas les transitions de la grammaire). Dans ces travaux, Reber soumet ces séquences à un panel de sujets, qui doivent les classer selon qu'elles respectent la grammaire ou non. Les sujets avaient été informés que les séquences étaient générées par une grammaire. Il met en évidence que les performances pour cette tâche de classification sont supérieures au niveau de hasard bien que les sujets étaient incapables de justifier leur décision ou de décrire les règles [Chambaron-Ginhac, 2005].

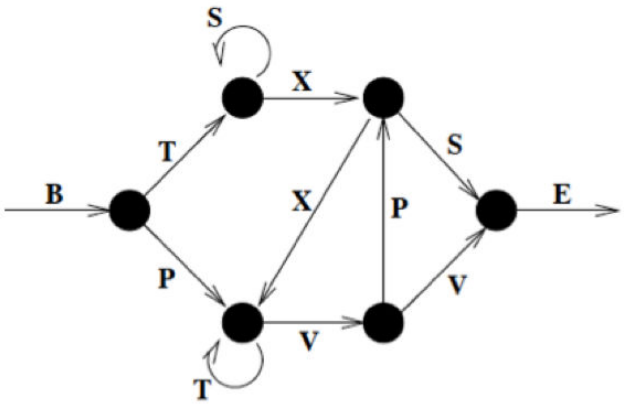


FIGURE 2 – Exemple de grammaire artificielle : B désigne "Begin" et E désigne "End". Image inspirée de Reber [1967], extraite Servan-Schreiber *et al.* [1988]

Parmi les nombreux travaux réalisés qui découlent de ceux de Reber, nous en soulignerons deux : le premier correspond à une tâche comportementale inspirée de celle de Reber et le second consiste à proposer un modèle de réseaux de neurones mimant le comportement observé. Les deux travaux sont abordés dans les articles suivants : Servan-Schreiber *et al.* [1988] et Cleeremans et McClelland [1991]. Pour la tâche comportementale, Cleeremans mesure le temps de réaction mis par les sujets pour faire un choix. Les sujets étaient face à un ordinateur sur lequel étaient présentés 6 éléments lumineux en ligne horizontale. Ils avaient 6 boutons à disposition, un correspondant à chaque élément lumineux. La tâche consistait à presser aussi vite et aussi précisément que possible l'une des six touches correspondantes pour reproduire l'élément présenté.

Les stimuli étaient générés en utilisant la grammaire à états finis utilisée par Reber. Certains d'entres eux n'étaient cependant pas "grammaticaux" et à chaque essai, il y avait 15% de chances de substituer un stimulus prescrit par la grammaire, par un autre choisi aléatoirement. Bien que les sujets ignoraient la nature séquentielle des stimuli, Cleeremans a observé une diminution du temps de réaction dans cette tâche de choix pour l'ensemble des sujets. Ces derniers devenaient ainsi de plus en plus sensibles à la structure séquentielle du matériel (lorsque la séquence était grammaticale) au cours de l'entraînement. Tout comme pour la tâche de Reber, les sujets étaient incapables de formaliser ou d'expliciter un quelconque apprentissage.

Les travaux de Reber et ceux de Cleeremans montrent que les individus soumis ou observant de manière répétée des régularités temporelles (pouvant correspondre à ce que l'on appelle des invariants ou des règles) finissent par apprendre ces dernières et les incluent aussitôt dans leur raisonnement et leur interaction avec l'environnement. Le tout sans en être conscient. Cela s'explique par le fait que l'apprentissage implicite extrait la structure statistique de l'environnement (ici les séquences) : les mécanismes d'apprentissage implicite produisent donc des performances améliorées à la suite de répétitions [Reber, 2013]. Par conséquent, il existe chez l'humain une capacité d'apprentissage séquentiel implicite qui se manifeste dès lors que l'individu est soumis de manière répétée à une même séquence d'événements et qui permet une amélioration de la performance sans que le sujet en soit conscient [Cleeremans, 1993]. Gombert [2006] affirme que "le moteur des apprentissages implicites est de nature fréquentielle" : la fréquence d'exposition à des éléments associés est fondamentale pour que le phénomène d'apprentissage implicite ait lieu et cela indépendamment de l'intention de l'apprenant [Perruchet et Pacton, 2004; Gombert, 2006; Lété, 2006; Deacon *et al.*, 2008; Nadeau et Fisher, 2011]. Forts de ces résultats, les auteurs de ces articles ont proposé un réseau de neurones récurrent simple (SRN pour simple recurrent network), pour comprendre ce phénomène d'apprentissage implicite. Ce modèle à trois couches, que nous décrirons par la suite dans nos travaux, leur a permis de mettre en évidence la capacité du réseau de neurones à encoder une représentation implicite des règles de la grammaire utilisée pour générer les séquences. Le SRN a ainsi été adopté comme cadre d'étude de l'apprentissage implicite.

La compréhension et l'étude de la mémoire implicite et des connaissances implicites sont aussi des questions qui se posent dans le domaine de l'expertise. En effet, l'acquisition de cette dernière implique une évolution des connaissances de l'individu du domaine de l'explicite vers l'implicite. Nous allons expliciter cela dans la section suivante.

Evolution des connaissances chez l'expert

Un expert, en général, est une personne sur-entrainée, qui a acquis des habilités particulières par son expérience et qui a notamment connu une migration de ses connaissances explicites à un savoir-faire implicite (figure 3) . Par exemple, un conducteur expérimenté ayant plusieurs années de conduite derrière lui est un expert en conduite par rapport à un jeune conducteur qui vient de décrocher son permis. Ce dernier fera appel à ses connaissances explicites du code de la route et du fonctionnement global de la voiture (positionnement des vitesses, positionnement des différentes pédales du véhicule, etc.) pour utiliser son véhicule. A contrario, le conducteur expérimenté conduira son véhicule de manière plus automatique ayant développé des habitudes de conduites [Mathern, 2012]. De nombreux travaux en psychologie considèrent ainsi l'expert comme un individu possédant un savoir-faire (connaissances et compétences particulières liés à ses expériences) et des ressources tacites par rapport à un novice [Coulet, 2014].

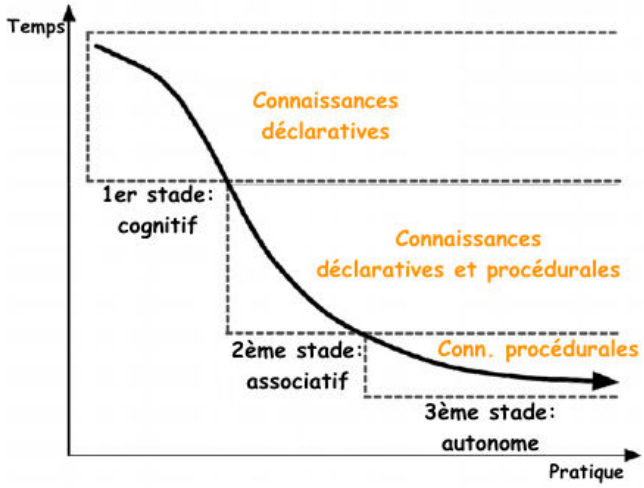


FIGURE 3 – Evolution du temps nécessaire à la résolution de tâche au fur et à mesure de la montée en compétence (pratique) selon trois étapes, basée sur la théorie de Fitts et Posner [1967], Anderson [1983],Rasmussen [1986] et VanLehn [1996]. Image adaptée de Kim *et al.* [2013]

Hoffman [2014] décrit 3 étapes d'acquisition de compétences (i.e de montée en expertise), proposées en premier lieu par Fitts et Posner [1967] et mises à jour par Anderson [1983] et Anderson [1987]. Le premier stade est qualifié de stade cognitif : les individus cherchent et récoltent des connaissances explicites de plusieurs sources sur le domaine considéré. Dans le cadre de la conduite, ces sources explicites sont le code de la route et les cours dispensés par une auto-école. Pour réaliser une tâche, l'individu sollicite ses connaissances explicites issues de sa mémoire à long terme en rapport avec la tâche à accomplir, sur

lesquelles il applique des procédures générales (i.e. des procédures qui peuvent être appliquées à n'importe quel domaine). Lors de cette première phase, les processus de prise de décision et de résolution de problèmes sont lents, fastidieux et sujets à erreurs [Anderson, 2005].

Le second stade est nommé le stade associatif et intervient au fur et à mesure qu'un individu devient de plus en plus compétent dans un domaine. L'utilisation répétée de connaissances explicites dans des situations données aboutit à la formation de connaissances procédurales spécifiques au domaine. Elles consistent en l'association directe entre des conditions spécifiques du domaine et l'action résultante, ce qui aboutit à l'apprentissage implicite de règle. La nécessité d'analyser des connaissances explicites est progressivement contourné : lorsque les conditions dans l'environnement correspondent aux conditions de la règle procédurale, l'action est automatiquement invoquée. Les processus longs et fastidieux de rappel (souvenir) des connaissances explicites et d'application de procédures générales à ces connaissances sont ainsi contournés [Charness et Campbell, 1988].

Le troisième stade est le stade autonome : les procédures deviennent automatisées. Les associations se renforcent, se spécialisent et/ou s'adaptent à des types particuliers de situations. Les connaissances procédurales deviennent très rapides et automatiques [Anderson, 1983; Charness et Campbell, 1988; Hoffman, 2014]. Durant ce stade, les connaissances procédurales simples deviennent composées ou remplacées par des connaissances procédurales plus complexes et inclusives. Ce dernier type de connaissances compressant (contenant) un large nombre de conditions d'instanciations et d'actions résultantes, induit une perte de capacité des individus à verbaliser leurs connaissances [Neves et Anderson, 1981]. Ainsi au fur et à mesure que les procédures deviennent composées et automatisées, la capacité de verbaliser les connaissances liées à la compétence décroit [Anderson, 2005]. Quand la réalisation d'une tâche devient complétement automatisée, le traitement ne nécessitant plus de ressources cognitives est autonome et inconscient (i.e implicite) [Dumais, 1981; Anderson, 1983; Carr *et al.*, 1982; Logan, 1988; Hoffman, 2014]. L'individu est alors un expert dans la résolution de tâche dans le domaine où il est monté en compétence. Dans le domaine de l'entreprise, ce processus est aussi pris en compte par Nonaka et Takeuchi [1995] qui le désigne par le terme d'internalisation (ou appropriation) des connaissances explicites en implicites (figure 4). La figure 3 illustre les trois stades d'évolution d'un expert et leur impact sur la capacité de verbaliser ses connaissances.

En résumé, un expert, dans un domaine donné, possède des connaissances et des procédures implicites qui lui permettent de choisir rapidement et automatiquement la bonne option dans différents contextes en analysant une situation

courante en fonction de ses expériences antérieures. Il évite ainsi l'utilisation d'un processus itératif et laborieux de prise de décision [Bootz et Schenk, 2014; Bootz, 2015, 2016].

Dans le domaine industriel où la gestion des connaissances est un axe concurrentiel pour les entreprises (la connaissance est l'équivalent d'un trésor pour ces dernières), la question de l'expert et de la transmission des connaissances représente un double enjeu pour les entreprises : au niveau des collaborateurs internes des entreprises, mais aussi au niveau des clients et les prospects (clients potentiels). Nous allons présenter dans la section suivante notre partenaire industriel Algo'Tech et sa problématique d'extraction des connaissances implicites. Nous reviendrons ensuite sur la problématique de l'extraction de l'expertise dans le contexte de entreprise.

Algo'Tech, contexte industriel

Nos travaux s'inscrivent dans le cadre d'une thèse CIFRE en partenariat avec l'entreprise Algo'Tech. Il s'agit d'un éditeur, distributeur et intégrateur de solutions logicielles dédiées aux métiers de l'électricité, de la gestion des documents techniques et de la gestion patrimoniale. Cette entreprise conçoit des logiciels de dessins et de conception assistés par ordinateur (respectivement désigné par DAO et CAO) pour des clients acteurs du bâtiment et de l'industrie - qui travaillent dans des domaines différents tels que la santé, l'aéronautique, la fabrication de machines (grues, convoyeurs) - avec pour but de leur proposer des solutions performantes de plus en plus automatisées.
Elle permet à ses clients qui sont des entreprises, grandes structures ou PME, de se constituer une base de données centralisant leur expertise métier à partir de l'étude de précédents dossiers. Cela leur facilite la maintenance, la mise à jour et l'utilisation de leurs connaissances pour répondre à leurs propres clients ou pour la transmission en interne (lors d'un recrutement par exemple). Ses solutions permettent de numériser ou migrer les précédents schémas électriques afin de les intégrer à la base de données et de les maintenir facilement par la suite. Les nouveaux schémas des clients sont ensuite générés en prenant en compte les spécificités de chaque client : non seulement les composants électriques, le matériel, mais aussi les habitudes de conception.
Actuellement, sur l'ensemble du processus d'extraction, c'est l'extraction du dernier élément - les habitudes de dessins - qui est le plus laborieux et énergivore en terme de temps et de ressources humaines. Pour chaque client souhaitant automatiser la génération de ses schémas électriques, le plus important est de retrouver son empreinte, (ou signature) dans les nouveaux schémas. Autrement dit, le but de l'entreprise est que les schémas soient tracés automatiquement, comme leurs clients l'auraient fait manuellement. En effet, chaque

client est capable de reconnaitre les schémas émis par sa société du fait de l'agencement des composants électriques. Assimilées à une signature implicite, retrouver ces habitudes de conception dans les nouveaux schémas est une condition nécessaire pour l'adoption des outils par les utilisateurs finaux (les collaborateurs de l'entreprise-cliente).

Pour réaliser l'extraction de cette signature et pour chaque nouveau client, un collaborateur d'Algo'Tech étudie les anciens schémas électriques, afin de les analyser et de préparer les bases de données de leurs outils. La difficulté majeure réside alors dans l'agencement des composants électriques. Le domaine électrique, industriel ou domestique, bénéficie d'un grand nombre de normes définies au niveau national mais aussi au niveau international. Cependant, et aussi surprenant soit-il, il n'y a pas de normes concernant la structure et la conception des schémas électriques, même si les symboles basiques et les notations restent sensiblement les mêmes. Chaque entreprise de ce domaine est de fait libre de tracer ses schémas selon sa propre expérience, son expertise et son savoir-faire. Les collaborateurs d'Algotech doivent donc passer en revue les schémas électriques, présentés souvent sous la forme de documents PDF, fournis par leurs clients et identifier ce qui constitue leurs signatures. Actuellement, ce processus d'étude et d'analyse se fait manuellement. Plusieurs facteurs expliquent cela : tout d'abord les documents fournis ne sont pas assez nombreux pour pouvoir mettre en place une technique de fouille de données automatisée. Le nombre de PDF fourni par des clients varie énormément en fonction de leur domaine : de quelques dizaines à une centaine, mais leur nombre reste difficilement compatible avec une approche de type Big Data. Par ailleurs, un schéma électrique pouvant être dessiné sur plusieurs feuilles au format A4 (chacune représentant alors une portion du circuit global), le nombre de feuilles, appelées folios, varie aussi entre une vingtaine de pages à plusieurs centaines pour certains clients. Ainsi il n'y a pas de garantie en termes de volume de données.

Un autre paramètre justifiant l'étude manuelle des schémas réside dans le type de PDF fourni par les clients. Les schémas électriques étaient traditionnellement réalisés sur machine puis imprimés sur support papier car ce format est plus simple à annoter pour les techniciens de terrains à chaque mise à jour des installations physiques (par exemple, la modification d'une installation électrique avec l'ajout de nouveaux composants). Pour une meilleure gestion de leur projets, de nombreuses entreprises-clientes se sont retrouvées à numériser leurs anciens schémas électriques papiers. Ainsi, parmi les PDF transmis à Algo'Tech pour analyse, nombre d'entre eux se composent d'ensemble d'images numérisées à partir d'anciens schémas. L'exploitation de tels documents nécessite alors l'utilisation de techniques de reconnaissance optique de caractères (OCR pour optical character recognition) dont les résultats dépendent fortement de la qualité des images numérisées. En fonction de celle-ci,

il sera possible d'extraire le texte contenu dans les images du PDF ou non [Fataicha, 2005; Belaïd et Cecotti, 2006]. Cette qualité d'image étant non garantie, les documents PDF restent complexe à traiter automatiquement. Enfin, un paramètre important est la nécessité d'extraire à la fois le texte et les symboles des schémas électriques : c'est l'alliance de ces deux informations qui permet d'extraire la connaissance métier du client : à la fois les habitudes de conception et les règles électriques.

La mise en évidence des habitudes de dessins nécessite une lecture attentive des schémas électriques. Au sein d'Algo'Tech, le collaborateur en charge de cette tâche, commence par le premier folio comportant une portion du circuit et se charge d'étudier la séquence de composants électriques sur chacun des fils (les fils s'étendant chacun sur un nombre variable de pages). C'est donc par une étude minutieuse des différents schémas électriques (agencement des portions de circuits, agencement des composants électriques, recensement des séquences de composants électriques) d'un même client, qu'Algo'Tech réussi à extraire les connaissances implicites et explicites contenues dans les documents techniques de leurs clients. Pour mener à bien ce processus d'extraction des connaissances techniques, des réunions d'analyses avec leurs clients sont nécessaires afin de confirmer, infirmer ou ajuster les informations extraites par Algo'Tech. Ce processus d'échange verbal amène l'entreprise-prestataire à faire face à la problématique d'extraction de l'expertise que nous décrirons par la suite. En résumé, Algo'tech fait donc régulièrement face à deux problématiques. La première est la compréhension des processus de planification et de raisonnement de leurs clients via l'étude des schémas électriques. La seconde est d'amener les experts de leurs clients à expliquer leur travail, autrement dit, de les faire expliciter leurs habitudes de travail implicites. Ce travail est donc difficile car cela implique d'accompagner les experts dans l'exploration de leurs propres habitudes et connaissances. Ce processus, est souvent l'opportunité pour le client de faire le point sur son expertise et ses méthodes de travail, en validant ou réformant certaines d'entre elles.

En quoi l'approche inspirée des processus cognitifs chez l'humain est-elle pertinente pour répondre à ces problématiques ? Notre hypothèse est la suivante : les schémas électriques et le processus de planification (impliquant le raisonnement) chez l'humain partagent des caractéristiques communes. Par exemple, lors de la réalisation d'un schéma électrique d'un bâtiment, il est possible d'approcher le problème selon une vision hiérarchique : d'abord un concept global (l'alimentation du bâtiment), puis ses sous-concepts (l'alimentation des différentes salles) et les sous-sous-concepts (l'alimentation des équipements dans chaque salle). Mais il est aussi possible d'adopter une vision modulaire du schéma et considérer tous les composants électriques relatifs à l'alimentation, puis tous ceux relatifs à la sécurité et enfin ceux relatifs aux

équipements. Enfin chaque schéma est réalisé en fonction de paramètres implicites (tel les règles et habitudes de travail) spécifiques à l'entreprise émettrice : une même alimentation pour un même équipement peut être représenté de manières différentes selon les habitudes d'entreprises différentes. L'ensemble de ces éléments permettent de faire un parallèle avec la fonction de planification chez l'humain. En effet, lors du processus de planification d'une tâche chez l'humain, ce dernier peut choisir d'adopter une approche hiérarchique (décomposer une tâche en sous-tâche et sous-sous-tâches) ou adopter une approche séquentielle et modulaire de la tâche (en regroupant les sous-tâches de même type puis en exécutant un type de sous-tâche puis l'autre). Mais surtout, il existe tout une dimension implicite qui va venir impacter son processus de planification et ses choix : ses expériences, son vécu et ses connaissances acquises. Deux personnes, mettront en place deux plans d'actions différents pour atteindre un même objectif. Ce parallèle entre les spécificités des schémas électriques et la fonction de planification chez l'humain justifie l'approche proposée dans nos travaux.

A travers la présentation de l'entreprise Algo'Tech, nous avons soulevé la question de l'extraction de l'expertise en entreprise. Dans le contexte industriel, il s'agit d'une problématique omniprésente liée à la pérennité des entreprises et à leur solidité [Barney, 1991; Prahalad et Hamel, 1999; Cowan *et al.*, 2000]. Nous allons expliquer cela dans la section suivante.

Extraction des connaissances et de l'expertise en entreprise

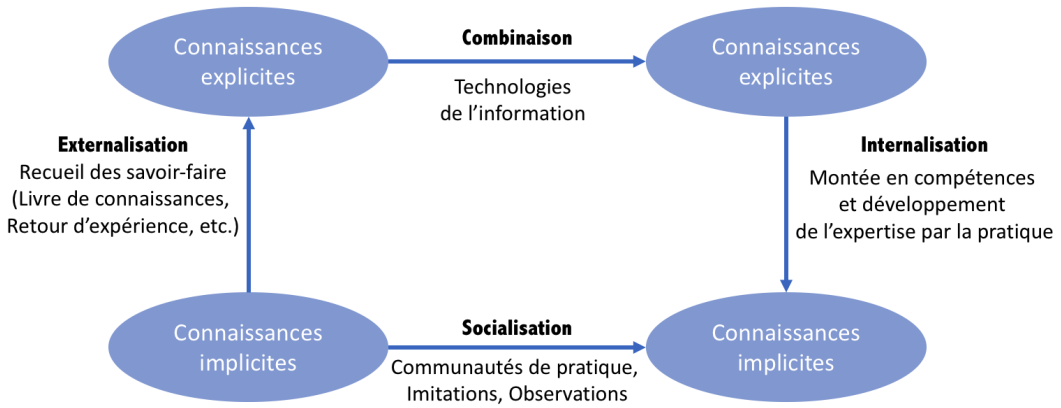


FIGURE 4 – Les quatre transformations des connaissances au sein d'une organisation d'après Nonaka et Takeuchi [1995]. Image adaptée de Viola [2005]

Une expertise partagée à tous les collaborateurs au sein d'une entreprise,

garantit le bon fonctionnement de cette dernière et prévient un ralentissement ou arrêt soudain lorsqu'un collaborateur clé s'absente. Il est donc essentiel que la raison des choix puisse être accessible et explicite. D'après Nonaka et Takeuchi [1995], les entreprises créent et utilisent des connaissances selon quatre moyens illustrés dans la figure 4. (1) La socialisation (ou le transfert direct) permet le partage des connaissances tacites par l'expérience (observation et imitation). (2) L'externalisation (ou explicitation) permet de rendre des connaissances implicites, explicites par le recueil des savoir-faires des individus de l'entreprise-cliente. (3) La combinaison consiste à recueillir les connaissances explicites en les décomposant en connaissances élémentaires, (4) l'internalisation (ou appropriation) intervient lorsque la connaissance explicite devient une connaissance implicite par exemple lors de l'acquisition d'une formation suivie de pratique [Serrour *et al.*, 2006; Viola, 2005; Hoffman, 2014]. Dans ce qui suit, nous nous focaliserons surtout sur les processus d'externalisation selon deux moyens : un échange directement avec les experts métier du domaine ciblé, et/ou via l'échange d'une documentation informant sur le domaine, ses procédés, ses règles et ses normes [Ermine, 2001].

Dans un contexte industriel, un expert a pour fonction d'analyser, de conseiller et de faire des recommandations de façon à améliorer l'efficience des solutions mises en places. Il est ainsi souvent l'interlocuteur à privilégier pour s'informer d'un domaine. Capable d'identifier les risques mais aussi les opportunités liées à son domaine de spécialité, il peut préconiser des plans d'actions et mener des projets complexes dans son domaine. Deux types d'experts se distinguent dans un même domaine : l'expert-débutant et l'expert-senior. Si le premier a acquis son expertise par l'apprentissage d'un enseignement, le second, pratiquant son métier depuis plusieurs années, a accumulé expériences de terrain et cas d'études concrets. Si cette distinction est importante à souligner, c'est qu'elle impacte aussi la représentation mentale des experts de leur métier. Lowe [1993] a démontré l'existence de différences entre ces deux types d'experts en utilisant une série de tâches de dessin. Il a ainsi confirmé qu'un expert-débutant possède une conception qui se concentre plus sur des aspects superficiels et généraux alors qu'un expert-senior, se focalise plus sur la structure sémantique profonde des éléments. Ces résultats ont été confirmés ensuite par Aimé et Charlet [2013]. Pour un domaine donné, les experts-débutants ont une conceptualisation de type plutôt textuelle, ce qui implique des concepts détaillés, analytiques, contrôlés et de bas niveau, alors que les experts-seniors ont une conceptualisation plutôt de type essentielle, plus floue, conceptuelle, intuitive, utilisant des prototypes et des concepts de haut niveau. Cela implique donc qu'une même connaissance peut être décrite et représentée différemment.

Cette différence de représentation possède surtout des répercussions sur les processus d'externalisation (processus 2) décrit par Nonaka et Takeuchi

[1995] (figure 4). En effet, parmi les moyens classiques d'extraction de l'expertise, l'entretien entre l'entreprise-prestataire (celle qui fournit le service) et l'entreprise-cliente (celle qui paie pour la prestation) arrive en tête. Cette rencontre, qu'elle soit virtuelle, téléphonique ou physique dans les locaux d'une des entreprises, est nécessaire pour échanger autour de l'expertise métier. Fonctions des moyens des entreprises, en terme de ressources financières et humaines, ces processus sont énergivores au niveau financier et humain, chronophages et potentiellement porteurs d'erreurs. Pour aider cette passation de connaissances, l'échange de documents techniques (portant sur l'aspect technique du métier), emails et compte-rendus est aussi privilégié. Si les deux derniers éléments sont importants pour apporter des précisions et répondre à des questionnements, l'information majeure se trouve surtout dans les documents techniques, car ce sont eux qui représentent des exemples concrets d'application de l'expertise à des problématiques données. Nous allons décrire dans la section suivante les moyens utilisés pour extraire la connaissance à partir de ce type de documents.

Application à l'analyse de documents

L'analyse des documents techniques est une partie importante du processus d'extraction de l'expertise. Ces documents sont souvent confiés à l'entreprise-prestataire pour analyse. La qualité et la quantité de ces documents techniques impactent fortement la capacité de compréhension du métier ciblé : ils doivent être exploitables, pertinents, d'actualité (respectant les dernières normes en vigueur) et surtout en relation avec la problématique ciblée. Ces documents techniques sont importants car ils sont porteurs d'informations. Il est important de distinguer une donnée d'une information à ce stade : d'après Chaudet [2009] une donnée est un élément brut qui n'a pas encore été interprété et qui est dénué de contexte, alors qu'une information est une donnée interprétée, à savoir, mise en contexte. Porteuse d'une sémantique, une information, lorsqu'elle est comprise est alors considérée comme une connaissance. Ainsi le but de l'analyse des documents techniques est d'extraire les connaissances qu'ils contiennent : les données mises en contexte deviennent donc des informations, qui une fois assimilées et comprises deviennent des connaissances.

Nonaka et Takeuchi [1995], deux experts de la question du management de la connaissance en entreprise, différencient deux formes de connaissances : tacite et explicite. Nous retrouvons ici les mêmes catégories abordées précédemment pour l'organisation des connaissances au niveau cognitif. Dans le domaine de la gestion des connaissances au sein des entreprises, les connaissances tacites désignent les compétences, les expériences, l'intuition ou encore les secrets de métiers qu'une personne acquiert au cours de sa vie professionnelle. Leurs

mises en pratique se fait sans intention et leurs transmissions se fait par l'imitation et l'imprégnation. La connaissance explicite (connaissance formalisée et transmissible sous forme de documents réutilisables) concerne généralement les projets, les clients, les fournisseurs et les processus. Il faut souligner cependant que la complexité majeure dans l'analyse de documents techniques réside dans la sélection des données : lesquelles sont pertinentes ? lesquelles sont déterminantes ? Selon le volume des documents techniques, la complexité de leur contenu, différentes approches de fouilles de données existent. Nous distinguons deux axes de fouilles des documents techniques : la fouille des images (l'image mining) [Haddad et Mulhem, 2001; Korczak *et al.*, 2005; Pham *et al.*, 2011] et la fouille du texte (le text mining) [Cherfi *et al.*, 2003; Nassirtoussi *et al.*, 2014].

L'image mining consiste en l'analyse et la reconnaissance d'images. Cette approche est généralement plébiscitée dans des domaines tels que l'ingénierie mécanique, l'étude des cartes de villes, les figures dessinées à la main, les cartes géographiques et les circuits électriques. Ses deux utilités sont la fouille de grandes collections d'images (exemple : mise en évidence des similitudes dans des images de satellites météorologiques) et l'extraction d'informations par l'analyse de grandes collections d'images et de données alphanumériques associées (exemple : les documents liés à l'imagerie médicale et les données des dossiers des patients). Pour plus de détails techniques sur les techniques liées à l'image mining, se référer à Zhang *et al.* [2001]. Cette approche, bien que répandue, possède cependant des contraintes qui limitent son adoption par toutes les entreprises, toutes tailles confondues. Parmi elles, le volume du corpus d'images et la qualité de celles-ci en sont les principales.

Le second axe de fouille de données applicable à des documents techniques est le text mining. Cette approche consiste à fouiller le texte extrait des documents afin d'extraire des connaissances selon un objectif donné [Neto *et al.*, 2000]. Les domaines d'applications sont par exemple la recherche d'information, le filtrage de documents, la sécurité [Gegick *et al.*, 2010] ou encore le domaine médical [Fang *et al.*, 2008]. Pour utiliser les techniques de text mining, deux étapes sont nécessaires : l'analyse des données, et l'interprétation de l'analyse [Tan *et al.*, 1999]. La première étape consiste à analyser les données selon l'objectif à atteindre afin de savoir lesquelles cibler : relation entre les mots, syntaxe recherchée, etc. La seconde étape consiste à identifier, sélectionner et extraire les textes correspondant à l'analyse. Par exemple, l'identification et la mise de coté des courriers indésirables dans les emails. Parmi les applications du text mining, nous soulignons l'importance de la recherche d'information au sein de documents ou de données selon les besoins d'un utilisateur [Tan *et al.*, 1999; Mooney et Bunescu, 2005]. Cette approche sera abordée dans nos travaux.

Tout comme pour l'image mining, le text mining est un domaine très actif où de nombreux travaux ont vu le jour ces dernières années [Mostafa, 2013; He *et al.*, 2013; Nassirtoussi *et al.*, 2015; Goyal, 2017]. La profusion des documents au format numérique (sites web, emails, contenu numérique, etc.) a engendré de nouveaux besoins en terme de capacité d'analyse des documents et de rapidité de traitement. La fouille de documents est par extension un domaine très dynamique [Singhal, 2001; Roche, 2011]. Il existe ainsi d'importants travaux dans les domaines du text mining et de l'image mining mais séparément. Or dans le cas de l'étude de documents techniques tels des schémas, le texte et l'image sont tous les deux porteurs d'informations.

Ces dernières années, de nouveaux axes d'analyse des documents techniques sont apparus. L'idée étant de ne plus se focaliser sur les données mais sur l'ensemble des documents : l'analyse syntaxique et l'analyse sémantique [Kembhavi *et al.*, 2016]. Par analyse syntaxique, nous désignons l'analyse de l'ensemble des règles régissant les documents techniques et par sémantique, l'analyse de la signification (le sens) des documents. L'analyse syntaxique consiste à mettre en évidence la structure d'un texte ou document. Le but est de mettre en évidence l'ensemble des règles de syntaxe combinées dans un texte. Dans le cas de diagrammes, il s'agira de détection des constituants et leurs relations syntaxiques. L'analyse ou interprétation sémantique vise à mettre en évidence le sens du document, son sujet (thématique). Pour un diagramme, cela consisterait à mettre en évidence les tâches de cartographie des constituants et leurs relations avec des entités sémantiques ou des entités réelles (concepts de mots). Appliquée aux documents techniques, l'alliance de ces deux méthodes permet de faire de l'interprétation de diagramme. Récemment, Kembhavi *et al.* [2016] se sont intéressés à ce problème d'interprétation en proposant une approche globale des diagrammes, indépendamment du domaine. Ils ont pour cela exploité un volume très important de données. Leur algorithme d'apprentissage de Deep learning a appris sur la base de 5 000 schémas. De tels travaux sont possibles grâce aux techniques de Deep learning et au Big Data. L'alliance de grands volumes de données avec des algorithmes et machines puissants rend possible l'interprétation de documents techniques. Or c'est ce qui en représente aussi les limites d'utilisation : le volume de données utilisé pour l'étude de Kembhavi *et al.* [2016] est bien plus important que ceux que la plupart des PME peuvent prétendre (environ entre 100 et 200 schémas pour leurs clients).

Lorsqu'il s'agit de documents techniques et notamment de schémas et diagrammes, les approches traditionnelles d'extraction des connaissances ne sont pas adéquates : la nécessité d'avoir de gros volumes de données à disposition ainsi que les moyens de les analyser ont induit la nécessité de rechercher de nouvelles solutions. Récemment de nouvelles approches ont émergées, notamment celles liées à une ingénierie centrée autour de l'homme : interface homme-

machine, ergonomie cognitive, management des connaissances, etc. Revenir à l'humain et comprendre son raisonnement afin de mieux comprendre la manière dont il le transpose dans les documents et diagrammes techniques semble être une solution prometteuse.

Contexte informatique actuel

Ces dernières années ont vu naître un engouement croissant pour l'intelligence artificielle. En 2013, IBM présentait un outil cognitif Watson, inspiré et mimant le raisonnement humain. Ce dernier a édité un livre de cuisine en 2015. En 2014, Amazon lance son assistant personnel intelligent nommé Alexa. En 2016, le logiciel de Google DeepMind a battu le champion européen de Go. En 2017, de nombreux constructeurs automobiles tel Tesla ou Toyota, travaillent sur le concept de voitures autonomes. Ces quelques événements ne représentent qu'une partie infime de l'ensemble des travaux et tests menés par des laboratoires et entreprises dans le domaine de l'Intelligence Artificielle (IA). Nous souhaitons ici illustrer le contexte informatique actuel dans lequel nous avons réalisé nos travaux. Nous aborderons les domaines qui nous ont inspirés et ceux où l'apprentissage séquentiel et l'extraction des connaissances sont importants : les réseaux de neurones, la gestion des connaissances et le langage naturel.

Face à la montée en puissance des différentes formes IA, une question revient souvent : que se passe-t-il dans les réseaux de neurones ? Ces IA sont vues comme des boites noires puissantes en terme de calcul. Capables d'apprendre et de traiter de très gros volume de données, elles ne "comprennent" néanmoins pas leurs données. Le fonctionnement de ces réseaux (les algorithmes et les mathématiques) est connu et compréhensible par la communauté scientifique, mais il est néanmoins difficile de comprendre ce qui se passe au sein des réseaux de neurones. Nombreux sont ceux qui s'interrogent sur les raisons des performances de ces réseaux. Par exemple, DARPA, l'agence du département de la Défense des États-Unis chargée de la recherche et développement des nouvelles technologies destinées à un usage militaire, a lancé son programme "Explainable Artificial Intelligence" (XAI) pour tenter d'apporter des réponses. Parmi les travaux qui s'intéressent à ces questions on retrouve notamment ceux de Geoffrey Hinton, un des pères du Deep Learning [Frosst et Hinton, 2017]. "Est-il possible d'observer et d'extraire ce qui se passe dans les réseaux de neurones ?" est ainsi un exemple de questions auxquelles des travaux, parmi lesquels les nôtres, tentent de répondre. Ce questionnement sur les IA, fait écho à celui sur le cerveau humain : que se passe-t-il au sein du cerveau lors de certaines tâches comportementales ? On observe de nombreuses tentatives d'expliquer le biologique par l'informatique. Par exemple, le SRN a été proposé

comme cadre d'étude pour modéliser l'apprentissage séquentiel chez l'humain [Servan-Schreiber *et al.*, 1989], ou encore des algorithmes de Deep learning ont été proposés pour étudier le cortex visuel [Cherfi *et al.*, 2003; Nassirtoussi *et al.*, 2014].

Autre domaine dynamique, celui de la gestion des connaissances. Comme souligné précédement, cette dernière est un capital non négligeable pour les entreprises. L'explosion des volumes de données ces dernières années avec le Big Data, alliée au développement d'algorithmes de machine learning de plus en plus performants (réseaux de neurones profonds liés au Deep Learning) ont précipité une course aux informations. Si la gestion des connaissances est un domaine stratégique c'est parce qu'il permet la création, le partage, la réutilisation et la représentation des connaissances au sein des structures, ce qui est sensé améliorer les performances et accroitre les innovations. Le développement rapide ces dernières années du domaine des ontologies (modèle de données représentatif d'un ensemble de concepts et des relations entre ces concepts dans un domaine)[Ding et Foo, 2002] et celui de la visualisation des données [Witten *et al.*, 2016] est la preuve de l'importance et l'intérêt grandissant pour le domaine [O'Neill, 2016]. La recherche d'informations est aussi un domaine très dynamique. Parmi les domaines d'applications, nous citerons l'analyse de sentiments à partir de tweets sur internet pour évaluer la réaction du public pour des films ou des événements en tout genre [He *et al.*, 2013; Goyal, 2017] ou encore les moteurs de recherches tels que Google et Bing [Amini et Gaussier, 2013]. Nous citerons aussi le PDF mining (fouille de documents PDF) : il s'agit d'un domaine de recherche qui traite de la problématique d'extraire des informations à partir de fichiers PDF.

Dans le domaine de l'apprentissage séquentiel, un domaine d'intérêt est celui du langage naturel. De nombreux travaux en recherche et en industrie s'intéressent à la question de l'interprétation (la sémantique) des textes. Comprendre un langage revient à décoder et interpréter des séquences de mots, qui eux même sont des séquences de symboles. Par ailleurs, chaque langage obéit à des règles explicites, les règles de grammaires (la syntaxe), mais aussi des règles implicites (les us et coutumes de chaque région, les expressions, le contexte, etc.). Cette proximité entre le langage naturel et notre problématique d'extraction des connaissances nous a souvent amené à regarder les travaux de ce domaine de plus près et à nous en inspirer. C'est donc dans ce contexte informatique riche, que nos travaux s'inscrivent afin de répondre à la problématique que nous décrivons dans la section suivante.

Problématique

A l'image de la programmation informatique, le traçage des schémas électriques est un domaine normé où il existe des règles établies. Pourtant dans le premier cas, il a été mis en évidence qu'il était possible de déterminer précisément l'auteur d'un code informatique en étudiant les caractéristiques implicites contenues dans ce dernier [Rosenblum *et al.*, 2011]. Dans le domaine des schémas électriques, nous savons qu'il existe une connaissance (l'expertise en l'occurence) implicite au vu des connaissances extraites par les collaborateurs Algo'Tech. Notre hypothèse est qu'il est possible d'extraire les connaissances implicites dans les schémas électriques d'un client donné, parmi lesquelles les règles de connectivité à partir des séquences de composants électriques. Nous nous focaliserons ainsi dans nos travaux sur cela et présenterons notre approche pour expliciter ces règles implicites.

D'un point de vue scientifique, la question qui se pose est celle de l'extraction sous un format explicite de connaissances qui ont été apprises implicitement. Comme nous l'avons vu précédemment, connaissance et expertise sont des notions transversales à de nombreux domaines de la recherche : cognition, knowledge management, IA et data mining. Répondre à la question posée, nécessite ainsi d'aborder précisément chacune de ces notions dans chacun de ces différents domaines, afin de déterminer les possibles ponts que l'on peut établir entre eux. Autrement dit cela revient à s'intéresser, selon différents axes, au cerveau en tant que modèle capable de manipuler et gérer à la fois l'information explicite et implicite, ainsi que de gérer l'apprentissage de séquences, et enfin d'extraire des régularités (règles) à partir de séquences.

Le premier axe, à partir duquel nous avons puisé nos connaissances pour étayer notre étude, est celui de la cognition : La question posée est celle de l'organisation et de l'évolution des connaissances liées à l'expertise d'un individu au fur et à mesure de sa pratique. Plus précisément, au niveau de la mémoire, c'est la capacité cognitive du cerveau à apprendre des régularités (règles) à partir de plusieurs séquences qui a été étudiée (mémoire implicite, apprentissage implicite et séquentielle). Le second axe d'étude est celui du knowledge management : la question était celle des moyens existants pour identifier et extraire les connaissances dans le but de les partager. Au niveau humain, l'organisation de l'expertise métier en entreprise et les moyens de l'extraire est une question qui s'est posée dès le début de nos travaux. Au niveau informatique, dans un premier temps, parmi les moyens existants dans l'IA notamment ceux relatifs au data mining, c'est la question des données qui s'est posée : quelles sont les données à prendre en compte ? et quels sont les moyens envisageable pour les extraire ? ont été deux questions importantes à soulever, sans oublier celle qui consiste à se demander comment transformer et représenter des données brutes

en connaissances. Dans un second temps, c'est le sujet de la mise en évidence de connaissances implicites à partir des données séquentielles que nous avons souhaité traiter en nous penchant sur les moyens automatiques actuels et futurs qu'il est possible de mettre en oeuvre pour cela.

La transdisciplinarité de la question scientifique de la mise en évidence de la dimension implicite des connaissances, nous amène à une approche connexionniste qui semble donc légitime pour répondre aux questions soulevées dans le domaine de la mémoire (apprentissage séquentiel) et dans le domaine de l'extraction des connaissances. Plus précisément, l'étude de réseaux de neurones récurrents, modèle bio-inspiré, semble pertinente. Parmi les modèles réalisant l'apprentissage séquentiel, ces réseaux de neurones ont été (et sont encore) l'objet de nombreux travaux visants à extraire les règles apprises qui serviront de base à nos travaux. Ils véhiculent en effet l'image de "boite noire" (les informations apprises ne sont pas explicites) ce qui rejoint notre problématique de mémoire implicite non accessible.

En résumé, les trois points importants de notre approche sont :

1. la transformation des données brutes en connaissances et l'identification de celles pertinentes pour l'apprentissage séquentiel : lesquelles sont porteuses de la connaissance implicite ?
2. le choix du modèle de réseau de neurones récurrent qui permet un apprentissage séquentiel en gérant les caractéristiques des séquences (contexte temporel et dépendances séquentielles) quelles que soit leurs tailles
3. les moyens d'extraire à partir des réseaux récurrents les règles (régularités) implicites contenues dans les séquences apprises

Ce travail de thèse aura une application au contexte industriel pour valider la transposabilité des travaux à des problématiques industrielles.

Contributions

Pour répondre aux questions soulevées précédemment, nous proposons trois contributions dans nos travaux.
La première vise à établir un procédé d'extraction et de représentation des connaissances techniques dans les schémas électriques. Nous décrirons comment construire une représentation selon différents niveaux d'abstraction à partir de ces documents techniques. Il s'agira de notre phase de fouille et de préparation des données.
Notre seconde contribution consiste à proposer un modèle neuronal d'apprentissage séquentiel. Nous décrirons particulièrement les différentes composantes

des séquences et leurs importances et nous réaliserons une comparaison entre différents modèles de réseaux de neurones.
Enfin, notre troisième contribution concernera l'extraction des règles encodées dans un modèle neuronal. Nous présenterons les différentes étapes qui visent à l'extraction et la représentation de manière explicite des règles implicites présentes dans des séquences.
Ce travail sera validé et mené grâce à des études expérimentales dont certaines utiliseront des données fournies par Algo'Tech. A travers ces trois approches, nous souhaitons ainsi proposer une solution générique pour la réalisation d'un processus d'extraction des connaissances à partir de schémas électriques.

Nous souhaitons aussi souligner ce que nos travaux de thèse n'ont pas pour but de faire. L'extraction des règles de connectivité électrique peut laisser penser que nous souhaitons potentiellement créer une ontologie (i.e. un modèle de données représentant les connaissances d'un domaine [Teulier *et al.*, 2005]) du domaine électrique global à partir de toutes les données extraites des différents clients (composants et relations entre eux) ou une ontologie du domaine électrique d'un client donné à partir des données extraites de ses schémas, comme cela a pu être le cas dans des travaux dans d'autres domaines [Aimé *et al.*, 2010; Charlet *et al.*, 2012; Bouzidi *et al.*, 2010]. Dans les faits, bien que ces deux idées soient très intéressantes, elles ne correspondent pas à la question que nous ciblons ici. Notre but est de trouver le moyen d'extraire une ontologie du métier d'une entreprise-client telle qu'elle l'utilise quotidiennement, avec les règles métier et ses habitudes de travail.

Structure du document

Dans le premier chapitre, nous présentons l'ensemble du travail de préparation et d'extraction des données réalisées en deux temps. En effet, plusieurs problématiques ont été soulevés : Comment exploiter les documents PDF au mieux ? Parmi l'ensemble des données extraites, quelles sont celles qui contiennent l'information implicite recherchée ? et enfin comment extraire les habitudes de travail à partir de plusieurs exemples de séquences ? Pour y répondre, dans un premier temps nous décrivons l'approche adoptée pour la mise en évidence de la structure hiérarchique et modulaire des connaissances contenues dans les schémas électriques. Ensuite, dans un second temps, nous décrirons le travail réalisé proprement dit : celui qui nous permet d'extraire les règles métiers explicites et implicites contenues dans les séquences de composants électriques.

Dans le second chapitre, nous présenterons une classification des réseaux de neurones selon leur prise en compte du paramètre temporel. Nous détaillerons

surtout deux réseaux de neurones récurrents à couches : le modèle de Elman, appelé aussi Simple Recurrent Network, qui a une représentation interne et implicite du temps et les LSTM, qui ont une représentation interne et explicite du temps. Chacun de ces modèles a marqué une rupture avec les modèles précédents en terme d'architecture ou de performance. Après avoir décrit ces deux exemples nous aborderons les procédés listés dans la littérature permettant d'extraire les règles encodées dans un réseau de neurones récurrent, après un apprentissage de séquences.

Dans le troisième chapitre, nous décrirons l'ensemble de nos travaux expérimentaux menés : nous détaillerons nos implémentations des modèles de Elman et des LSTM, ainsi que leurs performances selon différentes séquences issues de différentes grammaires artificielles. Nous détaillerons alors le nouveau processus d'extraction des connaissances encodées dans les séquences à partir d'un réseau de LSTM. Nous montrerons notamment un exemple innovant mené sur des séquences électriques issues d'une grammaire électrique adaptée d'un cas concret pour établir la transposabilité de nos travaux à des problématiques industrielles.

Enfin nous discuterons de l'ensemble de nos travaux, en détaillant notamment les différents objectifs atteints, et les perspectives qu'ils ouvrent.

Chapitre 1

Extraction et préparation des données issues de schémas électriques

Table des matières

1.1 Introduction

Dans ce chapitre, nous abordons le contexte industriel plus en profondeur et nous décrivons les données fournies par l'entreprise ainsi que leurs caractéris-

tiques. Nous présentons notre première approche qui nous permet de transformer et représenter ces données brutes en concepts métier. Nous verrons qu'il est possible de représenter chaque schéma électrique en un graphe unique. Nous décrirons ensuite nos travaux sur l'extraction de l'empreinte implicite d'une entreprise-cliente. Nous aborderons pour cela les techniques de clustering tel que le clustering hiérarchique et les différentes représentations possibles des résultats. Nous mettrons ainsi en évidence la structure hiérarchique et séquentielle des composants électriques, et leur importance. Nous présenterons notamment le prototype réalisé et qui découle des travaux menés dans ce chapitre.

1.2 Contexte environnemental et informatique

La première étape de nos travaux a consisté en la constitution du corpus de données. Nous avons observé et échangé avec les experts de l'entreprise Algo'Tech et les entreprises-prestataire. Ces dernières utilisent, pour paramétrer la base de données de leurs logiciels, des informations issues des schémas électriques. Ce travail d'extraction réalisé via un expert-Algo'Tech n'était, à notre arrivée, pas formalisé : le "pourquoi" et le "comment" étaient donc essentiellement présents dans l'esprit des opérateurs humains (ceux qui tracent les schémas) et les experts-Algo'Tech (ceux qui les analysent), et le "quoi" résidait dans les schémas électriques (au format PDF la plupart du temps). Nous avons donc dû réaliser tout un processus d'observation, d'entretien et d'analyse des procédés de travail existants afin de savoir quelles étaient les données sur lesquelles nous devions nous focaliser, et ainsi constituer nos corpus de données. Cette étape était nécessaire, car sans elle, nous n'aurions pas pu comprendre la complexité du travail d'extraction des connaissances réalisé par les experts de l'entreprise Algo'Tech. Néanmoins nous étions sûr de l'existence d'une structure implicite dans les schémas électriques, puisque les experts Algo'Tech arrivaient à formaliser les connaissances extraites dans une base de données existante structurée. Dans cette section nous allons donc détailler l'ensemble du travail mené pour la mise en évidence de cette structure implicite directement à partir des schémas électriques.

Pour cela, nous avons entamé nos travaux en nous intéressant au processus de gestion de projets informatiques existant au sein de l'entreprise Algo'Tech : Nous avons pu participer, avec les experts, à des réunions d'analyse des besoins et d'extractions des connaissances réalisées entre Algo'Tech et leurs clients et cela sur différents projets. Dans un premier temps, nous décrirons les schémas électriques et leur structure. Nous présenterons ensuite l'ensemble du travail préliminaire visant à extraire les concepts métiers globaux d'un client donné afin de vérifier l'existence de motifs communs dans la structure de différents

schémas d'un même client. Enfin nous aborderons le processus de découverte des connaissances dans les données, avant de décrire d'un point de vue informatique le prototype réalisé.

1.3 Les données : les schémas électriques

Si le domaine électrique est très normé en France, la représentation des schémas reste libre et à l'appréciation de l'opérateur humain qui trace le schéma. Nous avons donc étudié ces schémas du point de vue des opérateurs humains, afin de comprendre les codes et les usages qui dictent leur travail. x Dans le jargon électrique, une page d'un PDF représentant un schéma est nommée un folio. Nous allons détailler la structure et les informations apportées par ces documents dans la section suivante.

1.3.1 Structure d'un Folio

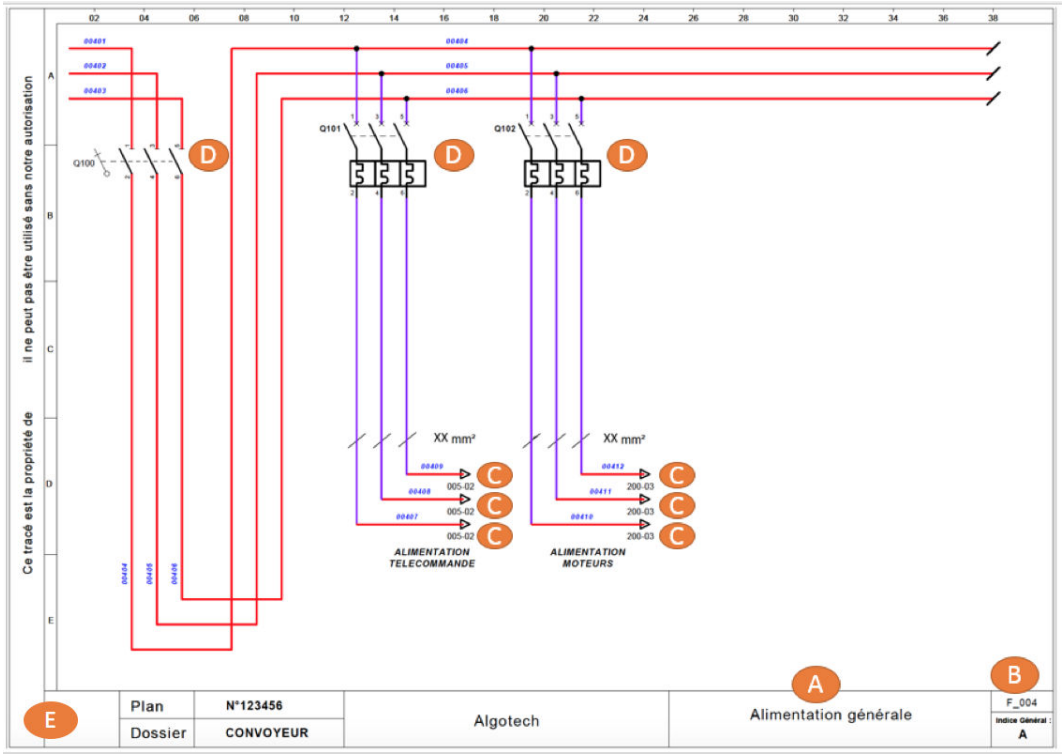


FIGURE 1.1 – Organisation et contenu du folio F_004 : A - Le titre du folio, B - Le numéro de folio, C - Les renvois folios, D - Les composants électriques, E - Le cartouche

Quelle que soit l'entreprise-client émettrice, les folios de différents schémas électriques partagent les mêmes caractéristiques sur le fond et la forme. D'un

point de vue structurel, un folio se compose de deux parties : le cartouche (Figure 1.1, label E) et le circuit dessiné.

Le cartouche désigne le cadre extérieur, au bord de la feuille A4. Il permet de délimiter la zone de dessin, de fournir une échelle de mesure (en haut dans Figure 1.1) et d'apporter un complément d'information textuelle sur la portion de circuit représenté. C'est au niveau du cartouche que l'on retrouve par exemple, le logo de l'entreprise, le nom et numéro du projet, ainsi que le titre du folio et le numéro de folio. Le titre du folio permet d'apporter une information sémantique sur ce que représente la portion du circuit. Le numéro de folio se compose d'une partie alphabétique et une partie numérique. La partie alphabétique dépend des usages en entreprise (exemple : Folio 004, F_004). La partie numérique encodée sur 3 chiffres, est plus standard et s'incrémente tout au long des pages d'un schéma électrique, mais pas toujours de manière continue (par exemple, la page 5 peut être numérotée F_004 et la page 6, F_200, selon les conventions en vigueur dans l'entreprise-cliente).

Le circuit représente schématiquement une portion d'une installation électrique réelle : les fils (câbles) électriques sont généralement désignés par des lignes droites sur le folio (en rouge et violet sur la figure 1.1) et les composants électriques sont représentés par des symboles et des noms appelés repères. Dans la figure 1.1, les repères des trois composants électriques sont de gauche à droite : Q100, Q101 et Q102. La structure des repères dépend beaucoup des conventions de nommage en vigueur dans l'entreprise cliente. Ils se composent, toutefois, souvent d'une partie alphabétique (désignant l'appartenance d'un symbole électrique à une catégorie) et d'une partie variable (chiffres et symboles). L'alliance des deux permet de créer un repère unique pour le symbole électrique sur toute l'installation. Parmi les autres indications textuelles, on retrouve aussi les numéros ou identifiants de chacun des fils (en bleu sur la figure 1.1), ainsi que des informations sur leur diamètre (XX mm^2). Les renvois folios complètent le dessin du circuit électrique en donnant une information sur la connectivité des différentes portions et donc des folios. Si dans le fichier PDF, deux portions peuvent être dessinées sur deux folios différents, dans la réalité il n'y a pas d'interruption physique dans le fils électrique lui-même. Un folio peut aussi être décrit comme deux ensembles d'informations complémentaires mais distincts, d'une part se trouve le tracé du circuit (symboles et fils électriques), et d'autre part se trouve l'ensemble des indications textuelles (nom des équipements et les indications sur les fils électriques). Ces ensembles peuvent être présentés, chacun indépendamment de l'autre, à un expert client du domaine qui reconnaitra alors la partie du circuit représenté, car il existe dans le tracé et dans le texte de chaque folio une partie de la connaissance globale du domaine du client.

1.3.2 Lecture d'un schéma électrique

La lecture d'un schéma électrique se fait folio par folio en suivant des renvois folios. Les usages les plus communs en schématique font que le premier folio représente l'alimentation de l'installation électrique et les derniers représentent les équipements électriques. En suivant les fils et les renvois folios, il est possible de lire le schéma et de connaitre quel câble alimente quel équipement en passant par quel composant. En d'autres termes, c'est en suivant les fils électriques que l'on arrive à extraire une séquence de composants électriques.

Comment lit-on un renvoi folio ? La lecture de la figure 1.1 informe de la présence de six renvois folios : trois indiquent 005-02 et trois autres indiquent 200-03. Les trois premiers renvois folios indiquent que les fils électriques continuent dans le folio 005 (F_005) et que leur représentation commencera, sur ce folio, colonne 02 selon l'échelle de mesure représentée (règle située en haut de la figure 1.1). De même, les trois derniers renvois folios indiquent une continuation des fils dans le folio 200, colonne 03.

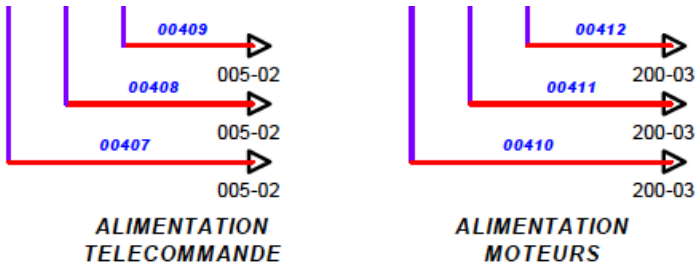


FIGURE 1.2 – Détail des renvois folios du folio 004

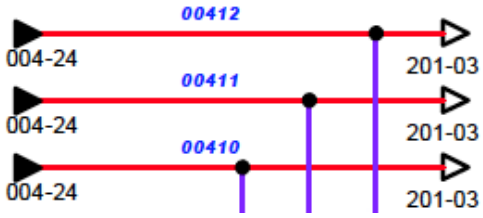


FIGURE 1.3 – Agrandissement des renvois folios du folio 200

La différence entre trois renvois folios similaires réside dans le numéro du câble associé à chaque renvoi. Ce numéro représente un identifiant unique dans un schéma qui permet d'établir le lien entre chacun des trois renvois folios du folio 004 et ceux des folio 200 et 005, comme on peut le constater sur les figures 1.2 et 1.3

1.3.3 Les données brutes : les PDF et DXF

Les schémas électriques peuvent se présenter, entre autres, sous deux formats : les fichiers PDF décrits précédemment et/ou des fichiers DXF. Le sigle PDF, signifiant Portable Document Format, est un format utilisé pour présenter et échanger des documents de manière fiable et cela indépendamment du logiciel, du système d'exploitation ou du matériel utilisé pour le générer. L'ensemble des schémas électriques que nous avons reçus se sont trouvés sous ce format là. Nous avons donc en premier cherché à exploiter ce type de document. Lorsque le format le permettait, il s'avère qu'il nous a été facile d'extraire l'entièreté du texte présent dans les folios des documents PDF. Après analyse, et pour chaque client, nous avons choisi d'extraire les informations suivantes de chacun des folios des schémas électriques :

— les titres des folios,
— les numéros des folios,
— les renvois folios
— les repères.

Le signe DXF, signifiant Drawing eXchange Format, est quant à lui un format d'échange ouvert standard de plans. Créé par la société Autodesk, il permet l'interopérabilité entre différents outils de conception assistée par ordinateur. C'est un format considéré ouvert car il s'agit en fait d'un fichier ASCII qui peut être lu par n'importe quel éditeur de texte et qui est organisé par sections (HEADER, BLOCKS, ENTITIES, etc.). Un fichier DXF est donc une traduction ASCII d'un schéma électrique : en analysant les blocs et leurs attributs, il est possible d'extraire des données et méta données (coordonnées, attributs, nom du bloc, etc.) relatives aux composants électriques.

Ainsi si le texte extrait d'un folio de PDF permet d'avoir le nom de composant électrique par exemple, l'analyse du DXF permet de fournir les coordonnées (x, y) au sein du folio. C'est cette information qui permet l'analyse du contenu que nous décrirons plus loin dans la section 1.5. La lecture et l'interprétation d'un schéma électrique nécessite de prendre en compte le texte et les symboles électriques utilisés. Comme notre souhait est d'explorer la valeur sémantique du texte et son importance dans la représentation des connaissances des dessinateurs, nous avons choisi d'opter pour une alliance de l'analyse des PDF et des DXF. Cette dernière permet ainsi une analyse quasi-complète des schémas électriques et une interprétation plus exhaustive sans avoir à utiliser des techniques de reconnaissance d'image, qui est un sujet complexe en soi.

1.4 Première approche : extraction des concepts

Dans cette section nous présentons les travaux préliminaires réalisés afin de mettre en évidence l'organisation sémantique des schémas, représentation abstraite du métier du client présente dans l'esprit des experts du domaine. A travers la fouille et l'analyse des textes issues des schémas, nous mettons en évidence les concepts métiers, ainsi que les relations entre eux. Ces travaux ont été menés sur des schémas électriques d'un même client ayant été réalisés sur une période d'une dizaine d'années.

1.4.1 Transformation d'un schéma en un graphe

En nous basant sur les informations structurelles d'un folio (les renvois folios) nous avons transformé chaque schéma électrique en graphe. Lors de la lecture d'un schéma électrique, un expert suit les fils électriques et les renvois folios afin de naviguer d'un folio à un autre. Nous avons mimé algorithmiquement ce raisonnement : Pour chaque folio de chaque schéma, le numéro de folio est désigné comme identifiant d'un nœud et le titre du folio comme étiquette du même nœud. Ensuite les renvois folios sont utilisés afin de créer des liens entre les différents nœuds. On obtient ainsi un graphe représentant, à un bas niveau d'abstraction, le schéma électrique d'origine. Nous appellerons ce graphe, "graphe original" dans la suite. La figure 1.4 présente le graphe original d'un schéma client que nous qualifierons de schéma A. Cette nouvelle représentation des schémas électriques a permis d'une part, de transformer un PDF linéaire de plusieurs pages en une seule représentation tenant sur une seule feuille, et d'autre part, elle a permis l'identification de motifs au sein de plusieurs schémas d'un même client. En comparant les graphes d'un même client, on retrouve en effet souvent une organisation similaire des nœuds et des arcs dans les graphes issus d'une même entreprise-cliente, mais aussi un même vocabulaire.

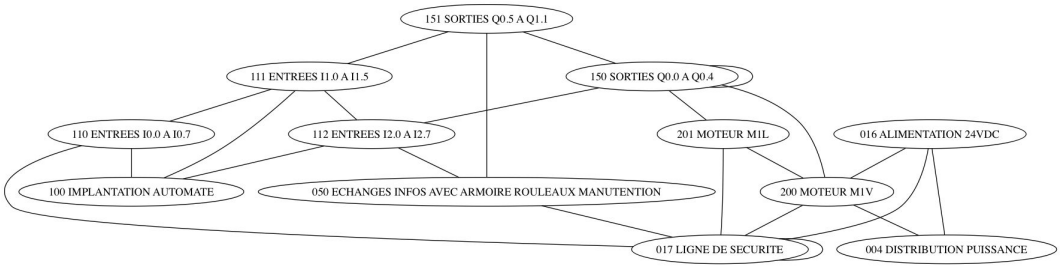


FIGURE 1.4 – Graphe original du schéma électrique A de 43 pages : seuls les folios comportant des renvois folios et donc des portions de circuits électriques sont représentées.

1.4.2 Abstraction des graphes par étude de similarité des mots

Par habitude, les experts des entreprises-clients dessinent souvent de la même façon leurs schémas. Ces habitudes se retrouvent aussi dans le vocabulaire choisi. Les experts Algo'Tech sont ainsi capables de savoir qui de leur client a pour habitude de nommer leur premier folio alimentation "Alimentation Générale" ou "Distribution puissance". Pour mettre en évidence ces habitudes de vocabulaire et s'en affranchir, nous avons procédé à l'analyse des titres des folios. Pour chaque schéma, après avoir extrait l'ensemble des titres, nous avons créé un sac de mots sur lesquels nous avons appliqué les filtres suivants, en utilisant des expressions régulières : les noms des équipements tel que MV1, les mots de liaison de la langue française (de, des, l', a, etc.), ainsi que les chaines numériques ayant été éliminés, seules les chaines de caractères alphabétiques de plus de 4 symboles ont été conservées. Une fois les sacs de mots filtrés obtenus (un sac de mot par schéma électrique) nous avons calculé l'intersection entre eux afin d'obtenir les mots communs. Le résultat obtenu présenté à un expert métier Algo'Tech et comparé aux titres issus des schémas client, nous a permis de déduire que ces mots communs constituent les piliers de la logique métier du client, et donc les concepts globaux de son métier. En d'autres termes, ce sont les termes qui sont habituellement les plus utilisés chez l'entreprise cliente. L'étape suivante a été d'utiliser ces mots communs sur les graphes générés précédemment afin d'étudier les relations sémantiques entre les concepts, mais tels qu'ils sont présents dans chacun des différents schémas électriques. Nous avons donc mis en place un dictionnaire python (liste dont les index sont des clés, à savoir des valeurs autres que numériques) dans un fichier JSON contenant pour chaque concept global, les déclinaisons et synonymes possibles, comme présenté ci-dessous :

'ALIMENTATION' : ['ALIMENTATION', 'ALIMENTATIONS', 'DISTRIBUTION', 'PUISSANCE'] 'BORNE' : ['BORNE', 'BORNES']

Dans l'exemple ci-dessus, les premiers mots sont les clés (ce sont les concepts métiers récurrents d'un client), alors que ceux entre " [ ] " sont les valeurs proches du mot commun. Une intervention des experts Algo'Tech a été nécessaire à ce niveau-là pour notamment déterminer les synonymes et valider les termes techniques. L'annexe B présente un exemple de dictionnaire de concepts métiers complet. Nous avons ensuite parcouru chacun des graphes d'origine et en appliquant le dictionnaire, remplacé chaque titre de folio, par le mot commun qui lui correspond. Grâce à cette approche, nous avons créé un graphe modifié, copie du graphe original, pour chaque schéma électrique, où la différence réside dans les étiquettes des nœuds. La figure 1.5 présente le graphe modifié du schéma A obtenu avec ce procédé. Nous avons ainsi obtenu un en-

semble de graphes avec des étiquettes (labels des nœuds) standardisées.

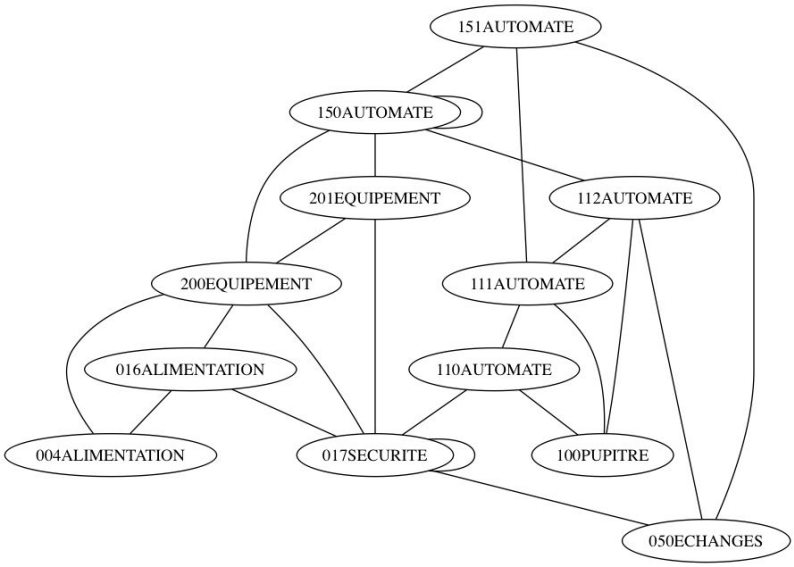


FIGURE 1.5 – Graphe modifié du schéma électrique A de 43 pages : Les étiquettes des nœuds ont été remplacées grâce au dictionnaire des mots communs.

1.4.3 Fusion de nœuds et concaténation de nœuds puis de graphes

Uniformiser les étiquettes a pour but de fournir des graphes avec une base commune : leurs étiquettes. Le but recherché lors des développements est de pouvoir obtenir, à partir de chaque graphe modifié, un graphe plus abstrait représentant la logique métier globale du schéma électrique de départ.

Pour cela, en premier lieu, à la lecture des graphes modifiés obtenus, les nœuds portant la même étiquette ont été fusionnés tout en conservant les liaisons des nœuds originaux. Nous obtenons ainsi un graphe abstrait représentant un réseau sémantique valide du point de vue du métier du client. A la fin de ce processus, nous avons obtenu autant de réseaux sémantiques que de schémas électriques. En reprenant l'hypothèse de départ, à savoir qu'un schéma est une partie de la connaissance globale du client dans son domaine, chaque réseau sémantique est alors une instanciation d'un réseau sémantique global. La figure 1.6 présentant le graphe abstrait du schéma A, est donc une instanciation de la connaissance globale du client sous la forme d'un réseau sémantique.

En second lieu, nous avons fusionné tous les réseaux sémantiques : autrement dit, nous avons regroupé toutes les instances afin de former le réseau

sémantique global du client, soit un graphe global de concepts métier. La figure 1.7 présente un exemple de la fusion de 3 graphes abstraits issus de 3 schéma électriques et leur fusion en un seul réseau sémantique représentant les concepts métiers présents dans ces 3 schémas et les relations.

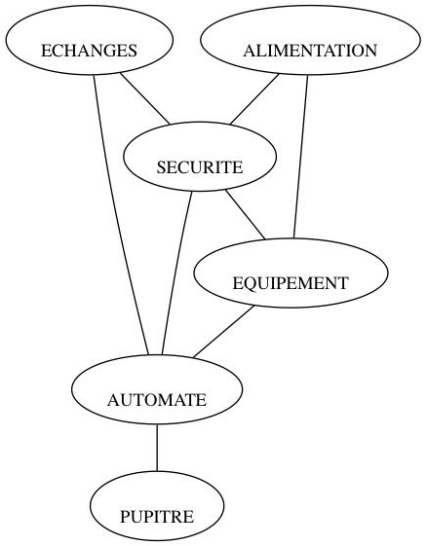


FIGURE 1.6 – Représentation du schéma électrique A de 43 pages sous forme de graphe de concepts métiers : Les nœuds portant des étiquettes similaires ont été fusionnés.

1.4.4 Analyse et résultat

Cette première approche bien que simple, a permis de confirmer l'existence d'une structure cachée dans les schémas électriques d'un même client. Le fait que l'on ait mis en évidence des répétitions dans le vocabulaire et que l'on arrive à extraire un graphe global de concept métier à partir des schémas, confirme l'existence de règles implicites qui dictent la manière de tracer ces schémas. Par ailleurs, cette approche a aussi permis de regrouper plusieurs folios de différents schémas électriques sous la bannière d'un concept métier. En effet, en réalisant le chemin inverse, à savoir du graphe global de concepts métier au graphe original, il est possible de constituer des groupes de folios par concepts tels que présenté dans la figure 1.8. Ce regroupement est exploité dans la section suivante. Pour la suite de notre approche, nous avons choisi d'explorer le contenu de chaque folio : le circuit électrique.

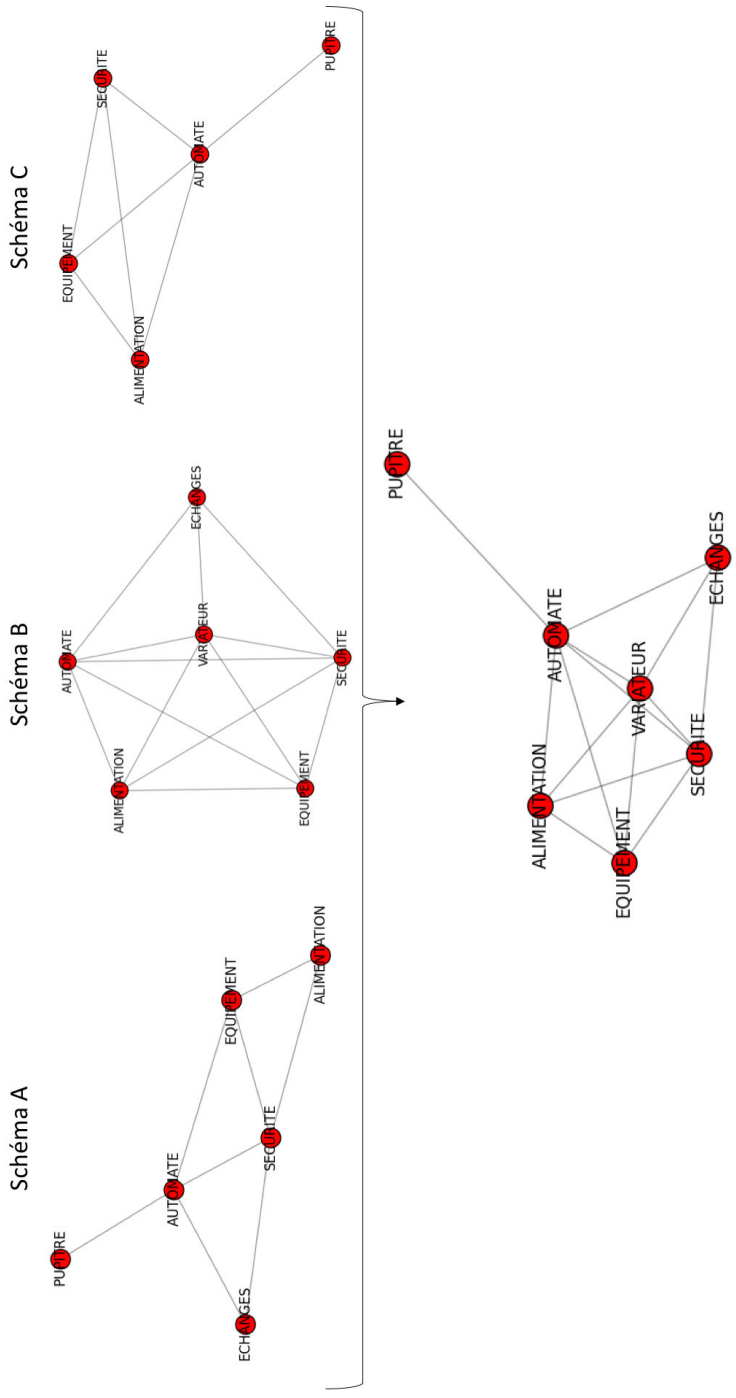


FIGURE 1.7 – Exemple de génération d'un graphe de concept métier d'un client à partir de 3 graphes de logique métier issus de l'analyse de 3 schémas électriques (le schéma A est le même que celui des figures 1.4, 1.5 et 1.6)

1.5 Seconde approche : extraction de l'empreinte implicite par data mining

Les experts d'une entreprise-cliente sont capables de reconnaitre, en regardant un circuit électrique (folio sans le cartouche), si celui-ci porte leur empreinte ou non et donc s'il provient de leur entreprise. Cette information implique qu'il existe, dans le tracé d'un schéma et donc dans l'agencement des composants électriques sur un folio, une empreinte implicite et donc une connaissance implicite de l'entreprise-cliente dans ce tracé. Le processus de découverte de connaissances à partir des données, ou Knowledge discovery of data (KDD) en anglais, a été décrit par Fayyad *et al.* [1996]. Itératif et interactif, ce processus implique de nombreuses étapes nécessitant une prise de décision de la part d'un opérateur humain. Dans la section suivante, nous utilisons ce processus pour la mise en évidence de l'empreinte implicite et par extension, pour découvrir quelles données sont porteuses de cette connaissance.

1.5.1 Une approche par le contenu

La donnée centrale considérée ici est le composant électrique (Figure 1.1, label D). Les fichiers DXF nous fournissant des informations sur les composants électriques sous la forme d'un symbole dessiné dans un folio, permettent aussi d'extraire ses coordonnées (x, y). Ainsi, pour une entreprise-cliente donnée, pour chaque folio de chaque schéma électrique, nous utilisons du text mining sur les PDF, pour extraire les repères des différents composants électriques. Puis grâce à cette liste de repères, nous extrayons depuis les DXF relatifs aux PDF, les coordonnées de chacun des composants électriques. Cette opération réalisée sur chaque folio, permet ainsi l'obtention d'autant de groupes de composants électriques que de folios, tout schémas électriques confondus. Puisque c'est l'arrangement interne des composants dans un folio qui nous intéresse, comme première étape, nous avons choisi d'ordonner les composants à l'intérieur de chaque groupe par ordre croissant selon leurs coordonnées (x, y) respectives (tri par x croissant, puis y croissant). En appliquant le même procédé à tous les ensembles de composants électriques par folio, il sera ainsi plus aisé de les comparer et d'étudier leurs similarités. Nous utiliserons à partir d'ici le terme de séquence de composants électriques (la séquence étant une notion très importante pour la suite de nos travaux) pour désigner un ensemble de composants électriques ordonnés selon leurs coordonnées (x, y) croissantes respectives. Ainsi nous obtenons autant de séquences que de folios pour une entreprise-cliente donnée, ces séquences représentant l'agencement des composants électriques au sein de chaque folio. Par exemple, dans la figure 1.1, la séquence est Q100, Q101, Q102.

Nous avons analysé ces séquences de composants électriques issus de différents schémas afin de mettre en évidence les motifs existants dans l'agencement

des composants électriques. Pour cela nous avons entamé nos travaux par l'analyse des repères. La comparaison des séquences issues de schémas distincts mais de folios ayant la même thématique (par exemple les folios Alimentations de 2 schémas différents) a mis en évidence l'existence de points communs dans les repères des composants. Ceux nommés avec le même préfixe (Q100, M2, M1T, G2, V1T) représentent des composants ayant les mêmes fonctions et sont souvent représentés par le même symbole. Nous avons ainsi identifié plusieurs groupes de repères (ceux commençant par M, ceux commençant par Q, etc) que nous avons soumis aux experts.

Il existe dans les repères des règles de nommage spécifiques à chaque entreprise-cliente qu'il est aussi possible de mettre en évidence. Nous exploitons l'existence de cette généricité afin d'étudier l'existence de corrélation dans le contenu de différents folios. Nous avons ainsi abordé avec les experts-clients mais aussi les experts-Algo'Tech la notion de famille de composants électriques et leur signification. Ensemble, nous avons répertorié 25 familles (catégories) de composants électriques. Ces familles sont assez courantes dans le domaine global des schémas électriques : comme elles ont été définies une fois, elles peuvent être utilisées pour tous les schémas dans le domaine électrique. Le tableau 1.1 présente un exemple de repères de composants électriques et de leurs familles. La liste des familles de composants est fournie en annexe C.

Désignation de la famille	Préfixe associé	Exemple de repères
Générateurs dispositifs alimentation	G	G2
Moteurs	M	M1T, M3
Appareils mécanique de connexion pour circuits de puissance	Q	Q100, Q1T

TABLE 1.1 – Exemples des différentes familles de composants électriques

Nous utilisons ainsi dans les étapes suivantes ces deux éléments - les séquences de composants électriques et la liste des catégories de composants électriques - pour la construction de nos ensembles de données et le processus d'exploration de ces données décrit ci-dessous.

1.5.2 Processus de Data mining : le clustering hiérarchique

Dans sa description, Fayyad *et al.* [1996] distingue le processus de découverte des connaissances du data mining : il présente en effet le second comme une étape du premier et non l'équivalent. Il définit le data mining comme l'application d'un algorithme particulier afin d'extraire des patterns à partir des

données. Il est en effet rejoint par Ramos *et al.* [2007] qui décrit le KDD comme un ensemble de 3 étapes :

— Pre-processing : la préparation des données
— Data mining : la fouille des données
— Post processing : l'analyse des résultats

Ainsi, nous allons aborder, dans ce qui suit, nos travaux selon ces 3 étapes.

Preprocessing

Deux traitements sont réalisés sur les séquences : la création de 2 corpus de données et leur transformation. Le premier corpus de données regroupe toutes les séquences, indépendamment des schémas électriques d'origines. Le second corpus de données se compose quant à lui de sous-corpus de données, chacun regroupant les séquences selon le concept dominant auxquels elles appartiennent. En effet, ayant déterminé précédemment des groupes de folios selon leur concept métier d'appartenance, (voir section 1.4.4), nous avons repris ces groupes et traité leurs folios de sorte à obtenir des groupes de séquences selon le concept d'appartenance.

Le second traitement réalisé est celui de la transformation. Ayant défini 25 familles de composants électriques, nous avons traduit chaque séquence en un vecteur binaire de taille 25, chaque unité désignant la présence ou l'absence d'une famille de composants électriques dans le folio considéré. Nous obtenons à la fin de cette partie deux corpus de vecteurs binaires.

Data mining : Clustering hiérarchique

Afin d'observer d'une part les clusters résultants et afin d'autre part, de mettre en évidence l'empreinte présente dans les séquences, nous nous sommes inspirés pour cette partie, des travaux de Servan-Schreiber *et al.* [1988] et de Servan-Schreiber *et al.* [1991]. Tous deux utilisent le clustering hiérarchique afin de mettre en évidence l'encodage d'une grammaire (ensemble de règles et de patterns) dans un réseau de neurones récurrent qui aurait appris plusieurs séquences de symboles issues de cette même grammaire.

Le clustering hiérarchique est une méthode d'analyse de clusters qui vise à créer une hiérarchie de clusters. Ce dernier terme désigne un ensemble de données qui possèdent une ou plusieurs caractéristiques communes (exemple : tous les individus de plus de 1,70 m, ou tous les documents qui sont des articles de 2016). Le clustering hiérarchique possède deux fonctions importantes : une distance (distance euclidienne, distance de hamming [Hamming, 1950] ou distance de Jaccard [Jaccard, 1901], par exemple) et une méthode de liaison (linkage method en anglais) qui calcule la distance entre deux clusters selon

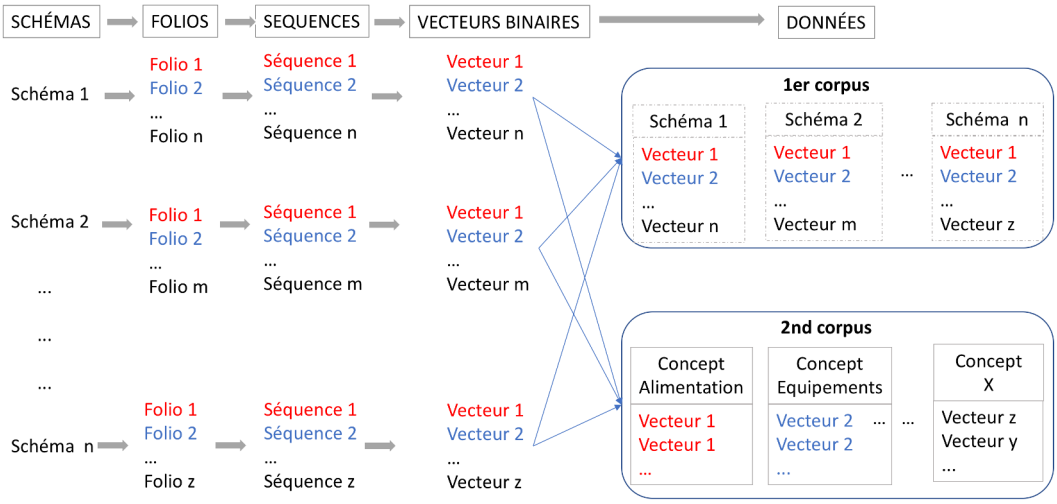


FIGURE 1.8 – Présentation des étapes de constitution des corpus. Les éléments qui partagent une même couleur appartiennent au même concept métier. Dans le 1er corpus, les pointillés indiquent les schémas d'origines dont sont issus les vecteurs.

la méthode (distance) choisie. Initialement, chacune des données constitue un cluster (que nous appellerons cluster-simple pour la suite). Ensuite, à chaque itération, l'algorithme fusionne les clusters les plus proches en un nouveau cluster, c'est à dire ceux qui partagent le plus de caractéristiques communes, selon les deux paramètres cités ci-dessus, jusqu'à ce qu'il ne reste plus qu'un seul cluster.

Pour choisir la meilleure combinaison "distance-méthode de liaison", il est conseillé de calculer le coefficient de corrélation de Cophenetic [Sokal et Rohlf, 1962] : plus sa valeur est proche de 1, plus le clustering préserve les distances originales. Le résultat d'un clustering hiérarchique peut être graphiquement représenté sous la forme d'un dendrogramme (la représentation standard) ou d'un arbre de cluster (un graphe) avec les séquences (les cluster-simples) en tant que feuille et les clusters obtenus comme nœud. Hees [2015] présente de manière concise et explicite le clustering hiérarchique et les dendrogrammes. Dans notre cas, la combinaison ayant donné le meilleur coefficient de corrélation de Cophenetic est la distance de Jaccard [Jaccard, 1901] alliée à la méthode de liaison moyenne (average linkage method). Nous avons ainsi appliqué ces paramètres sur l'ensemble des données de nos deux corpus et nous avons à chaque fois représenté cela sous la forme de dendrogrammes et d'arbres de clusters. La figure 1.9 représente un dendrogramme résultant du clustering hiérarchique avec les paramètres cités ci-dessus, et les figures 1.11 et 1.12 représentent des arbres de cluster des concepts Alimentation et Equipement. Ces résultats sont issus de l'analyse de 3 schémas électriques d'une même entreprise-cliente.

Post-processing

Pour mieux comparer les clusters obtenus entre le premier corpus de vecteurs (toutes les séquences de tous les schémas électriques confondus) et les clusters des sous-corpus de vecteurs, nous avons aussi choisi de décrire les clusters obtenus par le clustering hiérarchique par des vecteurs binaires. Chaque vecteur binaire de chaque cluster-nœud, résultant de l'union des vecteurs binaires des clusters-fils. La figure 1.10 illustre cette opération. Grâce à cette normalisation, nous avons comparé les arbres de clusters des différents corpus entre eux : nous avons comparé les arbres de clusters de chaque concept (sous-corpus du 2nd corpus de la figure 1.8) à l'arbre de clusters du premier corpus (de la figure 1.8). Ainsi nous avons observé ce qui suit sur les arbres de clusters : à l'exception de quelques folios, nous avons retrouvé dans l'arbre de cluster du premier corpus de vecteurs, l'ensemble des arbres de cluster des sous-corpus de vecteurs du second corpus. Autrement dit, nous avons une concordance entre les clusters des concepts et les clusters tous folios confondus. De plus, nous avons observé que des folios dessinés à différentes périodes (années différentes), et ne portant pas le même titre se retrouvent groupés avec souvent une distance de 0 : cela implique que la séquence de composants électriques est restée inchangé pendant la période séparant le dessin des deux schémas d'origines. En d'autres termes, les habitudes de dessins sont restées les mêmes au cours du temps. Enfin, l'utilisation d'arbre de clusters, a aussi permis la mise en évidence de cas particulier (ou d'erreur) dans un schéma électrique, lorsque par exemple, un cluster-simple de folio est dessiné à un endroit inattendu. L'analyse des résultats, lors de cette étape a donc permis la mise en évidence d'une récurrence - et donc d'habitudes de travail - dans l'agencement des symboles électriques. C'est bien dans les séquences de composants électriques des folios des schémas électriques, que l'on trouve encodée la connaissance liée aux habitudes de travail de l'entreprise cliente.

1.6 Analyse des résultats précédents

La seconde approche développé dans cette section a permis de confirmer l'existence d'une empreinte cachée dans les séquences de composants électriques. L'application d'un algorithme de clustering hiérarchique sur des séquences de composants électriques issues de folios différents a mis en évidence l'existence de motifs dans ces séquences. Cela est vrai pour des folios issus de schémas d'une même année, mais aussi pour des folios de schémas avec plusieurs années de différence. Il existe ainsi des habitudes de représentation qui sont pérennes au sein des entreprise-clientes.

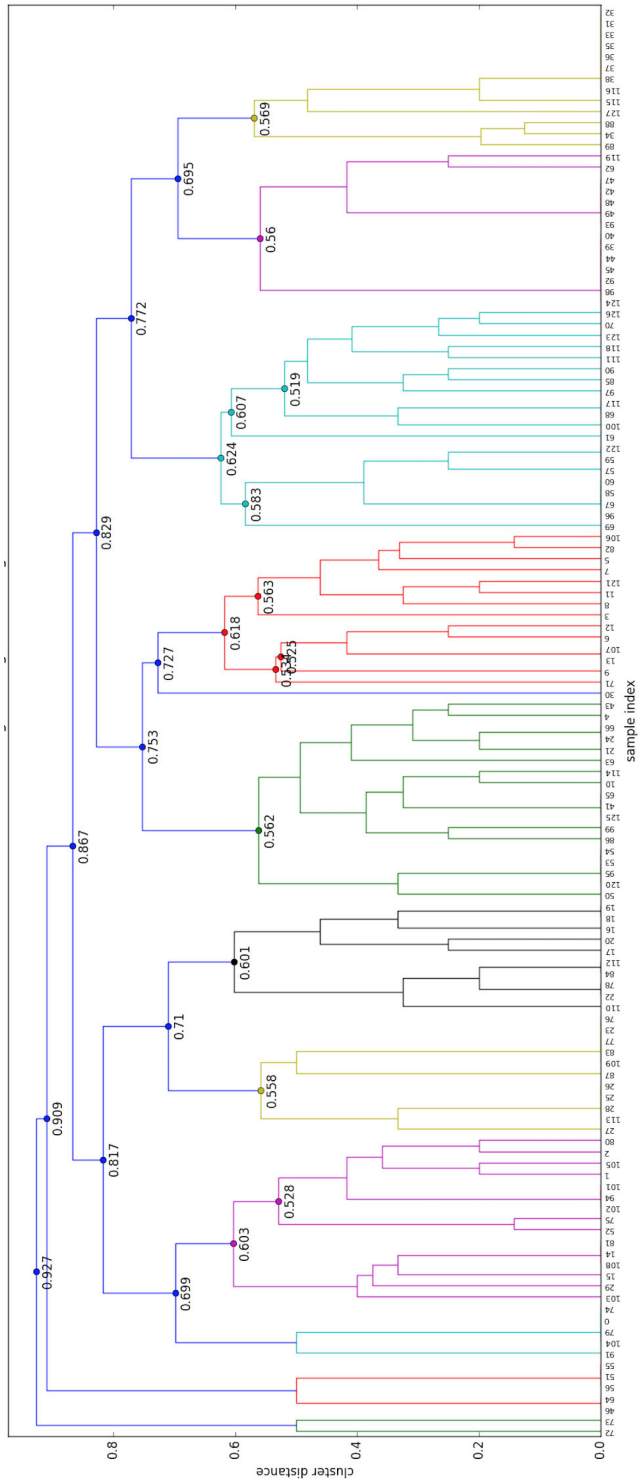


FIGURE 1.9 – Un dendrogramme, résultat d'un clustering hiérarchique appliqué sur les séquences issues de 3 schémas électriques d'un même client. Les chiffres présents sur l'axe x identifient les séquences et les chiffres sur le dendrogramme désignent les distances calculées entre les clusters.

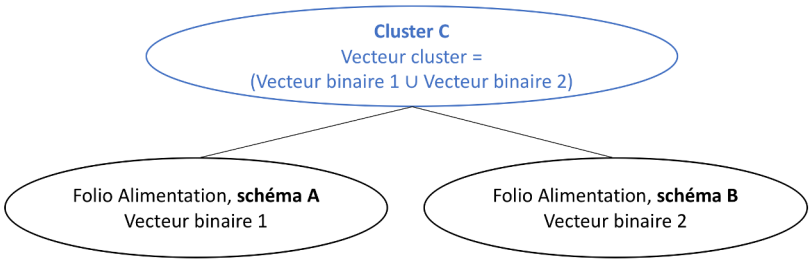


FIGURE 1.10 – Portion d'un arbre de cluster résultant du clustering hiérarchique : les vecteurs binaires des clusters-nœuds, résultent de l'union des vecteurs binaires des clusters-fils.

Il est important de souligner la différence entre les séquences utilisées dans cette section et celles qui seront utilisées dans la suite des travaux et d'expliquer les raisons de ce choix. Dans le cas présent, les séquences sont générées grâce aux coordonnées (x,y) de chaque composant électrique, alors que dans le chapitre 3, les séquences seront générées par la lecture d'un graphe orienté (ce qui sera détaillé par la suite). Le processus d'extraction des séquences de composants électriques à partir de la lecture des fils électriques n'ayant pas été opérationnel dans les temps impartis de la thèse, nous ne pouvions pas extraire les séquences directement depuis les schémas électrique. Nous avons donc pris le parti d'exploiter des séquences ordonnées selon les coordonnées (x,y) des composants électriques pour la mise en évidence de l'empreinte implicite dans la section ci-dessus. Notre but étant ici de souligner et mettre en évidence l'existence de règles et connaissances implicites au niveau des composants électriques.

1.7 Présentation du prototype

L'ensemble des travaux décrits dans cette section ont été réalisé avec le langage Python. Afin de permettre un transfert facilité des travaux réalisés dans le cadre de la thèse, l'ensemble du code réalisé a été regroupé dans un prototype afin de permettre la réalisation des tests supplémentaires sur différents clients et prospects par les experts de l'entreprise Algo'Tech sans mon intervention.

Voici quelques précisions techniques :

— le prototype a été réalisé pour le système d'exploitation Windows
— une installation a été fournie comportant l'exécutable, un manuel d'utilisation ainsi qu'un exemple de test avec les résultats associés.
— le prototype a pour fonctionnalités principales : la fouille de documents PDF et DXF, l'export des résultats sous Excel, et la mise en place d'algorithmes de clustering hiérarchique avec pour but de fournir les dendrogrammes et arbres de clusters associés.

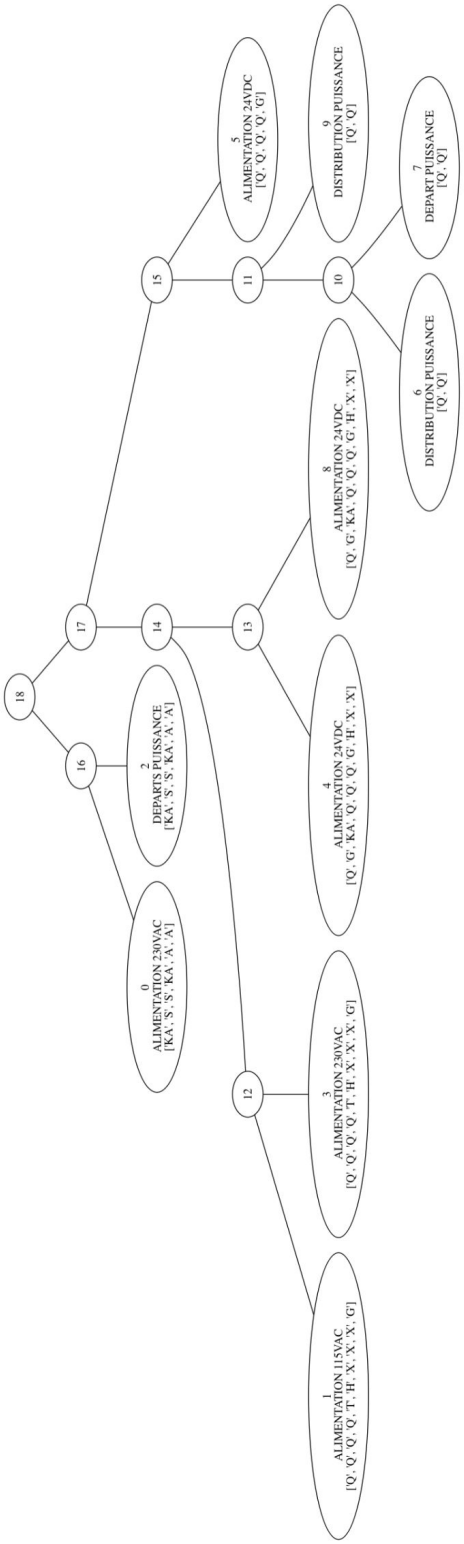


FIGURE 1.11 – Arbre de cluster résultant du clustering hiérarchique sur toutes les séquences rattachées au concept métier ALIMENTATION pour 3 schémas électriques d'un même client (2 réalisés en 2010 et un en 2014)

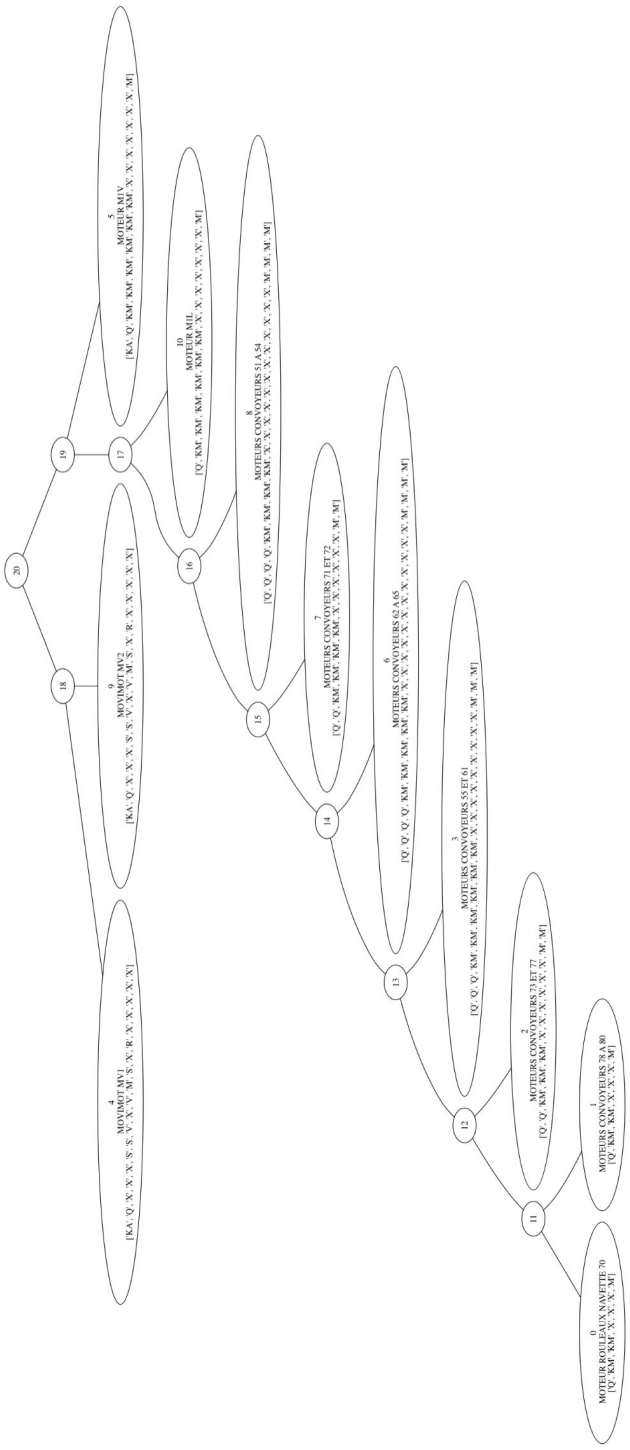


FIGURE 1.12 – Arbre de cluster résultant du clustering hiérarchique sur toutes les séquences rattachées au concept métier EQUIPEMENT pour 3 schémas électriques d'un même client (2 réalisés en 2010 et un en 2014)

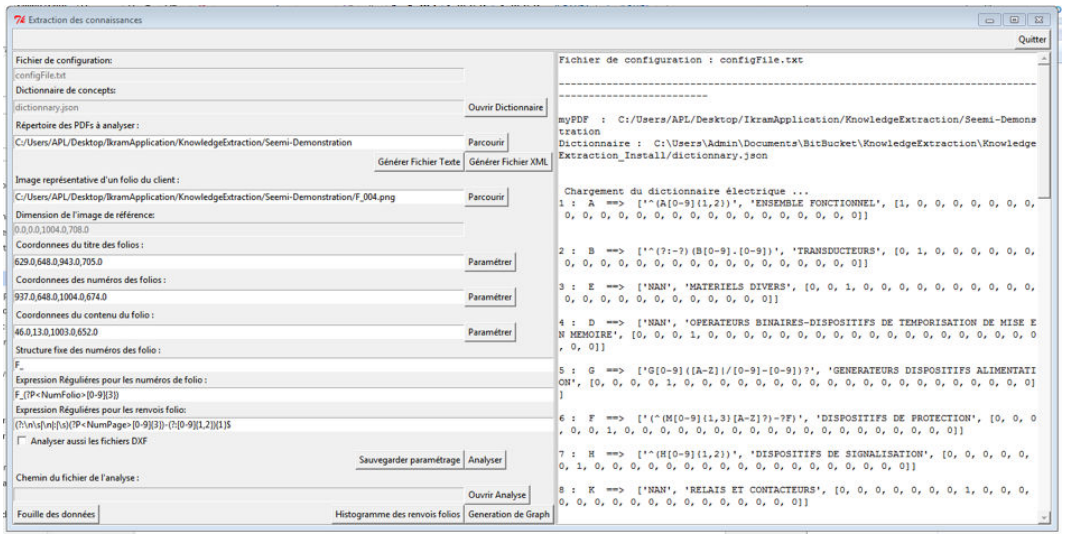


FIGURE 1.13 – Interface d'accueil du prototype d'extraction des connaissances réalisé

La figure 1.13 présente une capture écran de l'interface du prototype et le tableau 1.2 liste les librairies majeures en python utilisées pour la réalisation du logiciel. Ce prototype a pour but d'accompagner l'entreprise lors de son analyse du métier de ses prospects et clients, afin qu'elle puisse plus rapidement s'approprier le métier, les concepts et les habitudes de représentation de ses clients. Pour cela, le prototype est actuellement en cours de test au sein de l'entreprise Algo'Tech afin d'évaluer son utilité sur différents cas clients (connus en interne mais aussi nouveaux), ses limites mais aussi imaginer ses futures fonctionnalités. Il sera évalué en fonction du gain de temps qu'il procure aux experts-Algotech et l'exactitude des données extraites des schémas clients. Le manuel utilisateur décrivant l'ensemble du prototype et des fonctionnalités a été mis en annexe A.

Librairie	Version	Description
TKinter		Pour la réalisation de l'interface
PDFminer	20140328	Pour la fouille de PDF
Graphviz	0.5.2	Pour la représentation graphique des arbres de clusters sous forme de graphes
Matplotlib	1.4.3	Pour la réalisation des calculs
Scipy	0.16.0	Pour les techniques de data mining
Openxyl	2.4.0	Pour l'export excel

TABLE 1.2 – Liste des librairies pythons utilisées pour la réalisation du prototype : versions et objectifs de chacune d'entre elles

1.8 Conclusion des travaux préliminaires

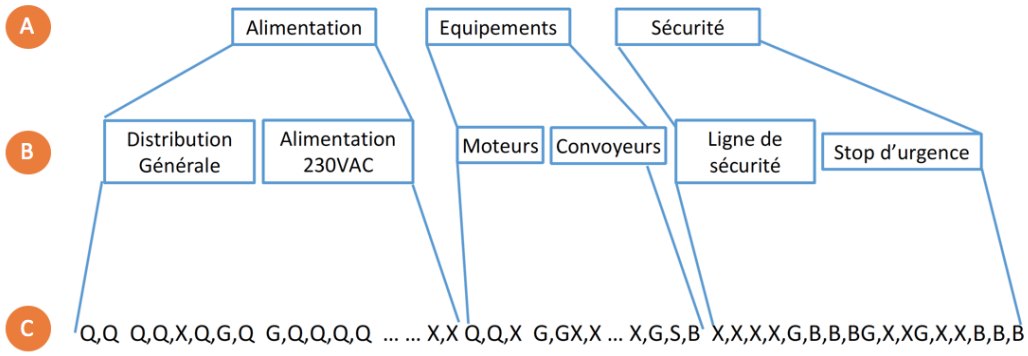


FIGURE 1.14 – Structure hiérarchique et séquentielle dans un schéma électrique :
A - Séquence de concepts métier,
B - Séquence de folios,
C - Séquences de composants électriques

Grâce aux travaux décrits dans les sections précédentes (1.4 et 1.5), nous avons mis en évidence l'existence d'une connaissance implicite dans l'arrangement des composants électriques au sein de chaque folio. Celle-ci est à la fois hiérarchique et séquentielle dans chaque schéma électrique. Lors de la première approche, la représentation d'un schéma électrique sous la forme d'un graphe original permet de décliner un schéma électrique sous la forme de plusieurs séquences de folios liés entre eux par des renvois folios (figure 1.14, label A). Cette même approche, en permettant d'abstraire un schéma sous la forme d'un graphe de concepts métiers globaux permet de transformer les séquences de folios en séquence de concepts métiers (figure 1.14, label B). Enfin, la seconde approche permet elle de représenter chaque folios par sa séquence de composants électrique. Chaque séquence de folios, peut être représentée une suite de séquences de composants électriques mises bout à bout. Il s'agit du niveau le plus bas et le plus détaillé pour décrire un schéma électrique (figure 1.14, label C). La hiérarchie va donc dans le sens suivant, du plus abstrait au plus détaillé : du concept métier, au folio, à la séquence de composants électriques.

Ces travaux préliminaires nous ont permis de confirmer l'existence d'une dimension implicite dans les schémas électriques et de localiser au moins une partie de cette connaissance au niveau des séquences de composants électriques. Au niveau technique, le prototype résultant de ces travaux permet de vérifier cette dimension chez d'autres clients et prospects. Des développements sont encore nécessaires pour avoir une version finalisée de l'outil (afin qu'il puisse correspondre à plus de cas clients) et le hisser à un niveau de prototype fini.

Pour la suite des travaux, nous avons choisi de nous focaliser sur l'analyse des séquences de composants électriques puisqu'elles sont porteuses de connaissances explicites et implicites, à savoir, les règles de connectivité électriques, et les habitudes de représentation d'une entreprise-cliente. De plus, étant donné le niveau le plus détaillé de description d'un schéma, nous pensons qu'elles peuvent conduire à la reconstruction de la grammaire d'un client. Nous nous appuyons pour cela sur le domaine spécifique du machine learning qui étudie l'apprentissage de séquences par des réseaux de neurones récurrent. Nous nous intéressons notamment à l'extraction des règles cachées dans ces séquences à partir de ces réseaux de neurones récurrent.

Chapitre 2

Réseaux de neurones récurrents : De l'apprentissage séquentiel à l'extraction de règles de grammaire

Table des matières

L'extraction de règles à partir de réseaux de neurones apprenant des séquences nous conduit à décrire deux éléments : un réseau de neurones récurrents (RNN pour *Recurrent Neural Network*) qui apprend les séquences et une extraction des règles encodées par le réseau de neurones. Le choix du premier est déterminant pour la qualité des résultats du second. La figure 2.1 présente les différentes étapes d'extraction de règles à partir d'un RNN : de l'apprentissage de séquences, au processus d'extraction des règles sous la forme d'un automate. Dans ce chapitre, nous allons définir dans un premier lieu quelques concepts importants dans nos travaux. Cette brève introduction nous per-

mettra d'aborder de manière sommaire les notions essentielles de nos travaux, avant de les détailler. Nous aborderons aussi la question de la gestion du temps dans les systèmes connexionnistes et des différentes méthodes existantes pour cela. Nous présenterons ensuite les caractéristiques des RNN, avant de décrire en détail deux modèles qui ont chacun marqué une rupture avec les modèles antérieurs : le *Simple Recurent Network* (SRN) et les *Long Short Term Memory*(LSTM). En dernière partie, nous aborderons la question de l'extraction des règles et patterns encodés dans un RNN, une fois un apprentissage séquentiel réalisé. Nous passerons alors en revue les différents moyens d'observer a posteriori ce qui se passe dans un RNN.

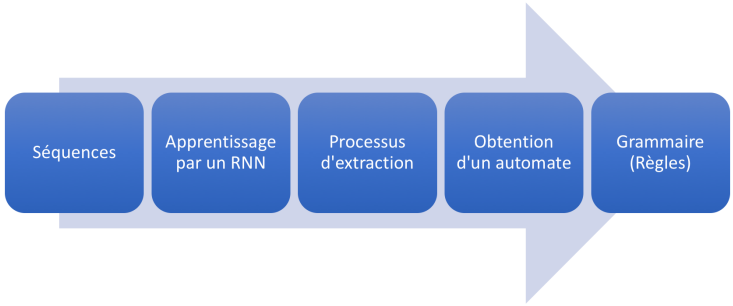


FIGURE 2.1 – Présentation schématique du processus d'extraction des règles à partir d'un RNN

2.1 Introduction : Des RNN à l'extraction des règles, quelques définitions

Dans cette première partie de ce chapitre, nous allons définir ce qu'est une séquence, ce que sont les règles que nous souhaitons extraire. Nous allons aborder les notions de grammaire formelle et d'automates et expliquer notre intérêt pour ces notions. Nous définirons ensuite les notions de contexte et de dépendances séquentielles, qui constituent les différentes composantes des séquences en les accompagnant d'exemples. Notre représentation du temps dans les séquences sera aussi explicitée. Enfin, nous présenterons les étapes de notre approche avant d'aborder dans les parties suivantes de ce chapitre la littérature existante pour chacune d'elles.

Une séquence est définie comme une suite d'éléments mis et traités les uns à la suite des autres. Sa longueur peut être finie (ex : des chaines de caractères ou des mots) ou infinie (ex : flux), et la position d'un élément dans la séquence est désignée par son index. Les éléments peuvent être des chiffres, des caractères, des mots, des symboles, etc. Nous souhaitons à travers nos travaux,

exploiter les séquences de composants électriques, telles qu'elles sont présentes dans les schémas, afin d'extraire les règles de connectivité électrique (i.e. l'expertise) qui y sont cachées. Par règle, nous entendons la formulation simple d'une convention ou d'un principe vérifié, comme par exemple les règles de succession des composants d'une grammaire. A travers l'extraction de ces règles, nous souhaitons extraire la grammaire formelle de connectivité électrique d'un client.

Une grammaire formelle est un formalisme qui définit une syntaxe, ensemble des règles régissant l'usage d'éléments et qui définit un langage formel. Une grammaire formelle se compose d'un ensemble de symboles ("lettres" du langage) qui sont soit terminaux (aux extrémités) soit non-terminaux, d'un point de départ et d'un ensemble de règles. Il existe différentes grammaires formelles (générale, contextuelle, algébrique et régulière) que nous ne détaillerons pas ici. Pour plus d'informations, le lecteur pourra se référer à Chomsky [1956]. Dans la littérature, une grammaire formelle est souvent représentée sous la forme d'un automate. Il s'agit d'un graphe orienté composé de nœuds et de transitions. Les transitions ont toutes un sens : allant d'un nœud source vers un nœud destination. On fera référence au graphe par le terme graphe-automate dans les parties suivantes. Une grammaire formelle permet de définir un langage formel, à savoir un ensemble d'éléments agencés de telle sorte à respecter les règles (la syntaxe) de cette grammaire. La lecture des transitions d'un graphe-automate permet de générer des séquences qui représentent les instances de ce langage formel [Jacobsson, 2005]. Ces séquences sont qualifiées de grammaticales car leur agencement respecte les règles de la grammaire formelle. Dans le cas où une séquence ne respecte pas une grammaire, elle est dite non-grammaticale. Ces notions de respect ou non des règles sont importantes et seront notamment utilisées lors du chapitre 3. A ce stade, nous n'irons pas plus loin dans la définition de la grammaire et du langage formel. Nous donnerons plus de détails par la suite, lorsque nous aborderons les grammaires utilisées pour nos travaux en chapitre 3.

Si la syntaxe représente la forme des séquences, il est important d'aborder et de définir ici la notion de sémantique, le fond des séquences. Le terme de "sémantique" désigne le sens (la signification) qui émerge de l'agencement des éléments au sein de séquences. En linguistique, la sémantique désigne les signifiés (éléments dont on parle) qui représentent la représentation mentale des concepts associés aux éléments dont on parle [De Saussure, 1989].
La sémantique globale d'une séquence dépend de deux notions. La première est celle de dépendances séquentielles (l'ordre des éléments). Il s'agit des contraintes d'intégrité qui déterminent les dépendances entre deux ensembles d'attributs en termes d'ordre [Golab *et al.*, 2009]. Par exemple les séquences A-B-C et A-C-B sont différentes du fait de leur ordre. La seconde notion est celle de contexte

temporel. Ce terme désigne l'ensemble des éléments passés (aux objets ou entités) ayant lieu avant un élément à l'instant t et qui ont une influence sur son interprétation. La distance de l'élément donné en terme de pas de temps avec les éléments précédents permet de fournir une mesure de ce contexte. Le contexte temporel peut porter sur l'ensemble des éléments de la séquence : il peut être local (sur un petit ensemble d'éléments qui se suivent), s'étendre sur plusieurs éléments, ou encore porter sur les extrémités d'une séquence. En voici deux exemple ci dessous.

	t_0	t_1	t_2
séquence 1	B	P	X
séquence 2	B	T	V

Dans ce premier exemple, le contexte temporel est local : Si le but est de prédire l'occurence d'un élément, la différence au temps t_2 entre X pour la première séquence et V pour la seconde, découle de la différence au pas de temps -1.

	t_0	t_1	t_2	t_3	t_4	t_5
séquence 1	A	C	C	C	C	F
séquence 2	B	C	C	C	C	E

Dans ce second exemple, le contexte temporel s'étend entre les extrémités d'une même séquence, soit cinq pas de temps. A l'instant t_5, la séquence 1 reçoit F si elle a commencé par A au temps t_0 et la séquence 2 reçoit E si elle a commencé par B au temps t_0. Dans cet exemple, ce sont les éléments à t_0 qui déterminent l'élément à t_5. Nous souhaitons souligner l'importance de la notion de "contexte temporel" dans notre étude des séquences de composants électriques. Ces dernières obéissent à des règles de connectivité électrique. Le choix d'un composant peut donc dépendre de l'élément précédent direct, mais aussi de composants placés au tout début d'une séquence (4 ou 6 folio en amont). La notion de contexte est très importante dans nos travaux, puisqu'elle est une partie de la connaissance métier cachée dans les séquences de composants électriques.

Extraire des règles cachées dans des séquences implique donc d'être capable de prendre en compte le contexte temporel, tout en détectant les ambiguïtés et en reconnaissant les symboles. Nous allons aborder dans la section suivante la gestion du temps dans les systèmes connexionnistes et notamment les différentes architectures permettant de gérer le contexte temporel. Nous aborderons ensuite plus en détail la solution que nous avons choisie, à savoir les réseaux de neurones récurrents.

2.2 Le temps dans les systèmes connexionnistes

Les réseaux de neurones artificiels (ANN pour *Artificial Neural Network*) ont été introduits comme modèle rudimentaire du traitement de l'information dans le cerveau humain [Gelly, 2017]. Depuis leur émergence dans les années 40, de nombreuses évolutions et architectures ont vu le jour dans des domaines très variés tels que la reconnaissances d'image, de la parole, ou encore la traduction automatique. Différents types d'apprentissage pour les ANN existent selon la tâche considérée. Nous définissons ici brièvement trois d'entre eux : l'apprentissage supervisé, non-supervisé et par renforcement. L'apprentissage supervisé consiste à utiliser un couple de données (entrée-sortie attendue). Le réseau ajuste son apprentissage (ses poids) en utilisant l'erreur entre la sortie fournie et la sortie attendue. On parle d'apprentissage supervisé car le réseau se voit fournir le résultat attendu. Dans un apprentissage non-supervisé, le réseau cherche à extraire des corrélations ou régularités parmi les données fournies en entrée. Il n'a pas d'indication sur le résultat à obtenir. Il modifie ses poids sur la seule base des entrées reçues. Dans le cas d'un apprentissage par renforcement, le réseau apprend à partir d'expériences à réagir selon différentes situations. Son but est de parvenir à optimiser une récompense quantitative au cours du temps.

Différentes architecture de réseaux de neurones, utilisant ces différents apprentissages, ont été conçus pour gérer et intégrer le temps. Ce sont alors des modèles temporels. Nous allons décrire ci dessous sans être exhaustif, les différentes catégories de réseaux de neurones artificiels existantes selon leur prise en compte du paramètre temporel. Pour chacune d'entre elles, nous définirons la classe de ANN, décrirons leur dynamique concernant le traitement de l'information et fournirons des exemples.

Classification des modèles temporels

Pour le choix de la classification des modèles temporels, nous nous inspirons des travaux de Durand [1995]. Bien que datée, nous estimons que la classification proposée par cet auteur correspond bien à notre approche du problème de prise en compte du temps par les réseaux de neurones. Pour un complément d'information, le lecteur pourra se référer à la thèse plus récente de Khouzam [2014] qui traite aussi de la question de l'encodage du temps dans les réseaux de neurones dynamiques. Celle-ci sera brièvement abordée par la suite. Durand [1995] distingue deux grandes classes de réseaux de neurones temporels : ceux qui représentent la séquence dans sa totalité à l'extérieur du réseau, et ceux qui intègrent le temps à l'intérieur de leur architecture. Dans le premier cas, cela revient à un problème de classification, car le temps est une donnée supplémentaire fournie en entrée. Les réseaux impliqués sont acycliques. Dans

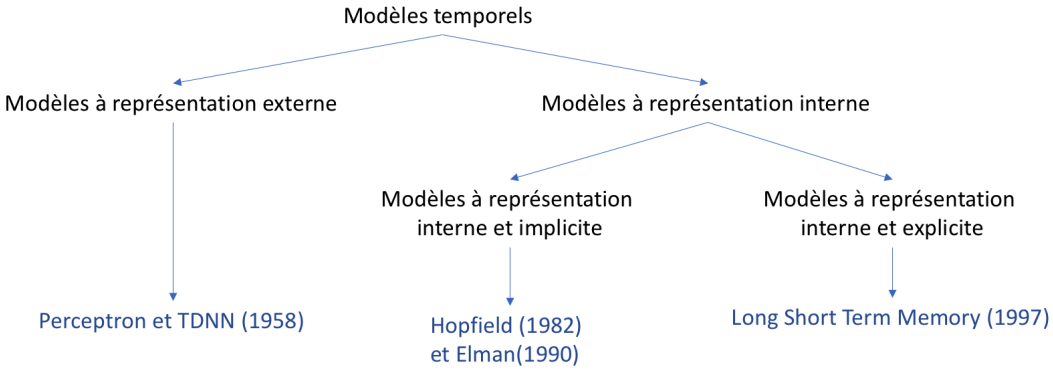


FIGURE 2.2 – Présentation de la classification des modèles temporels avec exemple de modèle pour chaque type

le second cas, le réseau est récurrent (comporte des boucles) et encode implicitement le temps au fur et à mesure qu'il traite les informations. Dans cette seconde catégorie il faut distinguer les réseaux de neurones récurrents qui ont une représentation implicite du temps et ceux qui ont en une représentation explicite. Les modèles ci-dessous sont énumérés selon cette classification.

Modèles à représentation externe : le Percetron et TDNN

Dans ce type de modèle, le temps est traité comme une dimension équivalente aux autres. On dit alors qu'il est "spatialisé". Par exemple, pour la tâche de classification de séquences, une séquence évoluant dans le temps sera représentée par une image. C'est cette dernière qui est fournie au réseau.

Parmi les réseaux utilisés pour cette tâche, nous allons présenter le percetron, et le TDNN (*Time Delay Neural Network*).

- **Le Perceptron simple et Perceptron multi-couches**

Le Perceptron simple est un réseau dit feed-forward car il propage l'information (l'activation des unités) de la couche d'entrée à la couche de sortie [Rosenblatt, 1958]. C'est un réseau acyclique car son graphe de connexion ne comporte pas de boucle, et dont la dynamique est déclenchée par la réception en entrée d'information. Ce réseau se compose de deux couches : l'entrée et la sortie, ce qui implique une seule matrice de poids [Martinez, 2011].

Un perceptron multi-couches est une extension du perceptron simple. Désigné par le sigle MLP (pour *Multi-layer Perceptron* en anglais), il est l'exemple le plus utilisé quand il s'agit de présenter un réseau de neurones acyclique feed-forward. Un percetron multi-couches est doté de plusieurs couches : l'entrée, la sortie et une ou plusieurs couches cachées. Si le réseau possède n couches,

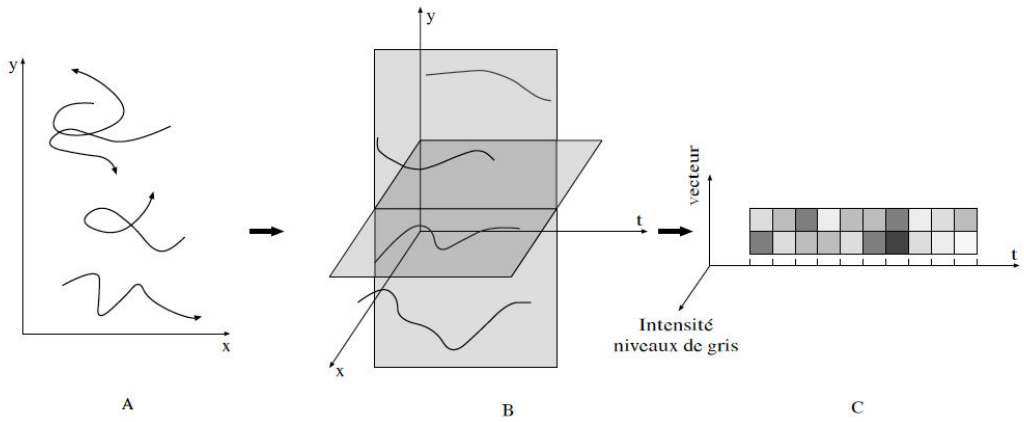


FIGURE 2.3 – Classification de séquences pour les modèles à représentation externe du temps.
A - Représentation de quatre séquences dans un espace de stimuli en deux dimensions (x,y).
B - Représentation de quatre séquences dans un espace de stimuli en trois dimensions (x,y, t), t codant le temps afin d'en dégager une image.
C - Représentation d'une image (portion de B) selon des vecteurs codant pour des intensités de niveaux de gris. Image extraite de Durand [1995]

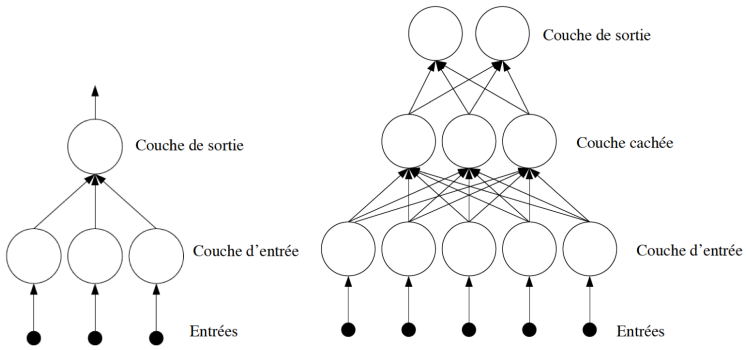


FIGURE 2.4 – Exemples d'un perceptron simple, à gauche et d'un perceptron multi-couches à droite. Image adaptée de Rougier [2000]

il possède alors $n-1$ matrice de poids (une entre chaque suite de couches). Le MLP à une couche cachée est théoriquement un approximateur universel de fonctions. En théorie, il suffit d'ajouter un nombre de neurones suffisant au niveau de la couche cachée pour approximer n'importe quelle fonction non linéaire [Gelly, 2017].

L'algorithme le plus répandu pour l'apprentissage d'un perceptron (simple ou multi-couches) est la méthode de rétro-propagation de l'erreur (error back-propagation en anglais) de Rumelhart *et al.* [1985]. Cet algorithme consiste à partager l'erreur commise par le réseau en sortie à l'ensemble des neurones du réseau. Il modifie tous les poids du réseau selon la fraction de l'erreur commise par chaque neurone. Tout le réseau apprend de l'erreur, et non seulement les couches supérieures. Cet algorithme sera détaillé ultérieurement à l'occasion de la description du SRN, section 2.3.1. La figure 2.4 présente des exemples de perceptron simple et de perceptron multi-couches.

Dans la question de la modélisation temporelle qui nous préoccupe, le MLP ne possède pas de gestion du temps. Si un MLP doit apprendre la séquence "1,2,**3**,4,5,4,**3**,2,1" en associant chaque élément à son successeur, il ne saura pas faire la distinction entre les deux occurrences du chiffre 3, et ne réussira donc pas à prédire les chiffres 4 ou 2. Néanmoins, il possède la capacité de généraliser son apprentissage par rapport à des données déjà apprises. Il peut donc reconnaitre des séquences, qui sont sensiblement similaire à celles déjà apprises, comme par exemple les données présentées en figure 2.3, label C, où le temps est a été spatialisé. La faiblesse de ce réseau réside dans les données apprises : si celles-ci comportent des distorsions temporelles, alors les vecteurs présentés au réseau en entrée peuvent être très différents de ceux appris. Ce réseau ne peut s'appliquer aux cas où les données sont soumises à de fortes variations temporelles comme c'est le cas pour la parole.

- **Time Delay Neural Network, TDNN**

Le réseau neuronal à retardement (TDNN) est un réseau dont le but principal est de classer les motifs de façon invariante, c'est-à-dire sans détermination préalable explicite du début et de la fin du motif. Le TDNN a d'abord été proposé pour classifier les phonèmes dans les signaux vocaux pour la reconnaissance vocale automatique, où la détermination automatique de segments précis ou de limites de caractéristiques est difficile ou impossible. Le TDNN reconnaît les phonèmes et leurs caractéristiques acoustiques / phonétiques sous-jacentes, indépendamment des décalages temporels, c'est-à-dire de la position dans le temps. Un signal d'entrée est augmenté de copies retardées en tant qu'autres entrées, le réseau de neurones est invariant au décalage temporel puisqu'il n'a aucun état interne. Il s'agit d'un réseau multi-couches dont la composante de

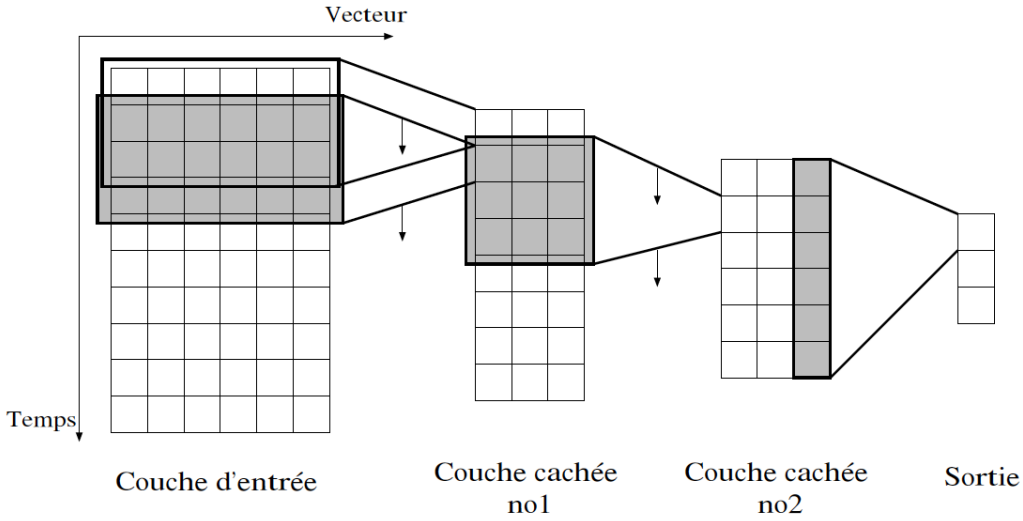


FIGURE 2.5 – Architecture du TDNN. Image issue de Durand [1995]

base est le MLP. Ce réseau possède la capacité de reconnaissance de décalage temporel au niveau du signal d'entrée, mais conserve les limitations des MLP au niveau de la distorsion temporelle. Tout comme pour le MLP, l'algorithme pour l'apprentissage est celui de la rétropropagation de l'erreur. Une alternative à ces modèles a été envisagée : elle consiste à fournir aux réseaux un flot d'information, évènement par événement. Le système fournit ainsi une réponse au bout d'un temps fini. C'est ce que nous allons voir dans ce qui suit.

Modèles à représentation interne et implicite : le modèle de Hopfield et les RNNs à couches

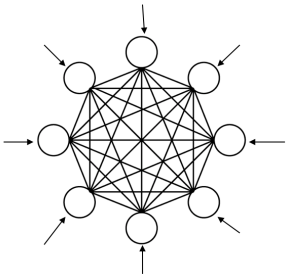


FIGURE 2.6 – Le modèle de Hopfield. Image extraite de Durand [1995]

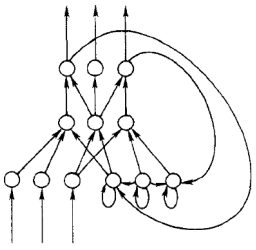


FIGURE 2.7 – Le modèle de Jordan. Image extraite de Servan-Schreiber *et al.* [1991]

Les RNNs sont des réseaux de neurones qui comportent un cycle dans leur graphe de connectivité. Ces cycles permettent au réseau d'entretenir une information en se l'envoyant à lui même. Cela change la dynamique du réseau de

neurones et l'amène à s'auto-entretenir. Ces modèles étaient souvent plébiscités notamment pour le traitement automatique de la parole, et plus généralement de séquences, car leurs caractéristiques leur permettent d'apprendre, de stocker et de prendre en compte l'information contextuelle passée lors de traitement de l'information à l'instant présent. Dans le cas de RNN bidirectionnels, ils prennent aussi en compte l'information contextuelle future [Gelly, 2017]. Un système dynamique est un système dont l'état courant dépend des états précédents. Autrement dit, le passé est codé par des états internes du système. Les RNNs que nous allons décrire obéissent à cette règle. Nous allons les décrire et expliquer comment le temps est pris en compte en leur sein.

- **Le modèle de Hopfield**

Le modèle de Hopfield [Hopfield, 1982], figure 2.6, est un réseau qui se compose d'une seule couche où toutes les unités sont interconnectées. Cette couche représente à la fois l'entrée du réseau et sa sortie. Ce réseau est assimilé à une mémoire adressable par son contenu : les connaissances mémorisées étant distribuées dans le réseau et non localisées à une adresse, il est possible de récupérer l'entièreté d'une donnée, juste en présentant une version dégradée (partielle ou bruitée) [Rougier, 2000]. Du point de vue temporel, le modèle de Hopfield est un système dynamique, puisque ses états à $t+1$ sont dépendants des états à t [Durand, 1995]. C'est ce qui fait de lui un réseau temporel. Il est à noter que le modèle d'Hopfield possède quelques limites : quelle que soit sa taille, le réseau ne peut stocker qu'un nombre fini d'états stables. Si le seuil est dépassé, le phénomène d'"oubli catastrophique" apparait : l'ajout d'exemples supplémentaires conduit à la suppression d'exemples précédemment appris. D'autres systèmes ont été développé en suivant le même type de formalisme que Hopfield, il s'agit de réseaux de neurones aléatoires [Samuelides et Cessac, 2007]

- **Les réseaux de neurones récurrents à couches**

Les réseaux de neurones à couches sont une catégorie particulière des réseaux de neurones récurrents qui ont pour spécificité d'être composés de couches (couche d'entrée, couche(s) cachée(s) et couche de sortie) et apprennent des couples (d'entrées, sorties) comme les perceptrons. Ces réseaux reprennent à la fois l'idée de la propagation avant et de la rétropropagation de l'erreur, et l'idée de la récurrrence de Hopfield [Hopfield, 1982]. L'entrée globale du réseau se propage à la fois en avant dans le réseau de couche en couche, mais elle contient aussi une information sur le passé [Martinez, 2011]. Plusieurs architectures ont été proposées pour ce type de réseau mais nous n'allons aborder ici que le modèle de Jordan. Nous souhaitons à travers la présentation de ce modèle illustrer la particularité et l'adéquation de cette catégorie de réseaux

pour notre tâche.

Le modèle de Jordan, figure 2.7, a été proposé en 1986 par Jordan [1986]. Son architecture est similaire à celle d'un perceptron multi-couches doté d'une seule couche cachée et dont les neurones sont tous interconnectés entre les couches successives. Sa particularité réside dans le fait que sa couche d'entrée comporte des unités de contexte. Ces dernières reçoivent à chaque pas de temps une copie de l'activation de la couche de sortie et une information due à leur boucle récurrente. Au pas de temps t, le réseau reçoit ainsi à la fois l'entrée courante de l'instant t et une partie de l'information passée à $t-1$. Cette connexion récurrente d'une unité de contexte à elle-même donne une dynamique (mémoire) individuelle à l'unité. Autre particularité importante à souligner, **les données représentant les séquences sont présentées de manière ordonnée au réseau**. Lors du traitement d'une séquence, à chaque pas de temps, le réseau reçoit les informations relatives à l'élément courant. Les connexions récurrentes lui permettent de garder une trace en interne (i.e. d'encoder un état) qui reflète implicitement les évolutions passées du réseau. C'est donc ce traitement séquentiel des données et la présence de connexions récurrentes qui permettent au réseau de développer sa représentation interne et sa gestion du temps.

Le SRN est une variante du modèle de Jordan qui possède aussi des unités de contexte. Ces dernières se distinguent à deux niveaux : elles reçoivent une copie de l'activation de la couche cachée (et non la couche de sortie comme pour le modèle de Jordan) et n'ont pas de connexion récurrente. Ce modèle sera décrit plus en détail par la suite dans la section 2.3.1.

Le comportement d'un RNN peut être assimilé à celui d'un automate à états finis [Durand, 1995; Casey, 1998]. Dans le cas des RNN à couches, la configuration d'activité des unités de contexte en entrée code l'état du système. L'autre partie de l'entrée, fournit l'information courante. C'est en fonction de ces deux informations, que le réseau finit par tomber dans un nouvel état. Ce comportement mime celui d'un automate à états finis et permet par le suivi des éléments reçus à chaque pas de temps de pouvoir reconstituer finement l'automate. Servan-Schreiber *et al.* [1991] ont proposé une application de ce type de réseau : faire apprendre une grammaire à un RNN à couches via des séquences de symboles ordonnés, afin d'observer l'automate extrait du RNN.

Modèles à représentation interne et explicite

Les LSTM représentent un modèle qui possède une représentation interne et explicite du temps. Les LSTM ont été proposés initialement par Hochreiter et Schmidhuber [1997]. L'inspiration à leur origine était le souhait de trouver

un nouveau modèle permettant de pallier les faiblesses des réseaux de neurones récurrents traditionnels. Les auteurs ont proposé un nouveau système avec une unité complexe, comparable à un automate, dotée d'une circuiterie interne reliant et exploitant des éléments simples, à savoir des neurones artificiels utilisant la rétropropagation pour leur apprentissage. Le modèle LSTM se distingue de ses prédécesseurs par une nouveauté : les cellules mémoires. Ces unités spéciales sont utilisées, en plus d'unités standards (neurone artificiel standard), au niveau de la couche cachée d'un réseau de neurones, afin de permettre le maintien d'information en mémoire pour de longues périodes de temps. Ce sont ces cellules mémoires qui permettent de garder une représentation explicite du passé. Ces modèles seront détaillés dans la section 2.3.2.

Parmi les ressources les plus utiles pour comprendre les RNNs et avoir une vision globale, figurent une thèse de doctorat [Gers, 2001] et une revue portant sur les dernières évolutions des RNN [Lipton *et al.*, 2015]. Ces deux travaux présentent à la fois les RNNs ainsi que les dernières innovations dans le domaine des LSTM.

Autre classification et autres réseaux de neurones

Dans ses travaux de thèse, Khouzam [2014] aborde la question du temps dans les réseaux de neurones en s'intéressant à ceux qui peuvent traiter les séquences temporelles. Il distingue quatre types de réseaux pour cela : les réseaux feed-forward, des réseaux récurrents, (que nous avons présentés précédemment), les réseaux dynamiques avec lignes à retard et les réseaux auto-organisés. Inspirées du cortex cérébral, les cartes auto-organisées de Kohonen (SOMs)[Kohonen, 1982] utilisent des techniques d'apprentissage non supervisé possèdant différentes variantes. Elles permettent de visualiser notamment la connectivité locale en deux dimensions et de distinguer des groupes dans des ensembles de données. Cette caractéristique permet le traitement de données qui possèdent des relations spatiales entre elles (comme par exemple les pixels d'une image [Martinez, 2011]) ou des données "statiques" (non temporelles). Le SOM est un réseau de neurones compétitif dont le fonctionnement se base sur un apprentissage non-supervisé. Chaque neurone est relié à l'ensemble des entrées. A chaque nouvelle entrée, il y a un neurone déclaré gagnant : le *Best Matching Unit*. C'est celui dont la valeur est la plus proche de la donnée présentée. Ce neurone et ses voisins sont modifiés (changement de valeur) afin de s'adapter au mieux à l'entrée courante. Cette spécialisation de la région de la carte permettra ensuite à ces neurones de mieux répondre aux prochaines entrées similaires. En fonction de la distance des voisins au neurone gagnant, l'impact du changement décroît. La carte est ainsi "déformée" au fur et à mesure des entrées afin de s'adapter aux données reçues [Rougier, 2000; Hinaut, 2013]. De nombreuses variantes temporelles des SOMs existent et sans entrer

dans les détails nous n'en citerons que deux : *Temporal Kohonen Map* et *Recursive SOM* [Khouzam, 2014].

Nous sommes conscients qu'il existe différents angles sous lesquels la question de l'encodage du temps dans les réseaux de neurones peut être traitée et qu'il existe aussi une multitude de réseaux de neurones (certains plus récents) que ceux abordés ici. Nous n'avons donc pas la prétention d'établir une liste exhaustive des réseaux de neurones. Notre but est de donner ici les axes de réflexion principaux concernant la question du traitement de séquences.

2.3 Deux modèles de RNN : SRN et LSTM

Parmi les solutions plébiscitées ces dernières années pour l'apprentissage séquentiel, on trouve le SRN et les LSTM. Le SRN a marqué une rupture dans les années 90, en faisant partie des premiers réseaux récurrents à couche prenant en compte une information au cours du temps afin de fournir une séquence d'informations en sortie. Quant aux LSTM, ils brillent par leurs performances et leur capacité à résoudre des tâches complexes qui n'avaient pas encore été résolues par les précédents algorithmes de RNNs [Hochreiter *et al.*, 2001]. Nous allons aborder en profondeur ces deux modèles dans la section suivante en détaillant leur architecture, les algorithmes d'apprentissage et surtout leur apport dans notre problématique d'apprentissage implicite de règles à partir de séquences. Le lecteur familier avec le SRN peut passer à la section suivante car cette partie peut lui sembler simple. Néanmoins au vu du contexte CIFRE de la thèse et afin d'assurer la bonne transmission des connaissances à l'entreprise partenaire, nous avons souhaité expliciter en détail le principe et fonctionnement de ce modèle.

2.3.1 Le SRN

C'est en 1990, que Elman [1990] a proposé un réseau récurrent simple (SRN, Simple Recurrent Network (SRN) en anglais). Il s'agit d'un RNN à couches qui représente le temps de manière interne et implicite grâce à ses connexions récurrentes, qui lui permettent d'encoder le contexte temporel des entrées [Cleeremans et McClelland, 1991].

Architecture du modèle

Le SRN est un réseau de neurones récurrent qui se compose de 3 couches : une couche d'entrée, une couche cachée et une couche de sortie. La particularité du réseau réside dans la couche d'entrée : celle-ci se compose d'unités d'entrée et d'unités de contexte. Si les premières ont pour fonction de transmettre une donnée (une lettre, un caractère ou un mot) au réseau à un instant t,

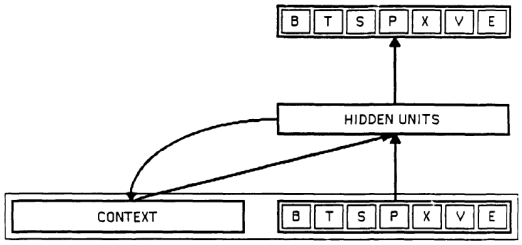


FIGURE 2.8 – Réseau de neurones récurrent simple de Elman décrit par Servan-Schreiber *et al.* [1988]

les secondes permettent de sauvegarder une trace des représentations internes du réseau lors des précédentes entrées. Pour cela, dans un SRN, il y a autant d'unités de contexte que d'unités dans la couche cachée. Lors de la propagation en avant, à chaque pas de temps, la couche de contexte reçoit une copie du motif d'activation de la couche cachée.

Ainsi au pas de temps suivant, la couche cachée reçoit un motif global : concaténation de celui des unités d'entrée et celui des unités de contexte. Autrement dit, à chaque pas de temps, la couche cachée reçoit deux informations : celle de l'instant t (le présent) et celles des représentations internes passées du réseau dues à l'entrée à t-1.

Au niveau des poids, l'ensemble est modifié lors de l'apprentissage, à l'exception de ceux de retour (permettant la copie) entre la couche cachée et les unités de contexte.

Apprentissage du modèle : Algorithme de rétropropagation

•Apprentissage et fonctionnement

Le SRN apprend des couples (entrée, sortie). Lors de l'apprentissage d'une séquence, l'entrée du couple correspond à l'élément courant reçu par le réseau à l'instant t, et la sortie correspond à l'élément suivant que le réseau doit traiter à l'instant t+1 (voir l'algorithme 1). Pour son apprentissage, et après la phase de propagation avant où l'information est propagée à travers les couches, le réseau utilise l'algorithme de rétropropagation du gradient pour ajuster ses poids [Rumelhart *et al.*, 1985]. L'idée derrière cet algorithme est de partager l'erreur globale en sortie du réseau à l'ensemble de neurones, de la couche de sortie à la couche d'entrée. Les fractions d'erreurs permettent ainsi d'ajuster les poids entre les différentes couches du réseau. Pour plus de détails techniques, Gelly [2017] réalise une explication approfondie de l'algorithme de rétropropagation. L'algorithme 1 présente l'algorithme de rétropropagation (aussi présent en annexe D).

A ce stade, il nous faut souligner l'importance de certains paramètres dans

l'apprentissage du SRN : le taux d'apprentissage et le momentum. Le premier se définit comme le pas d'apprentissage du réseau qui détermine la vitesse à laquelle le SRN apprend. Généralement compris entre 0 et 1, le taux d'apprentissage lorsqu'il est proche de 1, permet un apprentissage rapide du réseau (car les poids varieront sur de grandes amplitudes). Néanmoins, il peut aussi induire trop d'oscillations dans les valeurs des poids. Ces derniers ne convergeant pas, empêche le réseau d'apprendre. A contrario, lorsque le taux d'apprentissage est plus proche de 0 que de 1, le réseau apprend lentement en faisant converger petit à petit les poids vers de bonnes valeurs. Dans ce cas aussi, la valeur est importante : Si le taux d'apprentissage est trop petit, le nombre d'itérations peut être trop grand et le réseau peut aussi être bloqué dans un minimum local. Quant au momentum, il s'agit d'un paramètre inertiel, compris entre 0 et 1, qui permet de stabiliser l'apprentissage en évitant les oscillations ou de rester coincé dans un minimum local. Il permet la prise en compte des erreurs précédentes : si sa valeur est faible, les erreurs précédentes ne sont pas prises en compte lors de l'apprentissage à l'instant t. Si sa valeur est élevée, le calcul des poids à l'instant t prend alors en compte les erreurs précédentes.

D'autres algorithmes d'apprentissage existent telle la rétropropagation dans le temps (BPTT pour Back Propagation Through Time) et l'apprentissage récurrent en temps réel (RTRL pour Real Time Recurrent Network). Nous ne donnerons qu'une brève définition ici. La rétropropagation dans le temps (BPTT) a pour idée centrale le dépliement du réseau neuronal récurrent dans le temps, afin qu'à chaque instant il puisse être assimilé à un réseau de neurone feed-forward comme le percetron [Rumelhart *et al.*, 1985]. La figure 2.9 illustre un exemple de RNN déplié. Conceptuellement, à chaque pas de temps, le réseau est copié en une nouvelle instance, il y a donc autant d'instances du réseau que de pas de temps traité dans une séquence. Pour chaque instance, le réseau reçoit une entrée et l'état interne du pas de temps précédent, puis fournit une sortie. Les erreurs sont calculées et les poids mis à jour en utilisant la rétropropagation standard, avec une restriction cependant : les poids équivalents dans l'ensemble des instances doivent être les mêmes. Spatialement, chaque pas de temps du RNN déroulé peut être vu comme une couche supplémentaire puisque le réseau dépend de l'ordre des données présentées et que l'état interne du pas de temps précédent est pris comme entrée à chaque pas de temps.

L'apprentissage récurrent en temps réel (RTRL) [Williams et Zipser, 1989] est un algorithme d'apprentissage qui calcule le gradient d'erreur exact à chaque pas de temps. Autrement dit, cet algorithme calcule les dérivées des états et des sorties par rapport à tous les poids pendant la propagation avant à chaque pas de temps. Il est donc adapté aux tâches d'apprentissage en ligne. Mathématiquement plus simple que BPTT, il est néanmoins très lent : plus

Algorithme 1 Algorithme d'apprentissage du SRN

Pour tout séquence de taille n **faire**

Pour $t = 0$ à $(n - 1)$ **faire**
# Etape 1 : propagation avant
entrée_réseau = séquence[t]
sortie_attendue = séquence[$t + 1$]
sortie_obtenue = SRN.propagation_avant(entrée_réseau)
Erreur_SRN = sortie_attendue - sortie_obtenue
# Etape 2 : propagation arrière
# Etape 2a : calcul de l'erreur pour chaque unité de la couche de sortie
# σ' : dérivée de la fonction d'activation σ des unités du SRN
Pour tout unité k de la couche de sortie **faire**
δ_k = Erreur_SRN[k] * σ '(sortie_obtenue[k])
Fin Pour
Etape 2b :Calcul de l'erreur au niveau de la couche cachée pour chaque unité j et au niveau de la couche d'entrée
Pour tout couche du réseau, de la couche cachée à la couche d'entrée **faire**

Pour tout unité j de la couche courante c **faire**
σ' : dérivée de la fonction d'activation σ des unités du SRN
$w_{(k,j)}$: poids provenant de j et allant vers k
y_j : somme des entrées de l'unité j
$\delta_j = \sigma$ '(y_j) * $\sum_k(w_{(k,j)}$ * $(\delta_k))$
Fin Pour
Fin Pour
Etape 2c : Mise à jour des poids
Pour tout poids $w_{(j,i)}$ **faire**
x_{ji} : entrée associée au lien entre l'unité i vers l'unité j
Variation du poids = Taux d'apprentissage * δ_j * x_{ji} + momentum * (Variation du poids(t-1))
Nouveau poids = Ancien poids + Variation du poids
Fin Pour
Fin Pour
Fin Pour

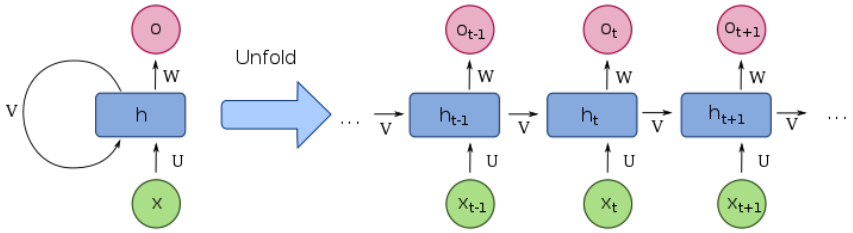


FIGURE 2.9 – Représentation d'un RNN déplié. Image issue de Deloche [2017]

le réseau comporte d'unités, plus il comporte de poids, et plus les calculs sont longs et complexes. Cet inconvénient font que l'algorithme ne peut être utilisé que sur de très petits réseaux. Par ailleurs, cet algorithme appliqué au SRN nécessite un nouveau paramétrage à chaque nouveau corpus de données [Cernansky et Benuskova, 2003]. Cette contrainte rend donc la combinaison SRN-RTRL difficile à utiliser dans différents contextes comme nous souhaitons le faire dans travaux présentés par la suite. Une version détaillée du RTRL est fournie par Giles *et al.* [2008]. Pour un descriptif détaillé des différents algorithmes d'apprentissage des RNN, le lecteur pourra se référer à Jaeger [2002].

- **Exemple : apprentissage de la séquence BTXSE**

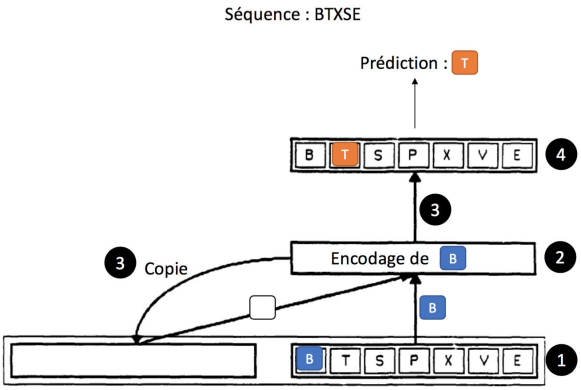


FIGURE 2.10 – Etapes de la propagation avant à un instant t dans un SRN

Nous allons décrire l'apprentissage du SRN pour la tâche suivante : l'apprentissage de la séquence BTXSE issue de la grammaire de REBER. Nous donnerons plus de détails sur cette grammaire ultérieurement dans la section 3.2.2. A chaque pas de temps, le réseau doit apprendre à prédire le symbole suivant, selon le symbole reçu à l'instant t mais aussi les symboles précédemment vus. Il reçoit un vecteur binaire en entrée, représentant pour le symbole

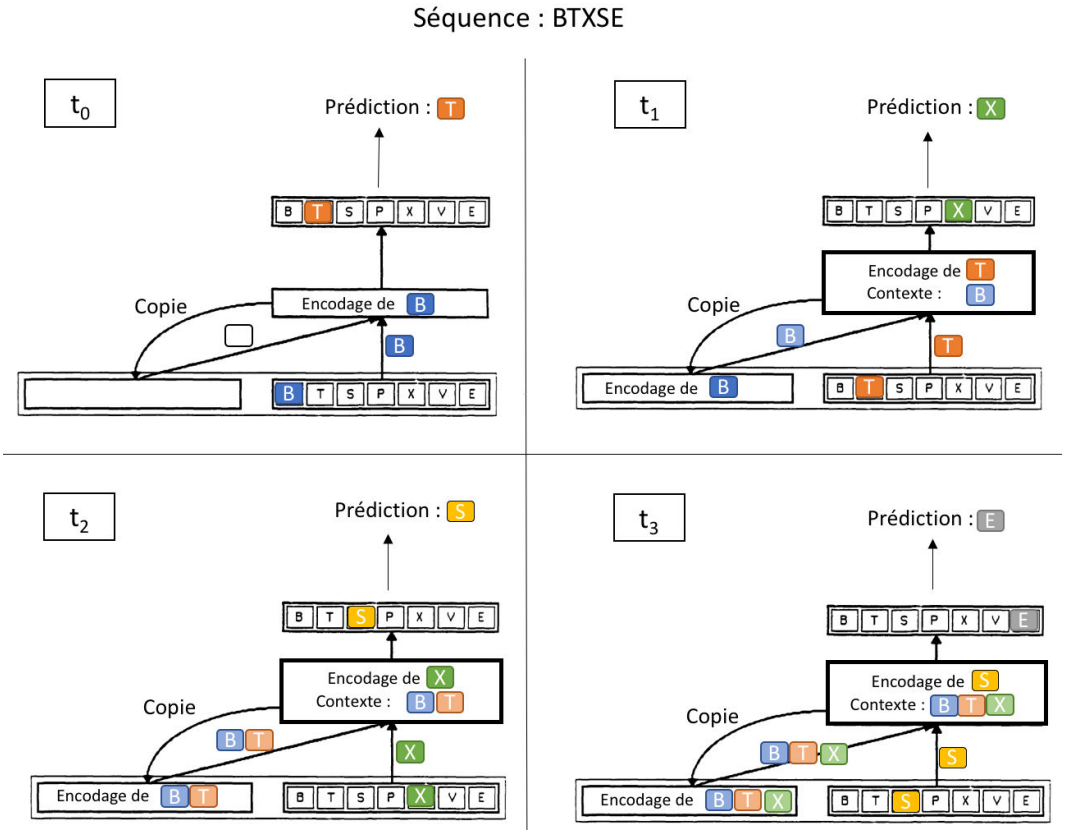


FIGURE 2.11 – Etapes de la propagation avant dans un SRN pour toute une séquence. Chaque symbole de la séquence est représenté par une couleur. Les symboles dotés d'une couleur plus claire représentent la version encodée des symboles présentés en entrée au réseau.

courant, et fournit un vecteur de valeurs réelles, représentant les probabilités de succession de chacun des symboles. Selon ces valeurs, le réseau peut prédire un ou plusieurs successeurs à un symbole à l'instant t. Les tables 2.1 et 2.2 montrent un exemple de l'état du réseau après la présentation du symbole B à t_0 et du symbole T à t_1. Dans le premier cas, le réseau prédit le symbole T (sa valeur d'activation est de 53%). Dans le second cas, le réseau prend en compte l'état interne du réseau à t_0 grâce à la couche de contexte et prédit X (sa valeur d'activation est de 56%).

Symbole	B	T	S	P	X	V	E
Sortie du réseau	00	53	00	38	01	02	00
Couche cachée			01	00	10		
Couche de contexte			00	00	00		
Symbole	B	T	S	P	X	V	E
Entrée du réseau	100	00	00	00	00	00	00

TABLE 2.1 – Etat du réseau après la présentation du symbole "B" (après apprentissage) de la séquence BTXSE. Les valeurs d'activation en interne (comprises entre 0 et 1) sont affichées ici sur une échelle de 0% à 100%. Exemple adapté de Servan-Schreiber *et al.* [1991].

Symbole	B	T	S	P	X	V	E
Sortie du réseau	00	01	39	00	56	00	00
Couche cachée			84	00	28		
Couche de contexte			01	00	10		
Symbole	B	T	S	P	X	V	E
Entrée du réseau	00	100	00	00	00	00	00

TABLE 2.2 – Etat du réseau après la présentation du symbole "T" (après apprentissage) de la séquence BTXSE. Les valeurs d'activation en interne (comprises entre 0 et 1) sont affichées ici sur une échelle de 0% à 100%. Exemple adapté de Servan-Schreiber *et al.* [1991]

L'algorithme d'apprentissage étant supervisé, le réseau a besoin pour apprendre, d'une entrée et d'une sortie attendue. A chaque pas de temps, le SRN compare la sortie qu'il a fournie avec celle attendue afin de calculer l'erreur et d'ajuster son apprentissage. Il a donc besoin du couple d'éléments : (élément entrée, élément suivant attendu). Pour chaque séquence de taille N, il y a N-1 couples d'éléments. Nous nommerons ces couples des échantillons. Pour

la séquence BTXSE, les échantillons sont : (B,T) (T,X), (X,S) et (S,E) et chaque symbole est encodé selon un vecteur binaire de taille 5 (car nous avons 7 symboles) ne comportant qu'une seule unité non nulle. Lors de la phase de l'apprentissage, le SRN apprend la séquence BTXSE selon le schéma suivant :

Instant t_0, échantillon (B,T) : le SRN traite le premier couple d'éléments. Il reçoit en entrée le vecteur binaire désignant B en entrée. Seuls les neurones de la couche d'entrée voient leur activité changer. Celles des unités de la couche de contexte restent nulles car celles-ci ne sont pas impactées par les entrées, mais par la couche cachée dont elles reçoivent une copie du pattern d'activation. L'activation de la couche d'entrée dans son intégralité (couche d'entrée + couche de contexte) est alors propagée à la couche cachée (Figure 2.10, étape 1). Les neurones de la couche cachée voient leur activité changer. Le pattern d'activation de la couche cachée encode alors l'état interne du réseau selon l'entrée reçue (Figure 2.10, étape 2). Le pattern d'activation de la couche cachée représente alors un encodage des caractéristiques des patterns d'entrée qui sont pertinents pour la tâche de reconnaissance d'une portion de séquence Cleeremans et McClelland [1991]. Le pattern d'activation de la couche cachée est alors simultanément, copié dans la couche de contexte pour l'itération suivante et propagé à la couche de sortie (Figure 2.10, étape 3). Cette dernière traite l'activation reçue. Le réseau émet une réponse sous la forme d'un pattern d'activation qui définit un élément ou plusieurs éléments selon les successeurs prédits (Figure 2.10, étape 4). Le SRN compare alors la sortie obtenue (sa prédiction) avec la sortie attendue (le vecteur binaire décrivant l'élément T), et calcule l'erreur. Cette dernière est rétropropagée en utilisant l'algorithme de rétropropagation pour ajuster les poids des connexions du SRN (à l'exception de ceux entre la couche cachée et la couche de contexte) et cela avant la présentation de l'élément suivant au réseau.

Instant t_1, échantillon (T,X) : A l'instant t1, le SRN traite le second couple d'élément (T, X). Il reçoit donc en entrée le vecteur binaire désignant T en entrée et suit les mêmes étapes décrites dans l'étape t_0. La différence réside ici dans le pattern d'activation de la couche cachée qui n'est plus nul. En effet, il contient une copie du pattern d'activation de la couche cachée à l'instant précédent t_0. Lorsque la couche cachée reçoit une information, elle reçoit à la fois l'activation de la couche d'entrée à l'instant courant, mais aussi celle de la couche de contexte qui contient une information sur la représentation du réseau à l'instant précédant. Ainsi la couche cachée peut encoder des informations à propos des caractéristiques pertinentes des 2 premiers symboles. La couche cachée réalise donc des calculs selon une information présente et passée.

Instant t_2, échantillon (X,S) : la couche de contexte contient une information à propos de l'élément à t_0 et t_1 (qu'elle communique en même temps

que la couche d'entrée à la couche cachée) et s'apprête ensuite à recevoir celle à propos de t_2. Le SRN doit prédire le symbole suivant, sachant qu'il vient de voir successivement les lettres BTX.

Instant t_3, échantillon (S,E) : la couche de contexte contient à présent une information à propos de l'élément à t_0, t_1 et t_2 et s'apprête ensuite à recevoir celle à propos de t_3 et du dernier échantillon.

La figure 2.11, illustre l'apprentissage de la séquence BTXSE aux différents pas de temps décrits ci-dessus. Arrivé à la fin de l'apprentissage d'une séquence, le réseau accumule deux types de "mémoires" : sa couche de contexte informe sur le passé récent qu'il vient de traiter, c'est la mémoire sur le court terme. Et ses poids, contiennent la mémoire sur le long terme : leur ajustement permettra au réseau de faire la bonne prédiction la prochaine fois qu'il sera soumis à la même séquence. Si le SRN apprend plusieurs séquences, il apprend "plusieurs contextes" qui le rendent capable, de distinguer entre deux séquences proches mais différentes. Si le réseau apprend les séquences BTSXSE et BPTVVE, il prédira X après avoir "vu" BTS et V après avoir "vu" BPT.

Limites des modèles RNN actuels

Les RNN souffrent de 3 problèmes, que nous allons détailler ici, qui nuisent à leurs performances pour l'apprentissage de séquences longues.

Le premier problème est celui de l'atténuation du contexte temporel. Lors du fonctionnement du réseau SRN par exemple, celui ci accumule, au fur et à mesure du temps, une copie des patterns d'activité passés dans sa couche de contexte. Or, cette couche finit à un moment par être saturée, du fait de sa taille limitée, ce qui génère un oubli des passés lointains au privilège de passé plus récents [Hinaut, 2013]. Dans le cas de Hopfield, le phénomène porte le nom d' "oublie catastrophique". et représente par conséquent un frein à l'apprentissage de dépendances sur le long terme.

Le second problème est celui du dépliement des RNN lors notamment de l'utilisation de la rétropropagation dans le temps (BPTT) (que nous avons présenté précédemment) comme algorithme d'apprentissage (section 2.3.1). Pour chaque séquence, à chaque pas de temps, une nouvelle instance du réseau est créée. Un RNN finit par ressembler une fois déplié, à un réseau profond doté de plusieurs couches. Si les séquences sont longues (un millier de pas de temps), le nombre d'instances (de couches) à prendre en compte pour les calculs sera tout aussi important et il faudra autant de dérivées à calculer pour chaque mise à jour des poids. La complexité de calcul engendré par cela, peut avoir pour conséquence l'évanouissement du gradient (que nous expliquerons ci dessous)

ou l'explosion des poids et rendre l'apprentissage lent et inefficace. Cet algorithme d'apprentissage, porteur d'une complexité spatiale et de calcul, n'est donc pas adéquat pour l'apprentissage de séquences avec des contextes temporels portant sur un nombre important de pas de temps.

Enfin, le troisième problème est celui de l'évanouissement du gradient lors de l'apprentissage qui survient dans les réseaux profonds (dotés de plusieurs couches comme le MLP). Ce problème est dû aux méthodes d'apprentissage utilisant certaines fonctions d'activation. Nombres d'entres elles (par exemple, sigmoïde ou tangente) projetant leur entrée dans une très petite plage de sortie ([0, 1] par exemple pour la fonction sigmoïde) d'une manière non linéaire, aboutissent à l'obtention d'un gradient faible.

Lorsque plusieurs couches sont empilées, le gradient transitant par ces couches devient de plus en plus faible. Par conséquence, même un grand changement dans les paramètres de la première couche n'a pas un grand effet sur les couches de sortie (couches profondes). Or les méthodes basées sur le gradient apprennent la valeur d'un paramètre en comprenant comment une petite modification de la valeur de ce paramètre affectera la sortie du réseau. Si une modification de la valeur du paramètre entraîne une très faible modification de la sortie du réseau, le réseau ne peut tout simplement pas apprendre le paramètre de manière efficace, ce qui pose problème. Dans le problème de l'évanouissement du gradient, les gradients de la sortie du réseau, par rapport aux paramètres dans les premières couches, devenant extrêmement faibles, empêchent l'apprentissage et l'ajustement des paramètres des couches profondes du réseau. Ce problème tend à s'aggraver à mesure que le nombre de couches dans l'architecture augmente.

Pour finir, un rappel plus qu'une limite : les performances des RNN dépendent des paramètres choisis. Nous avons présenté précédemment les notions de taux d'apprentissage et de momentum. Si le choix de ces paramètres peut nuire ou améliorer les performances du SRN, cela est vrai pour les RNN. Nous listerons aussi le nombre d'unités du RNN en général et le nombre d'unités de la couche cachée dans le cas du SRN comme autres paramètres importants. Ce dernier, avec le taux d'apprentissage et le momentum sont les paramètres que nous avons testés et fait varier dans notre étude des performances du SRN en section 3.3.1.

Apport du SRN dans l'apprentissage implicite

Dans le cadre de l'étude de l'apprentissage implicite, Reber défend l'idée que cet l'apprentissage s'explique par un processus inconscient d'abstraction des règles qui régissent une situation [Witt, 2010a]. D'après lui, les sujets arrivent à discriminer les stimuli qui correspondent aux règles, de ceux qui ne

les respectent pas, sans être capables d'expliquer ces règles. Il est rejoint ensuite par Axel Cleeremans [Cleeremans *et al.*, 1989; Cleeremans et McClelland, 1991]. Ce dernier introduit un cadre théorique qui unifie les données et les modèles existants sur l'apprentissage implicite (i.e inconscient), ainsi qu'un modèle informatique détaillé de la performance humaine dans les situations d'apprentissage de séquences : le SRN. Dans une série d'expériences de mesure du temps de réaction lors de la réalisation d'un choix, Cleeremans a ainsi montré, à l'aide d'un SRN capable de prédire parfaitement les éléments successifs de séquences générées à partir de grammaires à états finis, que des sujets humains développaient une sensibilité à la structure temporelle des séquences qui leur ont été présenté. Et cela sans que leur connaissance explicite de la séquence soit modifiée. Il s'intéresse alors au processus d'abstraction des règles dans le SRN pour tenter de comprendre celui qui a lieu de manière inconsciente au niveau cognitif chez les sujets humains et démontre ainsi la capacité du SRN à encoder des règles cachées dans des séquences [Servan-Schreiber *et al.*, 1988, 1991]. En utilisant des techniques de clustering hiérarchiques réalisées à différents moments de l'apprentissage du SRN sur les patterns d'activité de la couche cachée, les chercheurs ont démontré une modification de l'organisation interne de l'encodage des éléments.

En d'autres termes, le SRN modifie sa représentation interne des règles abstraites des séquences au fur et à mesure de l'apprentissage, jusqu'à ce qu'elles lui permettent de ne faire que des prédictions correctes.

Apport du SRN dans le domaine des RNN

L'utilisation de l'algorithme d'apprentissage BPTT pour les RNN induit des réseaux dotés de plusieurs couches tel un réseau profond. Ces réseaux ont souvent de faibles performances dues au problème de l'évanouissement du gradient, sans oublier la complexité de calcul que cela engendre (deux limites que nous avons détaillés précédemment section 2.3.1). Par son architecture, et notamment la présence d'une récurrence dans son architecture, le SRN (doté de seulement 3 couches) permet d'apprendre des séquences temporelles avec un algorithme standard de back propagation. En évitant d'utiliser le BPTT, le SRN permet d'éviter les problèmes liés au dépliement (la complexité de calcul lors de l'apprentissage de longues séquences) et à l'évanouissement du gradient (qui survient si le dépliement aboutit à l'obtention de réseaux profonds).

Discussion

D'un point de vue technique, parmi les modèles connexionnistes, le SRN semble particulièrement adapté pour le matériel séquentiel grâce notamment à son architecture dotée de récurrence. En effet, le modèle développe une sensibilité au contexte temporel à partir du moment où cela est pertinent pour

l'amélioration de ses performances et pour la tâche de prédiction de l'élément suivant [Cleeremans et McClelland, 1991]. De plus, le modèle ne nécessite pas de grandes ressources d'analyse : son architecture élémentaire a pour conséquence des mécanismes d'apprentissages élémentaires car tous les calculs sont locaux à l'élément courant (pas de représentation explicite des éléments précédents).

Néanmoins et malgré ses apports dans le domaine de l'implicite et des RNN, le SRN est un modèle qui souffre d'une limite importante. Son architecture qui lui permet de stocker le passé, est aussi sa faiblesse : Dût à sa taille non-extensible, la couche de contexte finit par être saturée en information sur le passé. Cela implique que chaque nouvelle entrée encodée par le réseau prendra peu à peu le pas sur les informations les plus anciennes. Un passé récent finit par écraser le passé plus ancien, ce qui limite l'utilisation du modèle pour l'apprentissage de séquences longues avec des dépendances sur le long terme [Sutton, 1988; Gers *et al.*, 1999; Hinaut, 2013].

Dans la partie suivante, nous allons aborder le modèle des LSTM, ou Long Short Term Memory, qui a connu un regain d'intérêt ces dernières années et qui offre un nouveau contexte pour l'étude des séquences et des règles cachées les régissant.

2.3.2 Long Short Term Memory, LSTM

Nous avons évoqué brièvement les LSTM précédemment comme des automates complexes utilisant des neurones artificiels pour pallier les faiblesses des RNNs traditionnels. Ces réseaux, nous le verrons plus en détail par la suite, possèdent une architecture complexe mais réalisent des calculs simples, ce qui leur permet essentiellement deux éléments. D'une part, ils explicitent le temps grâce à leur architecture : ils sont considérés proches du fonctionnement du BPTT dans leur manière de gérer les entrées séquentielles. D'autre part, grâce au "Carroussel d'erreur constante", unité linéaire de calcul que nous présenterons ultérieurement, ils peuvent maintenir l'erreur sur plusieurs pas de temps sans que celle-ci disparaisse ou ne soit altérée. Cette dernière caractéristique permet aux LSTM de gérer ainsi de longues séquences sans souffrir du problème de l'évanouissement du gradient ou aucun autre problème lié au dépliement. Dans cette section, nous décrivons plus en détail ce modèle en présentant l'architecture d'un réseau LSTM, mais aussi en détaillant la structure d'une unité LSTM. Nous aborderons ensuite les calculs réalisés lors de la propagation avant et l'algorithme recommandé pour la rétropropagation lors de l'apprentissage. Nous présenterons brièvement aussi les variations les plus connues et utilisées de ces dernières années. Enfin, à la lumière de l'ensemble de ces détails techniques, nous expliquerons l'apport des LSTM dans

l'apprentissage séquentiel.

Présentation des LSTMs

Le concept des LSTM, pour Long Short Term Memory, provient de l'idée suivante : dans un RNN, l'information est stockée sous deux formats. L'activation des unités représente l'historique récent du réseau et donc une mémoire sur le court terme du réseau, alors que le poids des liaisons entre les neurones représente l'expérience accumulée du RNN lors de son apprentissage et donc une mémoire sur le long terme. Forts de ce constat, Hochreiter et Schmidhuber [1997] ont introduit les cellules mémoires, une forme de mémoire intermédiaire permettant de stocker des informations importantes sur une période de temps plus longue que les RNN existants [Lipton *et al.*, 2015]. D'un point de vue technique, les LSTM sont motivées par l'envie de proposer un modèle qui ne souffre pas du problème du débordement ou de l'évanouissement de l'erreur lors de l'apprentissage. En effet, dans un RNN traditionnel tel que Elman, l'apprentissage finit à un moment par induire un signal d'erreur qui soit explose (ce qui induit des poids qui augmentent), soit s'évanouit (surtout lorsque le réseau doit apprendre sur de longs délais ce qui prend un temps prohibitif ou qui ne fonctionne pas du tout). Or sans erreur, il n'est plus possible de faire apprendre au RNN [Hochreiter et Schmidhuber, 1997; Hochreiter *et al.*, 2001]. Depuis 1997, la recherche liée au LSTM est un domaine très actif et de nombreuses variations ont été proposées. Nous allons présenter dans la section suivante la version des LSTM proposée par Gers *et al.* [1999] et qui est la première à avoir créé une rupture en termes de performances avec le modèle d'origine. Nous reprendrons les mêmes notations que celles de l'article de Gers *et al.* [1999] (similaires à celle de Hochreiter et Schmidhuber [1997]) pour la suite de nos travaux.

Architecture d'une unité LSTM

Bien souvent, on retrouve dans la littérature la notion de "cellule LSTM" (LSTM cell) pour désigner une unité LSTM dans sa globalité. Pour éviter toute confusion, nous utiliserons le terme d'unité LSTM lorsque l'on désignera l'ensemble : bloc et cellule. Nous allons ainsi décrire ci-dessous l'architecture globale d'un RNN-LSTM avant de détailler les composants bloc et cellule séparément.

•Comment s'insèrent les unités LSTM dans un RNN-LSTM ?

Dans un RNN tel que Elman, le réseau se compose de 3 couches : une couche d'entrée, une couche cachée et une couche de sortie. La couche cachée

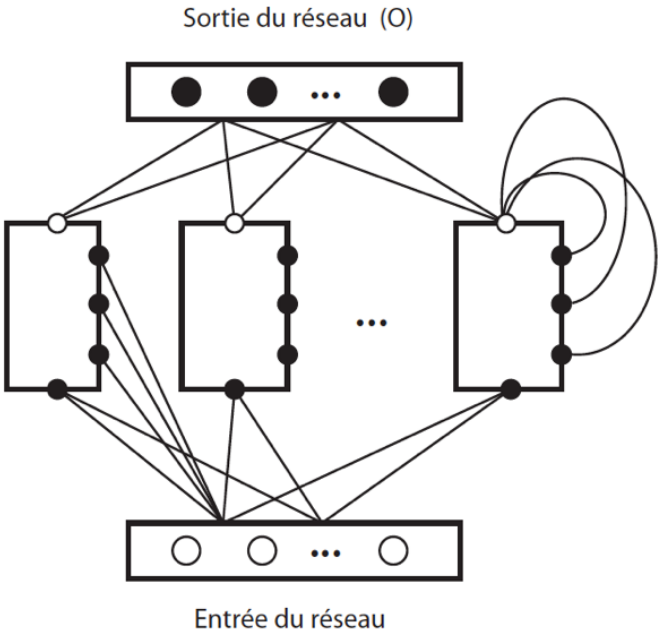


FIGURE 2.12 – Présentation d'un réseau doté d'unités LSTM au niveau de la couche cachée. Image issue de Lapalme [2006]

se compose de neurones artificiels, du même type que ceux de la couche d'entrée et ceux de la couche de sortie. Dans un RNN-LSTM, l'architecture du réseau reste la même, à savoir 3 couches. La différence réside essentiellement dans la couche cachée. Les unités utilisées sont alors des unités LSTM composées de blocs et de cellules. La couche cachée reçoit donc ainsi à l'instant t l'information provenant de la couche d'entrée, mais aussi l'information provenant de la couche cachée à l'instant t-1. Le pattern d'activité de la couche cachée se compose de l'ensemble des sorties des différentes cellules de chaque unité LSTM. La figure 2.12 présente un exemple d'un RNN-LSTM.

•Les Blocs

Deux notions sont importantes à distinguer lorsqu'il s'agit d'un LSTM : les blocs et les cellules. Si le premier terme désigne le contenant, celui qui apprend à laisser entrer, sortir les informations reçues ou qui choisit de les oublier, le second terme désigne le contenu, celui qui traite l'information.
Un bloc LSTM peut avoir une ou plusieurs cellules : il leur distribue l'information de manière égale. Une cellule au contraire n'appartient qu'à un seul bloc. La figure 2.13 illustre le cas d'un bloc doté de 2 cellules et la figure 2.14 celui d'un bloc à une seule cellule. Un bloc LSTM j a la particularité d'avoir $(3+c)$ voies d'entrées de l'information, c désignant le nombre de cellules dans le bloc. Les voies d'entrée de l'information sont la porte d'entrée in_j, la porte d'oubli

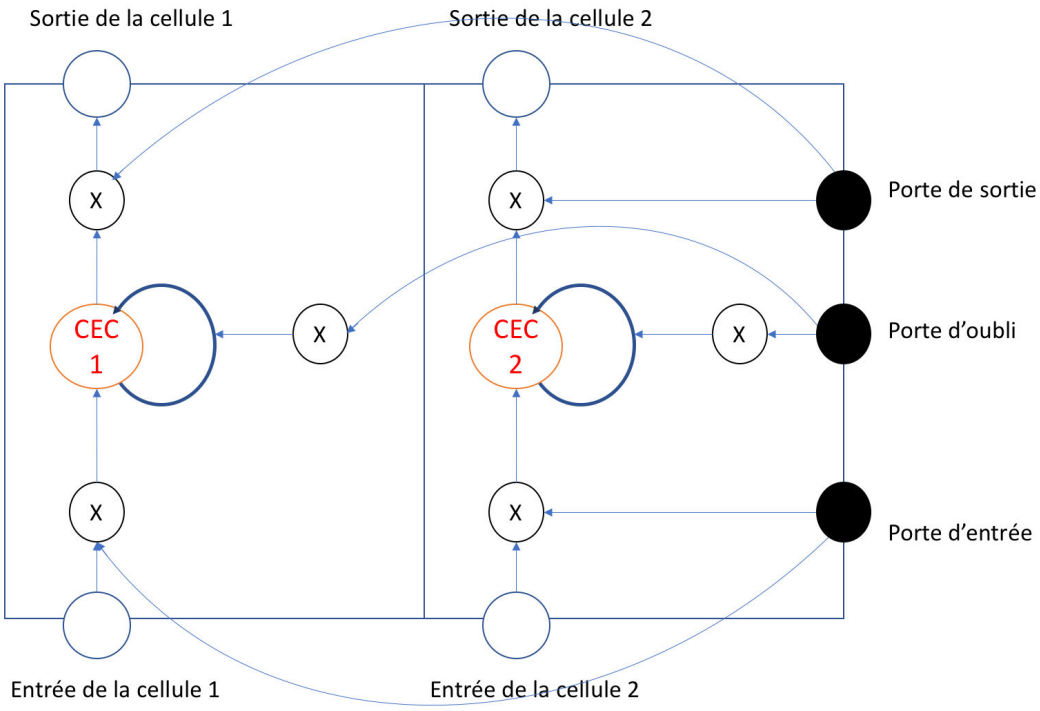


FIGURE 2.13 – Présentation d'une unité LSTM dotée d'un bloc composé de 2 cellules

ϕ_j et la porte de sortie out_j, et les entrées de chaque cellule. Chaque cellule de chaque bloc reçoit la même information. Nous détaillerons cela dans la partie suivante dédiée à la présentation des cellules.

— *la porte d'entrée* : Quand sa valeur est proche de zéro, cette porte empêche les informations extérieures de parvenir aux cellules des blocs et donc de mettre à jour leur valeur. L'information provenant des couches inférieures n'est donc pas transmise aux couches supérieures.

— *la porte d'oubli* : Quand sa valeur est proche de zéro, la cellule oublie son passé. Les cellules du bloc ne prennent alors en compte que le présent.

— *la porte de sortie* : Cette porte détermine si une cellule peut envoyer une valeur aux autres cellules ou non. En effet, quand sa valeur est proche de zéro, la cellule ne fournit aucune information en sortie (valeur proche de 0 aussi)

Chacune des portes reçoit à un pas de temps t, l'information associée au pas de temps t en provenance de la couche d'entrée, mais aussi celle associée au pas de temps t-1 en provenance des cellules de la même couche cachée. Bien

que l'algorithme d'apprentissage soit détaillé par la suite, il semble important de préciser que les poids arrivant sur ces portes sont tous modifiés durant l'apprentissage. Cela implique que chaque bloc apprend à reconnaitre le moment où il faut prendre en compte l'information extérieure, mais aussi les moments où il faut communiquer avec l'extérieur. Soulignons aussi que les portes de sortie décrites ci-dessus, ne faisaient pas partie du modèle original. Elles ont été proposées par Gers *et al.* [1999] et sont depuis proposées dans la majorité des implémentations au vue de l'amélioration des performances [Lipton *et al.*, 2015]

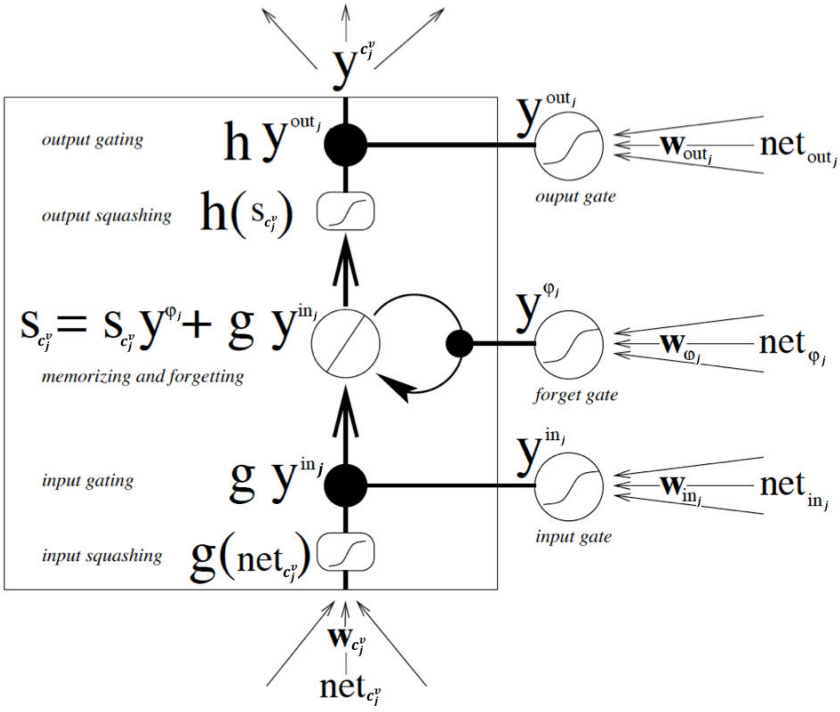


FIGURE 2.14 – Présentation d'une unité LSTM doté d'un bloc, lui même doté d'une cellule. Image issue de Gers *et al.* [1999]

•Les cellules

Une cellule c, appartenant à un bloc j et dont le numéro est v, et que nous noterons c_j^v, possède aussi une voie d'entrée et une voie de sortie (en plus des 3 voies liées aux 3 portes.)

En plus de cela, chaque cellule possède une unité linéaire de calcul nommée "Carrousel d'erreur constante" ou "Constant error carousel" en anglais (CEC), dont l'activation est notée $s_{c_j^v}$. On parle alors d'état de la cellule.
C'est cette cellule qui résout le problème de l'évanouissement du gradient qui empêche l'apprentissage. En l'absence de nouvelles entrées ou de signal d'erreur (lors de la rétropropagation) parvenant à cette cellule, le retour d'erreur local (induit par la boucle récurrente) du CEC reste constant. Il peut être donc

maintenu sur de long pas de temps (nous y reviendrons par la suite).
En effet, grâce aux portes du bloc, l'état de la cellule n'est pas mis à jour à chaque pas de temps. En termes d'informations reçues, une cellule CEC prend en compte différents types d'information : l'information calculée au pas de temps t (l'entrée de la cellule, celle de l'état de la cellule $s_{c_j^v}(t)$, les valeurs des différentes portes $y^{in_j}(t)$, $y^{\phi_j}(t)$ et $y^{out_j}(t)$) mais aussi de l'information relative au pas de temps t-1 (l'ensemble de valeurs de sorties de l'ensemble des cellules de la couche cachée $y^{c_j^v}$(t-1) et l'état de la cellule $s_{c_j^v}$(t-1)). Ce fonctionnement en deux temps est ce qui confère aux LSTMs leur performance.

Fonctionnement d'une unité LSTM

Une unité LSTM, comme nous l'avons vu, se compose de plusieurs éléments : 3 portes et autant d'entrées et de sorties que de cellules. Malgré la complexité de l'architecture, les performances sont au rendez vous grâce notamment à la simplicité des calculs réalisés. En effet, en dehors des sorties des cellules, l'ensemble des autres composants (porte et entrée de cellules) sont des neurones artificiels, qui utilisent la rétropropagation standard pour leur apprentissage. Et en interne, les calculs sont des produits et des sommes, ce qui rend l'ensemble facilement dérivable et manipulable pour l'apprentissage. Une autre manière de représenter une unité LSTM est de la considérer comme une représentation schématique de 3 algorithmes "si alors sinon" (pour les 3 portes) décrit ci dessous (Algorithme 2).

Nous allons décrire dans cette partie le fonctionnement d'une unité LSTM et les différentes équations nécessaires à cela.

•Notations et fonctions d'activations

Pour commencer, il est nécessaire de définir ici quelques notations :
Pour les blocs et cellules :
— j : bloc d'une couche cachée donnée
— v : cellule dans un bloc donné
— c_j^v : cellule v du bloc j
— in_j : la porte d'entrée du bloc j
— ϕ_j : la porte d'oubli du bloc j
— out_j : la porte de sortie du bloc j
— $s_{c_j^v}(t)$) : état s de la cellule v du bloc j

Pour la couche de sortie du réseau :
— k : neurone au niveau de la couche de sortie du réseau
— e_k : erreur calculée au niveau de l'unité k de la couche de sortie du réseau

Algorithme 2 Algorithme de fonctionnement des portes selon trois algorithmes "si alors sinon"

Si $Valeur\_porteEntree \gg 0$ **alors**
  Porte ouverte : Transmission de l'information extérieure à l'intérieur du bloc
  Si $Valeur\_porteOubli \gg 0$ **alors**
    L'état précédent du CEC impacte le calcul de l'état présent
    Si $Valeur\_porteSortie \gg 0$ **alors**
      Porte ouverte : Transmission de l'information intérieure vers l'extérieur du bloc
    Sinon
      Porte fermée : Pas d'information fournie en sortie
    Fin Si
  Sinon
    la cellule oublie son passé
  Fin Si
Sinon
  Porte fermée : Information bloquée
Fin Si

— t^k : sortie (valeur) attendue au niveau de l'unité k

Autres notations :
— m : neurone possédant une liaison avec une unité LSTM
— net_x : activation reçue par l'unité de calcul x (que ce soit une porte ou une cellule ou un neurone artificiel)
— y^m : activité de l'unité de calcul m
— $w_{(l,m)}$: poids provenant de m et allant vers l'unité de calcul l

Trois fonctions d'activation seront citées dans la section ci-dessous :
— $f : x \to f(x)$: Sigmoïde, fonction logistique entre [0,1] telle que

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

— $h : x \to h(x)$: Sigmoïde centrée entre [-1,1] telle que

$$h(x) = 2 * f(x) - 1 \tag{2.2}$$

— $g : x \to g(x)$: Sigmoïde centrée entre [-2,2] telle que

$$g(x) = 4 * f(x) - 2 \tag{2.3}$$

Nous noterons leurs dérivées respectives f', h' et g'

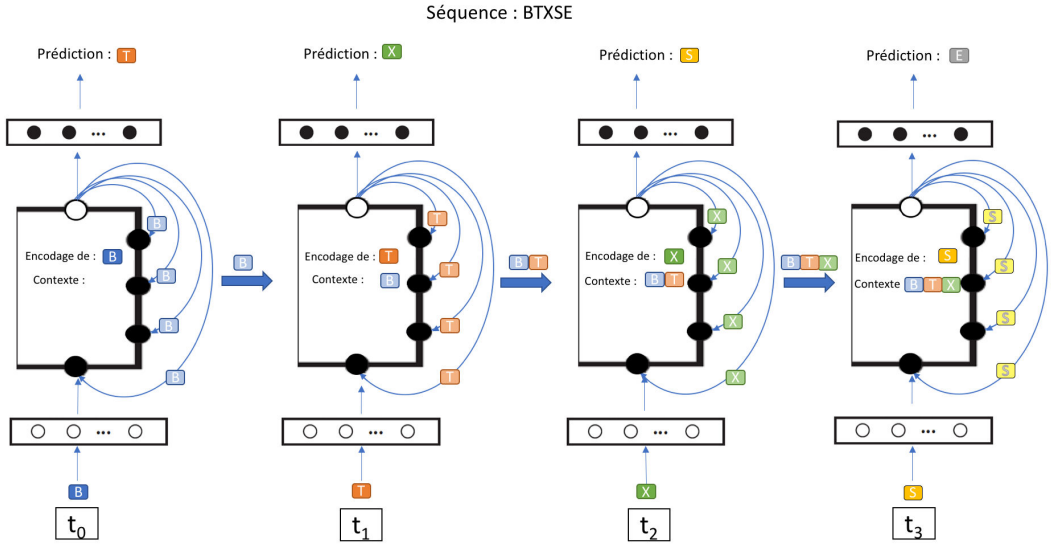


FIGURE 2.15 – Représentation schématique des étapes de la propagation avant dans un RNN-LSTM pour toute une séquence. Chaque symbole de la séquence est représenté par une couleur. Les symboles dotés d'une couleur plus claire représentent la version encodée des symboles présentés en entrée au réseau.

•Propagation avant

La réception par le réseau d'une information au pas de temps t, initie la propagation avant de l'information dans le réseau. Celle-ci doit alors être transmise de la couche d'entrée à la couche cachée puis à la couche de sortie. A l'instant t, chaque bloc de la couche cachée reçoit 2 informations : le pattern d'activité de la couche d'entrée de l'instant t, et le pattern d'activité de la couche cachée de l'instant t-1. Autrement dit, la couche cachée renvoie vers elle-même son propre pattern d'activité et cela grâce à l'information que chaque cellule renvoie vers elle-même, les autres cellules, et les portes des différents blocs. La figure 2.15 présente schématiquement les étapes de la propagation avant dans un RNN-LSTM doté d'une unité LSTM au niveau de sa couche cachée.
Cette information passée et présente est reçue au même moment par l'entrée de la cellule du bloc, la porte d'entrée, la porte d'oubli et la porte de sortie du bloc. Les calculs des activations reçues sont détaillés ci-dessous :

Au niveau des entrées de l'unité LSTM, bloc et cellule

Soit m le nombre de neurones[1] qui possèdent une liaison avec une unité

1. m désigne les neurones de la couche d'entrée, mais aussi les cellules des blocs de la couche cachée. Il s'agit de l'ensemble des neurones artificiels qui envoient une information à

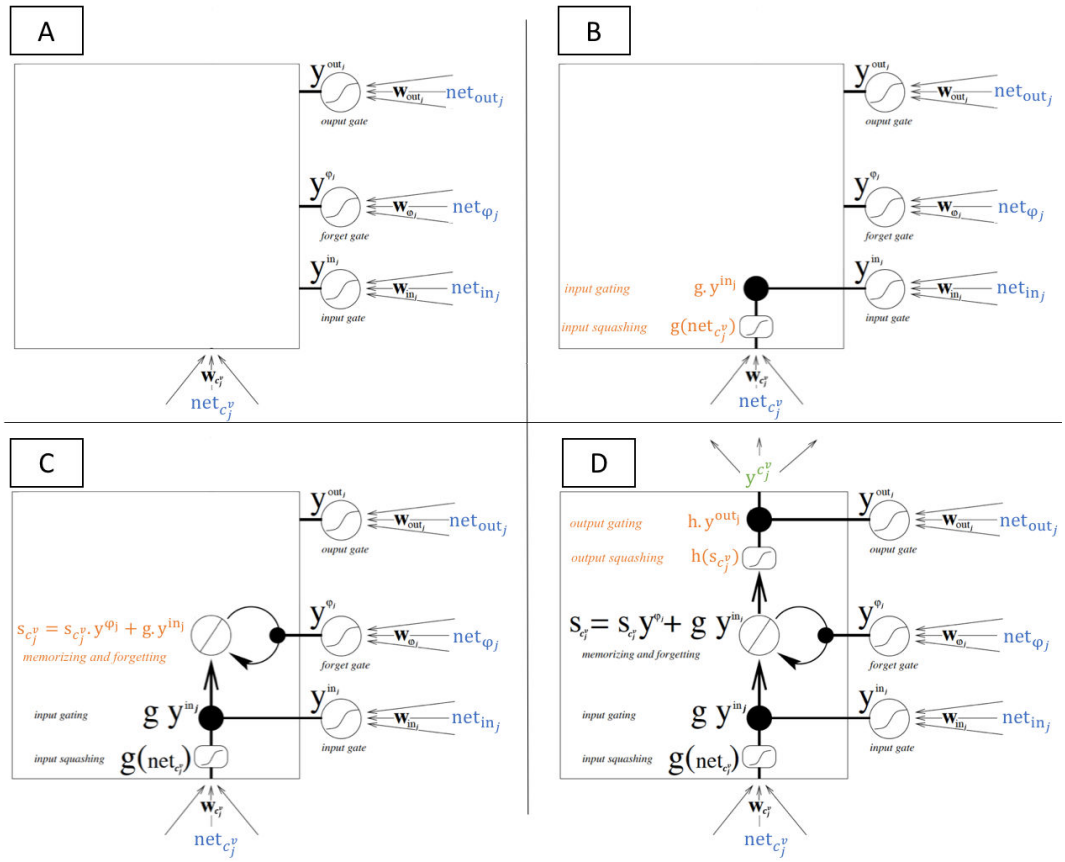


FIGURE 2.16 – Etapes de la propagation avant dans une unité LSTM dotée d'un bloc d'une cellule. Les informations entrantes dans une unité sont en bleu, celle sortante en vert. Les calculs au sein d'une cellule sont en orange.
A - L'unité LSTM reçoit les activations des autres unités du réseau puis calcule les activités des portes du bloc.
B - La cellule calcule l'activité entrante reçue (input squashing) et la module selon la valeur de la porte d'entrée du bloc (input gating).
C - Mise à jour de la valeur du CEC (memorizing) selon l'activité entrante modulée de la cellule et l'activité du CEC au temps précédent modulée par la valeur de la porte d'oublie du bloc (forgetting).
D - La cellule calcule l'activité résultant du CEC (output squashing) et la module selon la valeur de la porte de sortie du bloc (output gating). Selon le résultat, la cellule envoie une activation aux autres unités du réseau.
Images adaptées de Gers *et al.* [1999] (figure 2.14).

LSTM et qui envoient une information au temps t, p le nombre de neurones qui possèdent une liaison avec une unité LSTM et qui envoient une information au temps $t-1$ et soit net_x les activations calculées pour une unité de calcul x :

Pour la cellule, $x = c_j^v$:

$$net_{c_j^v} = \sum_m (w_{(c_j^v,m)} \cdot y^m(t)) + \sum_p (w_{(c_j^v,p)} \cdot y^p(t-1)) \tag{2.4}$$

Pour la porte d'entrée, $x = in$:

$$net_{in_j} = \sum_m (w_{(in_j,m)} \cdot y^m(t)) + \sum_p (w_{(in_j,p)} \cdot y^p(t-1)) \tag{2.5}$$

Pour la porte d'oubli, $x = \phi$:

$$net_{\phi_j} = \sum_m (w_{(\phi_j,m)} \cdot y^m(t)) + \sum_p (w_{(\phi_j,p)} \cdot y^p(t-1)) \tag{2.6}$$

Pour la porte de sortie, $x = out$:

$$net_{out_j} = \sum_m (w_{(out_j,m)} \cdot y^m(t)) + \sum_p (w_{(out_j,p)} \cdot y^p(t-1)) \tag{2.7}$$

La figure 2.16, label A, présente les différentes activations reçues par une unité LSTM dotée d'un bloc d'une cellule. Pour chacune des portes, la fonction d'activation f est appliquée sur les entrées reçues telle que :

$$y^{in_j}(t) = f_{in_j}(net_{in_j}(t)) \tag{2.8}$$

$$y^{\phi_j}(t) = f_{\phi_j}(net_{\phi_j}(t)) \tag{2.9}$$

$$y^{out_j}(t) = f_{out_j}(net_{out_j}(t)) \tag{2.10}$$

Pour la cellule, c'est la fonction d'activation g qui est appliquée sur les entrées reçues :

$$y^{c_j^v}(t) = g(net_{c_j^v}(t)) \tag{2.11}$$

Dans l'unité LSTM, au niveau de la cellule

La mise à jour de l'état de la cellule peut être décrite selon la séquence de calcul suivante :

l'instant t et qui sera pris en compte dans le calcul de l'activation de l'entrée de la cellule

— Etape 1 (figure 2.16, label B) : L'activité de l'entrée de la cellule $g(net_{c_j^v}(t))$ est multipliée par la valeur de la porte d'entrée du bloc $y^{in_j}(t)$. C'est à ce niveau là que le bloc peut empêcher une information non pertinente de venir modifier l'état interne de la cellule. Lorsque la porte d'entrée est close (activation proche de zéro), les entrées non pertinentes, ainsi que le bruit, n'entrent pas dans l'unité LSTM.

$$y^{c_j^v}(t) \cdot y^{in_j}(t) = g(net_{c_j^v}(t)) \cdot y^{in_j}(t) \tag{2.12}$$

— Etape 2 (figure 2.16, label C) : la porte d'oubli joue le rôle de bouton de remise à zéro du passé. Par exemple, lorsque l'étude d'une nouvelle séquence débute, son activité est proche de zéro. Le produit de l'équation 2.13 étant alors nul, la cellule "oublie son passé" et peut commencer à stocker un nouveau passé.

$$s_{c_j^v}(t-1) \cdot y^{\phi_j}(t) \tag{2.13}$$

— Etape 3 (figure 2.16, label C) : Il s'agit de la somme des deux étapes précédentes : autrement dit, la prise en compte du présent connaissant le passé (ou l'absence de passé lorsqu'il s'agit du début d'une séquence).

$$g(net_{c_j^v}(t)) \cdot y^{in_j}(t) + s_{c_j^v}(t-1) \cdot y^{\phi_j}(t) \tag{2.14}$$

L'équation de mise à jour de l'état interne de la cellule à l'instant t est donc la suivante :

$$s_{c_j^v}(t) = g(net_{c_j^v}(t)) \cdot y^{in_j}(t) + s_{c_j^v}(t-1) \cdot y^{\phi_j}(t) \tag{2.15}$$

Il est important de souligner que l'équation 2.15 résume à elle seule la puissance des LSTM puisqu'en une équation dérivable, 3 algorithmes "si alors sinon" sont représentés (voir algorithme 2). Cette équation, réalisée pour chaque cellule, a permis de gagner du temps de calcul. L'utilisation d'algorithmes de conditions "si alors sinon" aurait été plus consommatrice en calcul et moins fiable. Cette équation n'est réalisée que si l'ensemble des portes ont des valeurs supérieurs à 0 (si toutes les conditions "Si alors" de l'algorithme 2 sont remplis). Si les deux premières conditions (porte entrée et porte d'oubli) ne sont pas remplis, le CEC ne reçoit pas d'entrée. N'étant pas mis à jour, l'information en son sein est préservée pour ce pas de temps sans dégradation.

— Etape 4 (figure 2.16, label D) : Une fois l'état interne calculé, il ne reste plus que le calcul de la sortie de la cellule, ce qui fait intervenir la porte de sortie du bloc comme ce qui suit :

$$y^{c_j^v}(t) = h(s_{c_j^v}(t)) \cdot y^{out_j}(t) \tag{2.16}$$

Il est important ici de souligner l'application de la fonction d'activation h sur l'état de la cellule :

$$h(s_{c_j^v}(t)) \tag{2.17}$$

•Propagation arrière

Dans cette section nous avons choisi de présenter une version de l'algorithme de propagation arrière alliant BPTT et RTRL, telle que décrit par Gers *et al.* [1999] et que nous avons choisi d'implémenter pour nos travaux. Dans un RNN-LSTM, les unités de sortie utilisent BPTT pour la propagation arrière du gradient. Les portes de sortie des blocs utilisent, elles, une version tronquée de BPTT et enfin les poids allant aux cellules, aux portes d'entrées et aux portes d'oubli des blocs seront mis à jour via une version tronquée de RTRL. Ce choix se justifie par le fait que grâce à la troncature, toutes les erreurs seront coupées lorsqu'elles sortiront d'une cellule ou d'une porte, bien qu'elles servent à modifier les poids entrants. Ainsi le CEC d'une cellule devient la seule partie du système à travers laquelle l'erreur peut être rétropropagée et maintenue pour toujours. Cela rend les mises à jour des LSTM efficaces sans affecter significativement le pouvoir d'apprentissage : le flux d'erreur à l'extérieur des cellules a tendance à décroître exponentiellement de toute façon [Hochreiter et Schmidhuber, 1997]. L'algorithme de propagation arrière sera donné en entier en annexe D, mais nous abordons ici ses grandes étapes.

Calcul des dérivées durant la propagation avant :
Les calculs de la propagation arrière commencent pendant la propagation avant. En effet, à chaque pas de temps, après le calcul de la sortie du réseau selon une entrée à l'instant t, le modèle RNN-LSTM calcule aussi des dérivées partielles pour calculer le gradient et cela pour les portes d'entrées, les portes d'oubli et les cellules de chaque bloc. L'idée est de calculer la dérivée de la valeur des cellules par rapport aux poids du réseau pour ces voies d'entrée de l'information là.

Ainsi pour chaque bloc j, les dérivées sont calculées comme ce qui suit :
Pour les cellules :

$$dS_{c,m}^{jv}(t) = dS_{cm}^{jv}(t-1) * y^{\phi_j}(t) + g'(net_{c_j^v})(t) * y^{in_j}(t) * y^m(t-1) \tag{2.18}$$

Pour les portes d'entrées :

$$dS_{in,m}^{jv}(t) = dS_{in,m}^{jv}(t-1) * y^{\phi_j}(t) + g(net_{c_j^v}(t)) * f'_{in_j}(net_{in_j}(t)) * y^m(t-1) \tag{2.19}$$

Pour les portes d'oubli :

$$dS_{\phi,m}^{jv}(t) = dS_{\phi,m}^{jv}(t-1) * y^{\phi_j}(t) + h(s_{c_j^v}(t)) * f'_{\phi_j}(net_{\phi_j}(t)) * y^m(t-1) \tag{2.20}$$

avec au temps $t = 0$, $dS^{jv}_{c,m}(0) = 0$, $dS^{jv}_{in,m}(0) = 0$ et $dS^{jv}_{\phi,m}(0) = 0$

Rétropropagation pour une unité LSTM

L'étape suivante est la rétropropagation en elle même. Dans un RNN-LSTM, une erreur est calculée au niveau de chaque unité k de la couche de sortie telle que :

$$\delta_k(t) = f'_k(net_k(t)) * (t^k(t) - y^k(t)) \tag{2.21}$$

Cette erreur $\delta_k(t)$ est ensuite rétropropagée et utilisée pour calculer l'erreur au niveau des portes de sortie, d'oubli, d'entrée et des cellules des blocs comme ce qui suit :

Pour les poids entre les unités k de la couche de sortie et les portes de **sortie** des blocs j :

$$\delta_{out_j}(t) = f'_{out_j}(net_{out_j}(t)) * (\sum_{v=1}^{S_j} h(s_{c^v_j}(t)) * (\sum_m w_{kc^v_j} * \delta_k(t))) \tag{2.22}$$

Pour les poids entre les unités k de la couche de sortie et les portes **d'entrée**, les portes **d'oubli**, ainsi que les **cellules** des blocs j :

$$e_{c^v_j}(t) = y^{out_j} * h'(s_{c^v_j}(t)) * (\sum_m w_{kc^v_j} * \delta_k(t))) \tag{2.23}$$

Mise à jour des poids d'un réseau RNN-LSTM

C'est lors de cette étape que l'on va utiliser les dérivées partielles calculées précédemment :

Pour les unités k de la couche de sortie et les portes de sorties out_j des blocs j :

$$\Delta_{lm}(t) = \alpha * (\delta_l(t)) * y^m(t-1) \tag{2.24}$$

avec $l \in k, i, out$

Pour les cellules :

$$\Delta_{c^v_j,m}(t) = \alpha * e_{c^v_j}(t) * dS^j v_{c,m}(t) \tag{2.25}$$

Pour les portes d'entrée :

$$\Delta_{in,m}(t) = \alpha * (\sum_{v=1}^{S_j} e_{c^v_j}(t) * dS^j v_{in,m}(t)) \tag{2.26}$$

Pour les portes d'oubli :

$$\Delta_{\phi,m}(t) = \alpha * (\sum_{v=1}^{S_j} e_{c_j^v}(t) * dS^j v_{\phi,m}(t)) \tag{2.27}$$

Il faut souligner que selon les modèles, d'autres poids existent et qu'il faut aussi les mettre à jour dans le RNN-LSTM. Ils seront alors mis à jour comme pour les poids entre les unités de la couche de sortie et de la couche cachée.

Les variations les plus connues du modèle

•Les connexions peepholes

Dans nos travaux, nous avons choisi de travailler sur le modèle de LSTM tel que défini par Gers *et al.* [1999]. Mais depuis, de nombreuses variations ont été proposées comme notamment les connexions "peepholes". Proposées la même année et par les mêmes auteurs, ces connexions lient ensemble et directement le CEC aux portes d'entrée et d'oubli sans devoir d'abord être modulées par la porte de sortie Gers et Schmidhuber [2000]. Ils rapportent que ces connexions améliorent les performances sur les tâches de synchronisation où le réseau doit apprendre à mesurer des intervalles précis entre les événements. Concrètement, ces connexions impliquent que les portes d'entrée et d'oubli prennent l'état de la cellule à l'instant t-1 en entrée supplémentaire. La porte de sortie ne prend elle que l'état de la cellule à l'instant t puisque la nouvelle valeur de l'état de la cellule a alors déjà été calculée. Ces informations supplémentaires induisent une amélioration de la capacité des LSTM à apprendre des tâches qui nécessitent un timing précis et une énumération des états internes [Lipton *et al.*, 2015; Greff *et al.*, 2017].

•Gated Recurent Unit, GRU

Cho *et al.* [2014] ont proposé une simplification de l'architecture LSTM sous le nom de Gated Recurrent Unit (GRU). Ils n'utilisent ni connexions peepholes ni fonctions d'activation de sortie, mais ont choisi de fusionner les portes d'entrée et d'oubli en une seule porte, dite de mise à jour. Enfin, ils ont modifié la fonction de la porte de sortie (renommée aussi porte de réinitialisation) afin qu'elle ne soit plus en charge que de la transmission des connexions récurrentes à l'entrée de bloc. Si les premiers résultats de comparaison entre GRU et LSTM ont amené des résultats mitigés, depuis de nombreuses recherches se sont focalisées sur cette variante des LSTM et l'amélioration de ses performances [Chung *et al.*, 2014].

D'autres implémentations ont été proposées (comme le full gradient de Graves et Schmidhuber [2005]) et testées pour différentes tâches (reconnaissance de l'écriture manuscrite, modélisation acoustique) et différents corpus de

données (corpus de parole TIMIT, base de données en ligne d'écriture manuscrite IAM ou encore un jeu de données de modélisation musicale polyphonique nommé JSB Chorales). Cependant, il en ressort que malgré les très bonnes performances des différents modèles, il n'y a pas de modèle dominant : chacun est performant pour une tâche donnée ou pour traiter un corpus donné [Greff *et al.*, 2017]. Ces résultats nous ont donc convaincus d'étudier et d'incorporer les LSTM standards dans nos travaux, plutôt qu'une de ses variations. La reproduction des expériences sur des variations pourra être envisagée en perspective de ce travail de thèse.

Apport des LSTM dans l'apprentissage séquentiel sur le long terme

Tous les RNNs ont des boucles de rétroaction au niveau de la couche récurrente : c'est ce qui leur permet de maintenir l'information en mémoire sur plusieurs pas de temps. Le problème, comme nous l'avons vu précédemment, est que selon la longueur des séquences, la capacité des RNNs à conserver le passé lointain est altéré puisque celui-ci est progressivement écrasé par un passé plus récent (expliqué en section 2.3.1). Face à cette problématique les LSTM semblent apporter une solution efficace : grâce au système de mémoire intermédiaire dû au mécanisme de portes d'une unité LSTM, l'information importante est sauvegardée, mais surtout préservée, jusqu'à son utilisation. Elle n'est ni impactée par le bruit, ni oubliée ou écrasée. Cela permet donc que les dépendances séquentielles soient prises en charges par les LSTM et qu'elles ne constituent plus un obstacle à leur utilisation. Gers *et al.* [1999] a notamment montré que les LSTM étaient performants pour la prédiction des éléments constituant des séquences de longueur finie de tailles variables, mais aussi de flux continus. Gers et Schmidhuber [2001] ont aussi montré la supériorité des LSTM (leur modèle comportait des connexions peepholes) pour l'étude et l'apprentissage de langages non-contextuels et contextuels. Ils ont d'ailleurs été les premiers à utiliser avec succès un RNN-LSTM pour l'apprentissage de langage contextuel. Au vu de ces résultats et de l'ensemble des détails techniques fournis dans cette partie, les LSTM semblent donc un dispositif de traitement de séquence adaptatif très général et prometteur. Nous verrons dans le chapitre 3 notre implémentation et les travaux menés sur ce modèle.

Apport des LSTM dans l'évanouissement du gradient

Au delà de pallier la limite du SRN, les LSTM permettent de résoudre le problème de l'évanouissement du gradient dont souffre la plupart des RNN utilisant l'algorithme d'apprentissage BPTT (expliqué en section 2.3.1). Dans un réseau multi-couches, les gradients pour les couches les plus profondes sont calculés en tant que produits de nombreux gradients (des fonctions d'activation). Lorsque ces gradients sont petits ou nuls, ils disparaissent et lorsqu'il sont supérieur à 1, ils tendent à exploser. Il devient donc très difficile de les

calculer. Dans la récurrence du LSTM au niveau du CEC, la fonction d'activation est la fonction d'identité avec une dérivée de 1.0. Ainsi, le gradient rétropropagé ne s'annule pas et n'explose pas, mais reste constant. Lorsque la porte d'oubli est activée (activation proche de 1.0), le gradient ne disparait pas. Puisque l'activation de la porte d'oubli ne peut être supérieur à 1.0, le gradient ne peut pas exploser non plus. Cette récurrence, alliée au système de portes, explique pourquoi les LSTM sont adaptés et efficaces pour apprendre les dépendances à long terme, et cela indépendamment de la profondeur du réseau ou la longueur de la séquence d'entrée.

2.4 Extraction des règles et de la grammaire à partir d'un RNN

Il existe deux types d'algorithmes dans le domaine d'extraction des connaissances : ceux qui s'appliquent sur des modèles généraux de réseaux de neurones et ceux qui raffinent des connaissances déjà connues [Remm, 1996]. Dans nos travaux, nous nous intéressons surtout à la première catégorie, et plus précisément aux réseaux de neurones récurrents, puisque nous souhaitons accéder à la mémoire implicite de ces réseaux et mettre en évidence les représentations implicite qu'ils encodent. Pour plus de détails sur les différents algorithmes d'extraction des connaissances à partir de différents modèles, tels les MLP, le lecteur pourra se référer à Remm [1996].

L'extraction des règles à partir d'un RNN se définit comme le processus consistant à rechercher, trouver et construire les grammaires encodées par les RNN après un apprentissage séquentiel. Autrement dit, cela revient à extraire un automate à état finis A tel que les séquences du langage L (généré par une grammaire) acceptées par A soient équivalentes à celles acceptées par le RNN [Jacobsson, 2005; Weiss *et al.*, 2017]. Le but d'un tel processus vise, entre autres, à éclaircir ce qui se passe dans un RNN lors de l'apprentissage et résoudre la question des RNN "boites noires" [Omlin et Giles, 2000] en mettant en évidence les règles encodées. De nombreux travaux ont été menés dans ce sens pour extraire ces règles sous la forme d'un automate [Lawrence *et al.*, 2000; Omlin et Giles, 2000; Smith, 2003; Jacobsson, 2005; Zeng *et al.*, 1993; Bhargava et Purohit, 2011; Karpathy *et al.*, 2015; Strobelt *et al.*, 2016; Wang *et al.*, 2017; Murdoch et Szlam, 2017]. Nous allons aborder et décrire dans cette section les étapes générales du procédé d'extraction des règles. Pour cela nous utiliserons les concepts définis dans la section 2.1. Nous finirons cette section en discutant les paramètres critiques qui peuvent impacter les résultats recherchés. La figure 2.1 présente une vision globale du processus d'extraction des règles à partir d'un RNN.

2.4.1 Les cinq étapes de l'extraction des règles à partir d'un réseau de neurones récurrents

Parmi les références les plus complètes portant sur la question de l'extraction des règles, nous soulignerons celle de Jacobsson [2005] et plus récemment celle de Weiss *et al.* [2017]. La première offre une revue de l'ensemble des méthodes d'extraction des règles à partir des RNN et propose une taxonomie de ces différentes méthodes. Le point majeur de cet article est que Jacobson réussit à déterminer ce qu'il désigne comme "les quatre ingrédients clés" d'un processus d'extraction. La seconde référence quant à elle, offre une revue plus récente, mise à jour et surtout montre le regain d'intérêt de la communauté scientifique pour ce domaine. Nous avons choisi de présenter ici cinq ingrédients et non pas quatre. Pour plus de clarté, nous ajoutons la génération des patterns d'activations de la couche cachée en tant qu'ingrédient 0 afin d'inclure l'ensemble des éléments. Nous détaillerons dans ce qui suit le processus d'extraction global à partir d'un RNN en commentant les différentes techniques existantes pour chaque ingrédient.

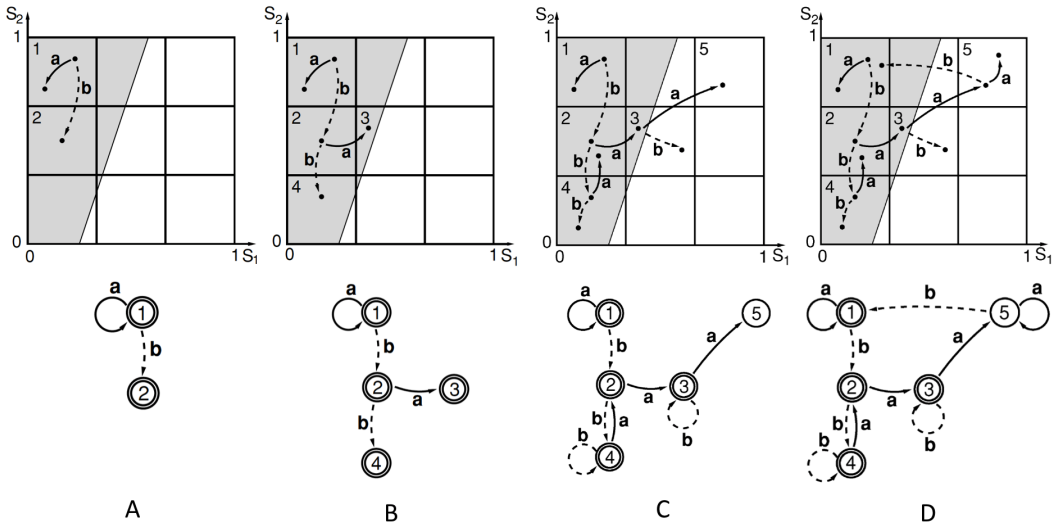


FIGURE 2.17 – Itérations d'un algorithme d'extraction d'automate avec un paramètre de quantification q = 3 utilisé sur un RNN avec deux neurones, entrainé sur un langage binaire. L'espace d'état est divisé en une région d'acceptation et une de rejet (respectivement en gris et en blanc). L'algorithme développe le graphe jusqu'à ce que tous les nœuds aient deux arcs sortants. A et B représentent les deux premières itérations, C et D représentent les 2 dernières. Exemples de Giles *et al.* [1991] extrait de Jacobsson [2005]

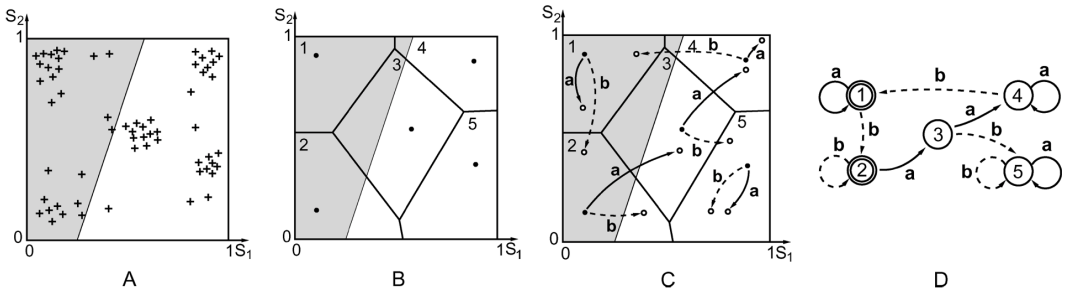


FIGURE 2.18 – Exemple d'extraction de règles à travers une recherche en largeur dans un espace d'états groupés par k-means. A - Les états du RNN sont échantillonnés pendant l'entraînement. B - Ces états sont regroupés en un nombre prédéfini de clusters. C - Une recherche en largeur est effectuée (voir la figure 2.17). D - L'automate est construit. Images extraites de Jacobsson [2005]

Ingrédient 0 : Les patterns d'activations de la couche cachée du RNN

Bien qu'il ne soit pas cité par Jacobsson [2005], il nous a semblé important d'expliciter quelles sont les données nécessaires pour le processus d'extraction sous la bannière "Ingrédient 0" pour la bonne compréhension des explications qui suivent. Ainsi nous considérons que l'élément initial incontournable d'un processus d'extraction de règles est le RNN à partir duquel l'extraction est réalisé. Certains auteurs affirment que le processus d'extraction peut aussi avoir lieu durant l'apprentissage. Pour nos travaux, nous avons choisi de nous focaliser sur le processus d'extraction une fois l'apprentissage réalisé et la représentation interne stabilisée, car celle ci évolue durant l'apprentissage [Servan-Schreiber *et al.*, 1988]. Une fois l'apprentissage finalisé, il faut réaliser la phase de test et enregistrer à chaque pas de temps, le pattern d'activation de la couche cachée du RNN. En effet, c'est l'ensemble des patterns d'activation de la couche cachée à chaque pas de temps qui contient la représentation implicite des règles que nous souhaitons extraire. Ce sont ces patterns d'activités qui serviront de données à la suite du processus d'analyse.

Ingrédient 1 : La quantification de l'espace d'états continu

Chaque pattern de la couche cachée d'un RNN peut être défini comme une représentation interne i du réseau à un instant t, en fonction de l'entrée reçue par le réseau à l'instant t, mais aussi de celles traitées avant. Cette représentation "se retrouve localisée" quelque part dans l'espace d'états du RNN. Cet espace est considéré continu car les patterns de la couche cachée et de la couche de sortie sont composés de valeurs réelles qui appartiennent à des intervalles continus. Ainsi l'exploration d'un espace d'état d'un RNN peut théoriquement et très probablement être de durée infinie. Pour solutionner cela, la première

étape du processus consiste en la quantification de cet espace d'états en un ensemble d'états discrets. Le but est de cartographier l'ensemble des états discrets correspondant aux états de l'automate que nous souhaitons extraire.

Depuis les années 90, de nombreuses stratégies ont été proposées pour cela : Omlin et Giles [1996b] ont proposé l'équipartitionnement. Cette technique consiste à séparer chaque coordonnée de l'espace de sortie en intervalles égaux nommés hypercubes et de traiter chacun d'entre eux comme un état. Le paramètre central est alors le degré de quantification q. Les auteurs ont commencé avec q=2 et ont ensuite incrémenté ce paramètre, jusqu'a obtenir un automate qui reconnaisse les séquences du corpus de test. La figure 2.17, label A, présente un exemple d'équipartionnement avec un paramètre q=3 sur un RNN de deux neurones entrainé sur un langage binaire. L'espace d'état de chaque neurone est divisé en trois, ce qui forme neuf partitions égales, soit les hypercubes. D'autres méthodes pour déterminer ce paramètre q ont été explorées [Giles *et al.*, 1991, 1992a; Omlin et Giles, 1996a], mais ont toutes présenté des limites. En appliquant une quantification uniforme sur l'ensemble de l'espace de sortie, elle aboutit à une explosion du nombre des états (même lorsque cela porte sur un petit réseau), ce qui rend l'exploration fastidieuse et longue.

Une autre approche de quantification de l'espace d'états se base sur des approches de clustering (regroupement) telles que la quantification vectorielle avec l'algorithme des k-means [Zeng *et al.*, 1993], les cartes auto-organisatrices [Blanco *et al.*, 2000] ou encore le clustering hiérarchique [Cleeremans *et al.*, 1989; Servan-Schreiber *et al.*, 1989; Elman, 1990]. Dans ces approches, il est considéré que chaque élément d'une séquence traitée par le réseau appartient à un état via le pattern d'activité généré par le réseau à ce moment là. Ce sont ces patterns qui sont soumis aux techniques de clustering afin de dégager des clusters. Ce partitionnement de l'espace d'états défini par les clusters est ensuite exploré de la même manière que celui décrit dans Omlin et Giles [1996b]. Sa taille, beaucoup plus petite, le rend alors plus facilement exploitable. La figure 2.18 , label B, présente un espace d'état ayant été quantifié par la technique de clustering du k-means. Pour des revues sur les techniques d'extraction de règles à partir de RNN, le lecteur se référera à Wang *et al.* [2017] et Jacobsson [2005].

Ingrédient 2 : La génération des états et des sorties

Une fois le partitionnement réalisé, le seconde étape consiste en la génération des états et des sorties (et de la classification de sortie si nécessaire) en donnant les patterns d'entrées au RNN. En reprenant le corpus de test, (le même que celui utilisé pour la quantification) et en le fournissant en entrée

au RNN, on obtient alors les mêmes patterns que précédemment. Dans le cas où des techniques de clustering ont été utilisées pour la quantification, chaque pattern h_i sera alors associé à une partition que nous avons préalablement identifié. Dans le cas où ce sont des techniques d'équipartionnement selon Omlin et Giles [1996b] qui ont été utilisées, c'est une recherche en largeur sur les états qui est effectuée. Chaque pattern pointant une partition, en alimentant le réseau avec des patterns d'entrée, il est possible de déterminer quelles sont les partitions qui correspondent aux données. Ce processus est réitéré jusqu'a ce qu'aucune nouvelle partition ne soit visitée. Il est alors possible de regrouper les patterns de la couche cachée selon la partition (l'état) auxquels ils sont liés, mais aussi de trier ces états et ne garder que ceux qui correspondent aux données. C'est la phase d'élagage.

Ingrédient 3 : La construction de l'automate

La construction de l'automate à états finis peut avoir lieu en même temps que l'étape précédente, mais pour plus de clarté, ces deux étapes ont été séparées conceptuellement. Le processus est le suivant : Lorsque les entrées (du corpus de test) sont soumises au RNN, à chaque pas de temps, le pattern d'activité de la couche cachée va indiquer un état donné. Au premier pas de temps t_0, un premier pattern d'activité de la couche caché h_0, désignera un état (figure 2.17, label A). Ce dernier devient le nœud de départ du graphe représentant l'automate : ce nœud initial représente alors l'état initial q_0. Le second pattern d'activité au pas de temps suivant t_1 va fournir le second état qui sera alors représenté par un second nœud dans le graphe-automate. La transition entre les deux clusters (et donc les deux états) sera alors représentée par un arc dans le graphe-automate et ainsi de suite, jusqu'a ce que tous les patterns de la couche cachée ait été traités. Si un pattern d'activité h_i, correspond à un état déjà visité et dont le nœud existe déjà dans le graphe-automate, alors seul un arc (une transition) vers ce nœud est ajouté à l'automate (figure 2.17, label C, cas des transitions entre les nœuds 4 et 2). Si deux patterns h_i et h_j correspondent aux mêmes clusters, et donc au même nœud, un arc récurrent sur ce nœud est alors ajouté au graphe-automate. Cet état possèdera alors une connexion récurrente sur lui même (figure 2.17, label C, cas des nœuds 3 et 5).

Ingrédient 4 : La minimisation de l'automate

Selon Jacobsson [2005], ce dernier ingrédient ne fait pas réellement partie du procédé d'extraction des règles à partir des RNN. Cependant, il insiste pour le présenter car de nombreux auteurs utilisent l'algorithme de minimisation standard Hopcroft *et al.* [1979] afin de réduire l'automate obtenu à l'étape précédente en le plus petit automate possible. En effet, selon la grammaire encodée dans les séquences, et la technique de partitionnement, l'automate extrait peut

s'avérer très dense et complexe à interpréter. Pour faciliter cette étape, certains ont donc recours à l'algorithme de minimisation standard qui permet ainsi de réduire la complexité de l'automate-graphe en terme de nombres de nœuds et de transitions. Une fois l'automate obtenu, il n'est pas garanti qu'il soit correct. Il faut donc le tester pour le valider. Cette phase de test consiste à utiliser à nouveau les séquences du corpus de test sur l'automate résultant du processus d'extraction afin de vérifier sa validité.

2.4.2 Les étapes critiques du procédé

Dans l'extraction de règles à partir d'un RNN, l'espace d'état continu est approximé par un ensemble fini d'états, et la dynamique du RNN est représentée par les transitions entre cet ensemble discret d'états. Cependant, il existe deux étapes critiques lors de ce processus d'extraction : La première concerne l'étape de quantification et la seconde l'étape de construction de l'automate ou plutôt sa validation. Selon la stratégie de quantification de l'espace d'état d'un RNN qui a été choisie, le nombre de partition peut varier. Or c'est ce dernier qui détermine le nombre d'états (de nœuds) de l'automate recherché. Il n'existe pas de solutions préconisées concernant le choix de la méthode pour l'étape de quantification. De nombreux auteurs proposent des approches différentes (quantification, clustering, approche stochastique, etc.), chacune dotée de ses limites, afin de tenter de trouver l'automate qui approxime le mieux la grammaire recherchée. Parmi ces méthodes, il faut souligner les approches de clustering. Il s'agit d'un domaine de recherche en soi dans le machine learning où de nombreux travaux sont en cours et voient régulièrement le jour. Par ailleurs, l'un des problèmes essentiels en utilisant le clustering est qu'il n'existe pas de manière fiable de relier les clusters entre eux selon un ordre temporel. En effet, si l'on se réfère à Elman, il a été prouvé et nous le verrons aussi par la suite, que la relation temporelle entre les états finit par être perdue à un moment, surtout dans le cas où les séquences d'entrées sont porteuses de dépendances sur le long terme. Il devient donc difficile de relier de manière fiable les clusters entre eux selon un ordre temporel précis.

2.4.3 Grammaire connue versus grammaire inconnue

Dans l'ensemble des travaux cités et décrits ci-dessus, il est souvent fait référence à l'extraction d'un automate que l'on compare à une grammaire connue. Celle-ci, ayant servi à générer les séquences utilisées pour les phases d'apprentissage et de tests, se retrouve encodée dans l'espace d'états du RNN utilisé. Si en effet, de nombreux travaux sont réalisés dans des domaines où les grammaires sont connues, il en existe d'autres dans le but de mettre en évidence des grammaires inconnues [Giles *et al.*, 1992b]. Le procédé d'extraction reste alors le même et celui de validation de l'automate aussi. La différence majeure

réside dans le fait qu'il n'est pas possible de comparer le ou les automates obtenus à une grammaire afin de déterminer le bon nombre de clusters par exemple ou de partitions.

2.5 Conclusion

Dans ce chapitre nous avons souhaité présenter l'ensemble de connaissances techniques nécessaires à la compréhension de nos travaux. Nous avons ainsi défini et décrit ce que sont des réseaux de neurones récurrents, ainsi que la problématique à laquelle ils semblent répondre. Ensuite nous avons décrit plus en détails deux réseaux en particulier. Le SRN est l'un des plus utilisés dans le domaine de l'apprentissage implicite et a fait l'objet de nombreux travaux sur l'extraction des règles. Les LSTM sont un modèle qui bien qu'ancien, connait un véritable regain d'intérêt ces dernières années de la part de la communauté scientifique grâce à leurs nombreuses caractéristiques qui semblent pallier les défauts des précédents modèles. Nous avons ensuite détaillé et décrit le processus d'extraction des règles à partir des réseaux de neurones récurrents, en soulignant les étapes critiques, afin de rappeler qu'il existe dans ce domaine encore de nombreuses questions à résoudre. En effet, que ce soit au niveau des RNNs, ou des procédés d'extraction des règles, de nombreux travaux sont encore en cours (notamment dans le traitement du langage naturel) et la question de l'adéquation "solution choisie - problématique considérée" reste donc une question ouverte à discussion.

Dans le chapitre suivant nous allons présenter les travaux réalisés dans le cadre de cette thèse : nous détaillerons nos implémentations d'un SRN et d'un RNN-LSTM. Nous présenterons notamment, les travaux d'extraction des connaissances que nous avons mis en place à partir des LSTM.

Chapitre 3

Implémentation des RNNs et extraction de règles

Table des matières

3.1 Introduction

Nous avons présenté dans le chapitre 1 les données porteuses d'informations que nous avons mises en évidence via un travail de préparation et d'extraction. Nous avons ainsi mis en avant l'importance des séquences de composants électriques dans les schémas d'une entreprise cliente, car ce sont elles qui portent la connaissance métier d'une entreprise : l'empreinte en matière de conception et les règles de connectivité électrique. Dans nos travaux, nous avons présenté une

séquence comme une suite de composants électriques ordonnés selon leurs coordonnées (x,y) extraites des folios. On pouvait ainsi caractériser chaque folio par une seule séquence, et étudier ensuite les similitudes entre différents folios de différents schémas électriques. En réalité, les séquences lues et extraites par les experts d'Algo'Tech sont plus complexes. Elles correspondent toujours à une suite de composants électriques, mais ces derniers sont sur le même câble. En effet, à la lecture des schémas, les experts d'Algo'Tech parcourent les folios d'un même schéma un à un, en suivant les fils électriques. Or si ces derniers commencent tous (schématiquement) depuis le même folio, ils ne font pas tous le même chemin physique (dans la réalité) et ne sont pas destinés à la même fonction. Cela implique donc qu'il existe plusieurs lectures à réaliser au sein d'un même schéma électrique et qu'un folio peut être parcouru plusieurs fois, selon le câble étudié.

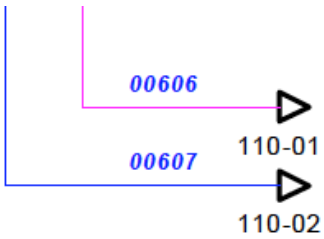


FIGURE 3.1 – Deux renvois folios de 2 fils différents mais allant à un même folio

L'importance de ces séquences réside dans le fait qu'elles sont porteuses de dépendances séquentielles implicites sur le long terme : en effet, le choix des équipements à la fin du câble dépend de l'ensemble des choix réalisés en amont et cela dès le premier folio. Ainsi, notre but est de trouver une approche pour mettre en évidence à la fois les connaissances implicites (les règles de connectivité électrique entre les composants) mais aussi les dépendances séquentielles. La particularité de notre domaine de travail est qu'il n'existe pas de grammaire explicite pour chaque entreprise-cliente, présentant les règles et habitudes de travail de chacun. Vouloir extraire ces règles revient donc à reconstruire une grammaire que nous n'avons pas et que nous ne pouvons valider sans l'échange avec un expert du domaine concerné, qu'il soit d'Algo'Tech ou de l'entreprise-cliente. Pour cela il aurait fallu réaliser la vérification des différentes règles extraites (à chaque changement de paramètrage pour chacun de nos RNNs) en comparant celles-ci aux schémas électriques et aux bases de données de chaque client, ce qui engendre un travail volumineux trop chronophage pour solliciter des experts d'Algo'Tech dessus.

Au vu des nombreux paramètres et contraintes que l'exploitation des données électriques imposait, nous avons choisi dans un premier temps, de tra-

vailler sur des séquences, non seulement similaires en termes de caractéristiques, que nous détaillerons ensuite, mais aussi issues d'une grammaire connue. Nous souhaitions ainsi fournir une preuve de concept valide dans un environnement dont les paramètres sont maitrisés. En nous inspirant de la littérature [Reber, 1967; Cleeremans, 1993], nous avons donc choisi pour nos travaux d'utiliser comme base de travail, la grammaire de Reber, qui se rapproche le plus du domaine électrique. En effet, cette grammaire permet de générer des séquences qui ont les mêmes caractéristiques que les séquences électriques :

- elles sont de tailles variables
- elles sont porteuses de contexte et donc de dépendances : le choix des derniers éléments se fait en fonction des premiers
- l'occurrence de chaque élément dans chaque séquence est variable : une occurrence ou plusieurs, une fois ou plusieurs fois dans une même séquence.
- Les règles de connectivité entre les éléments d'une même séquence sont fixes, ce qui rejoint le coté normé, codifié, réglementé du domaine électrique.

Dans un second temps, une fois notre approche validée sur des grammaires artificielles, nous la testerons sur des séquences électriques. Bien que le travail d'extraction d'une grammaire électrique et de sa validation soit conséquent, nous avons tenu à tester notre approche dans un cas réel afin de la valider sur une problématique industrielle concrète.

Pour aborder ce chapitre, nous allons dans un premier temps préciser les différentes grammaires utilisées ainsi que leurs caractéristiques. Ensuite nous présenterons nos résultats selon deux axes : le premier axe proposera un comparatif des performances du SRN et d'un RNN-LSTM lors de l'apprentissage des séquences issues de la grammaire de Reber et de ses variantes. Le second axe, quand à lui, décrira l'approche nouvelle que nous avons mis en place pour l'extraction des règles encodées par un RNN-LSTM.

3.2 Les grammaires utilisées

Quatre grammaires ont été utilisées lors de nos travaux : une grammaire manuelle créée par nos soins, la grammaire de Reber telle qu'il l'a proposé en 1967 [Reber, 1967], ainsi que deux de ses variantes. Chacune a permis de tester un aspect particulier des réseaux de neurones : telles que les performances lors de la prédiction, la résistance à la taille des séquences ou encore la bonne rétention des informations lors de dépendances sur le long terme entre les éléments. Les figures 3.2 et 3.4 présentent les quatre grammaires et la table 3.1 décrit leurs paramètres.

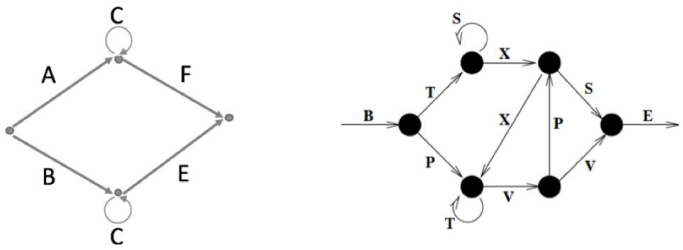


FIGURE 3.2 – Grammaire manuelle à gauche et Grammaire de Reber à droite (Image extraite de Servan-Schreiber *et al.* [1988])

Grammaire	Nombre de symboles	Nombre de nœuds	Nombre d'arcs
Manuelle	5	4	6
Reber	7	6	10
Reber embarquée	7	14	24
Reber embarquée et continue	7	14	26

TABLE 3.1 – Descriptif des différentes grammaires

3.2.1 La grammaire manuelle

Le but de la grammaire manuelle est de permettre l'étude des performances des réseaux dans leur capacité à retenir des dépendances sur le long terme. Elle permet en effet de générer deux types de séquences dont les débuts et fins sont différents :

— Le type 1 est ACF
— Le type 2 est BCE

La différence entre des séquences d'un même type réside dans le nombre de C intermédiaire entre les symboles de début et de fin de séquence. Après apprentissage de ces séquences, lors de la phase de test, en étudiant les patterns d'activations de la couche de sortie à la fin de chaque séquence (au moment de prédire le dernier symbole), il est possible d'étudier la capacité du réseau à conserver une information sur le long terme : En d'autres termes, nous testons la résistance du réseau de neurones à prédire F ou E à chaque fois que la séquence commence par, respectivement A ou B, et cela en faisant varier le nombre de C intermédiaire. Le symbole D ne fait pas partie de la grammaire manuelle, mais est utilisé dans les tests, notamment comme une mesure de contrôle. Ne faisant pas partie du séquences générées par cette grammaire, il ne devrait pas être prédit par le réseau.

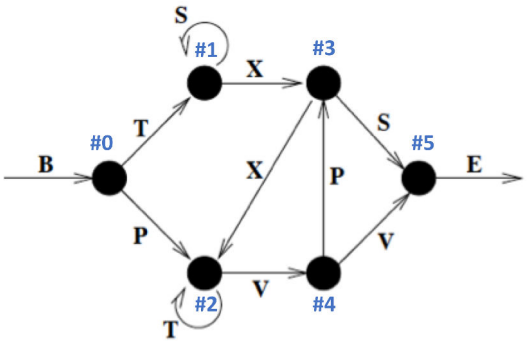


FIGURE 3.3 – Grammaire de Reber avec nœuds numérotés. Image extraite de Servan-Schreiber *et al.* [1988]

3.2.2 La grammaire de REBER (RG)

La grammaire à états finis de Reber (RG pour Reber's Grammar), présentée dans la figure 3.3, a longuement été étudiée dans la littérature notamment dans le cadre d'expériences d'apprentissage de grammaire artificielle pour étudier les mécanismes d'apprentissage implicite [Cleeremans et McClelland, 1991; Cleeremans *et al.*, 1989]. Bien que nous l'ayons présenté précédemment, nous allons maintenant détailler cette grammaire et présenter son intérêt pour nos travaux. Cette grammaire à états finis est dans les faits un graphe orienté, se composant de nœuds et d'arc de transitions. Ces derniers possèdent à la fois un sens ainsi qu'une étiquette. Sur la figure 3.3 l'arc qui va du nœud #0 vers le nœud #1 a pour étiquette le symbole T. Dans les expériences menées, les séquences générées possèdent les particularités suivantes :

— elles sont de tailles variables
— elles commencent toutes par le symbole B pour "Begin", qui désigne donc le début d'une séquence
— elles se terminent toutes par le symbole E pour "End", qui désigne donc la fin d'une séquence

Algorithme de génération des séquences

Les séquences sont générées comme suit : chaque séquence (vide au début) est initialisée avec un B, et le nœud #0 est désigné comme nœud initial. Les arcs dirigés (avec un sens) ayant tous une probabilité de 0.5 d'être choisis, c'est de manière aléatoire qu'est sélectionné le nœud suivant parmi les successeurs du nœud courant. Une fois le nœud sélectionné, l'étiquette de l'arc choisi est ajoutée à la séquence. A ce stade, le résultat est soit BT ou BP. Une fois le nœud suivant sélectionné, le processus de sélection aléatoire est répété parmi les successeurs du nœud courant, et ainsi de suite jusqu'à ce que le nœud #5 soit atteint. Lorsque cela arrive, la séquence est considérée finie puisque le nœud #5 ne possède pas d'arcs vers d'autres nœuds. Le symbole E est alors

ajoutée à la séquence, la concluant ainsi. La table 3.2 fournit un exemple de séquences générées à partir de la grammaire.

Numéro séquence	séquence
1	BTXSE
2	BPVVE
3	BTSXXVPSE
4	BPTTTTVPSE
5	BTSSSSSSSSSSXSE

TABLE 3.2 – Exemple de séquences générées à partir de la grammaire de Reber

L'importance du contexte dans la grammaire de Reber pour la modélisation des séquences de composants électriques

La particularité de la grammaire de Reber est de proposer, à l'exception des symboles de début et de fin de séquence, chaque symbole deux fois, mais pas dans le même contexte. En effet, si l'on observe, par exemple le symbole T, celui-ci apparait, à la fois après B et après P. Sauf que dans le premier cas, il mène à S ou X et dans le second cas, il mène à lui-même (il s'agit d'une boucle) ou V. Cette différence de symbole en amont et symbole en aval, peut être assimilée à une différence de contexte. Or dans le contexte électrique, chaque composant électrique possède ses propres contraintes dues à son ampérage, le type d'alimentation et tout un ensemble de caractéristiques techniques électriques. Une entreprise-cliente prendra donc l'habitude de connecter chacun des composants à d'autres qui lui correspondent (par exemple même ampérage) selon le contexte environnemental à l'instant t. Un même composant peut donc se retrouver dans deux séquences de composants électriques différentes mais correctes électriquement car elles répondent à deux contextes différents. Il est donc important d'être capable d'étudier dans notre approche des séquences où l'on retrouve les mêmes éléments (symboles pour la grammaire ou composants électriques pour le domaine électrique) mais dans des contextes différents. Notre choix de la grammaire de Reber se trouve donc justifié par l'ensemble des caractéristiques que partagent les séquences qu'elle permet de générer avec les séquences du domaine électrique. Par ailleurs, le nombre de séquences et de nœuds étant petit, cela permet de lancer des simulations de calculs à faible coût, en terme de temps et de puissance de calcul.

3.2.3 Les variantes de la grammaire de REBER

Afin d'éprouver les performances des réseaux de neurones récurrents, nous avons choisi d'exploiter deux variantes de la grammaire de Reber que nous

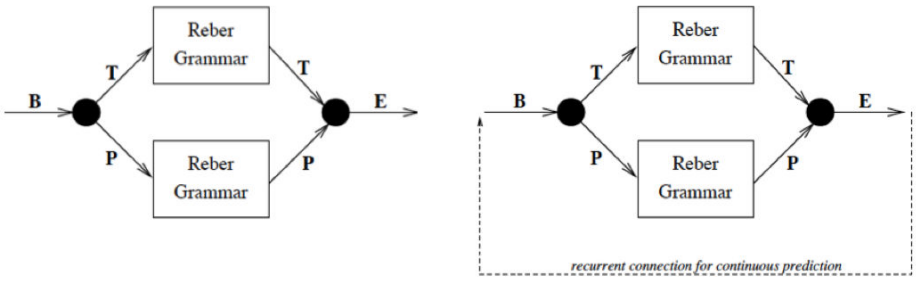


FIGURE 3.4 – Grammaire de Reber embarquée à gauche et Grammaire de Reber embarquée et continue à droite. Images extraites de Servan-Schreiber *et al.* [1988]

présentons ci dessous et dans la figure 3.4.

La grammaire de Reber embarquée (ERG)

La grammaire de Reber embarquée (ERG) a pour but, tout comme la grammaire manuelle, de tester les performances du réseau dans le fait de retenir des dépendances sur le long terme. La différence réside dans une grammaire beaucoup plus complexe : celle-ci ajoute de l'ambiguïté au niveau des symboles de début et de fin de séquences (ils se retrouvent au sein des séquences aussi et non plus exclusivement aux extrémités) et fournit des séquences plus longues.

La grammaire de Reber embarquée et continue (CERG)

La grammaire de Reber embarquée et continue (CERG) est une variante très complexe de la grammaire d'origine car elle supprime la notion de début et de fin de séquence. Le réseau de neurones doit apprendre un flux continu de séquences ERG mises bout à bout.

3.3 Comparaison du SRN et RNN-LSTM

Dans cette section, nous allons aborder les différences de performances avec le SRN et un RNN-LSTM, modèle doté d'unités LSTM. Les deux modèles possèdent une architecture commune avec néanmoins une variation. Tout deux se compose d'une couche d'entrée, d'une couche cachée et d'une couche de sortie. Les variations résident d'une part dans la nature des cellules de la couche cachée (dans SRN, les cellules sont des neurones artificiels et dans RNN-LSTM, ce sont des unités LSTM qui composent la couche cachée) et dans la constitution de la couche d'entrée (dans SRN, la couche d'entrée comporte aussi des unités de contexte). Pour chacune de nos implémentations, nous nous sommes

inspirés de la littérature existante pour guider nos choix concernant l'architecture de chaque modèle et les paramètres de tests. Pour le SRN, l'article de référence est Servan-Schreiber *et al.* [1988], et pour le RNN-LSTM, l'article de référence est Gers *et al.* [1999]. Nous allons ainsi dans un premier temps, décrire deux variantes du SRN d'un point de vue technique, et les tests menés, ensuite nous aborderons le modèle RNN-LSTM, ses performances et caractéristiques avant de réaliser un comparatif entre les deux modèles. La table 3.3 présente les différents modèles implémentés et leurs caractéristiques.

Pour chaque test, à chaque fois que nous générons un corpus de séquences, nous donnerons la taille de celui-ci, la longueur moyenne et la déviation standard des séquences composant le corpus. Nous fournirons aussi le nombre d'échantillons moyen générés par ce corpus et analysé par le réseau. Pour rappel, les échantillons (couples de symboles entrée-sortie) proviennent des séquences générées. Chaque séquence de taille N engendre N-1 échantillons.

Paramètres	SRN 1	SRN 2	RNN-LSTM
Nombre unités couche entrée	7	7	7
Nombre unités couche sortie	7	7	7
Nombre de couches cachées	1	1	1
Nombre unités couche cachée	3	15	8 cellules (4 blocs de 2 cellules chacun)
Nombre unités couche contexte	3	15	
Nombre de poids modifiés durant apprentissage	51	435	424

TABLE 3.3 – Description des différentes architectures des réseaux de neurones récurrents testés

3.3.1 SRN

Protocole

Servan-Schreiber *et al.* [1988] ont été parmi les premiers à se pencher sur la problématique de l'encodage des règles de grammaire au sein d'un SRN. Nous avons donc choisi de construire notre modèle de SRN selon les paramètres fournis dans leur article. Le nombre d'unité de la couche cachée a été fixé à 3 unités puis 15 unités, ce qui portait le nombre d'unités de la couche d'entrée

à $7 + 3 = 10$ unités puis $7 + 15 = 22$ unités. 7 désignant le nombre d'unités en entrée (et en sortie) nécessaire pour encoder les symboles de la grammaire de Reber. Dans ces deux cas, le nombre de poids paramétrés $nb_{W_Rnn-Elman}$ se calcule comme ce qui suit :

$$nb_{W_{Rnn-Elman}} = (nb_{entree} + nb_{contexte}) * nb_{cachee} + nb_{cache} * nb_{sortie} \tag{3.1}$$

avec : nb_{entree}, le nombre d'unités de la couche d'entrée, $nb_{contexte}$, celui de la couche de contexte, nb_{cachee} celui de la couche cachée et nb_{sortie} celui de la couche de sortie.

Ainsi pour le SRN comportant 3 unités cachées, le nombre de poids est de :

$$nb_{W_{Rnn-Elman}} = (7 + 3) * 3 + 3 * 7 = 51 \tag{3.2}$$

Et pour 15 unités cachées :

$$nb_{W_{Rnn-Elman}} = (7 + 15) * 15 + 15 * 7 = 435 \tag{3.3}$$

Pour le choix des paramètres, nous avons utilisé la méthode de *Grid Search* appliquée aux paramètres recommandés par la littérature. Le taux d'apprentissage a été fixé à [0.01, 0.015, 0.02] , le momentum a pris les valeurs suivantes : [0.01, 0.1, 0.3, 0.5, 0.7, 0.9]. Pour la fonction d'activation, c'est la fonction sigmoïde qui a été appliquée. La table 3.4 récapitule les paramètres du SRN utilisés ainsi que les tailles des corpus de séquences. La table 3.6 présente les 36 configurations d'apprentissages utilisée pour SRN. Nous avons regroupé les configurations trois par trois sous le nom de groupe afin de faciliter la discussion des résultats ci dessous.

Nombre d'unités de la couche cachée	3, 15
Taux d'apprentissage	0.01, 0.015, 0.02
Momentum	0.01, 0.1, 0.3, 0.5, 0.7, 0.9
Taille du corpus d'apprentissage	200 000
Taille du corpus de test généré à partir de RG	20 000
Taille du corpus de test généré aléatoirement	130 000

TABLE 3.4 – Descriptif des différents paramètres utilisés lors de l'implémentation et du test de SRN

Pour chacune des 36 configurations, un corpus d'apprentissage composé de 200 000 séquences (générées à partir de la grammaire de Reber) a été créé, en accord avec l'expérience de Servan-Schreiber *et al.* [1988]. Sur les 36 corpus, la longueur moyenne des séquences est 8.00 et la déviation standard moyenne

est 3.37. Le nombre moyen d'échantillons est de 1 402 360.

Pour tester les performances du réseau pour chaque configuration, après apprentissage, nous avons réalisé 10 simulations. Pour chacune d'entre elles, nous avons généré deux corpus de tests : un corpus composé de séquences grammaticales et un corpus composé de séquences aléatoires. Pour ces deux types de séquences, les probabilités de choix des symboles étaient égales (0.5).

Le premier corpus est composé de 20 000 séquences générées à partir de la RG. Sur l'ensemble des 360 corpus de test générés, la longueur moyenne des séquences est de 7.94 et la déviation standard moyenne de 3.30. Le nombre moyen d'échantillons de test est de 140 510.

Le second corpus se composant de 130 000 séquences, a été généré aléatoirement en utilisant les symboles de la RG. Ces séquences non grammaticales, dites aussi aléatoires, débutent toutes par B et finissent toutes par E mais ne respectent pas les transitions de la grammaire. La table 3.5 présente un exemple de séquences aléatoires. Sur l'ensemble des 360 corpus aléatoires générés, la longueur moyenne de ces séquences est de 7.98 et leur déviation standard moyenne de 6.48. Le nombre moyen d'échantillons traité se porte à 908 627. Nous avons aussi testé la probabilité que des séquences générées aléatoirement respectent la grammaire. Pour chaque séquence, nous avons testé les successions des symboles : A chaque symbole d'une séquence, si le symbole suivant est un successeur potentiel d'après la grammaire, alors l'analyse de la séquence continue, sinon elle est interrompue. Une séquence (entièrement analysée) dont la succession de symboles respecte la grammaire est considérée grammaticale. Sur les 360 corpus générés aléatoirement, le pourcentage de séquences non grammaticales moyen est de 99.89%.

Numéro séquence	séquence
1	BE
2	BVPXE
3	BTPPPPPE
4	BPSE
5	BSPPTTTTTTTE

TABLE 3.5 – Exemple de séquences générées aléatoirement à partir des symboles de la grammaire de REBER

Groupe de configuration	Numéro configuration	Nombre d'unités de la couche cachée	Taux d'apprentissage	Momentum
G0	C0	3	0.01	0.01
	C1	3	0.01	0.1
	C2	3	0.01	0.3
G1	C3	3	0.015	0.01
	C4	3	0.015	0.1
	C5	3	0.015	0.3
G2	C6	3	0.02	0.01
	C7	3	0.02	0.1
	C8	3	0.02	0.3
G3	C9	3	0.01	0.5
	C10	3	0.01	0.7
	C11	3	0.01	0.9
G4	C12	3	0.015	0.5
	C13	3	0.015	0.7
	C14	3	0.015	0.9
G5	C15	3	0.02	0.5
	C16	3	0.02	0.7
	C17	3	0.02	0.9
G6	C18	15	0.01	0.01
	C19	15	0.01	0.1
	C20	15	0.01	0.3
G7	C21	15	0.015	0.01
	C22	15	0.015	0.1
	C23	15	0.015	0.3
G8	C24	15	0.02	0.01
	C25	15	0.02	0.1
	C26	15	0.02	0.3
G9	C27	15	0.01	0.5
	C28	15	0.01	0.7
	C29	15	0.01	0.9
G10	C30	15	0.015	0.5
	C31	15	0.015	0.7
	C32	15	0.015	0.9
G11	C33	15	0.02	0.5
	C34	15	0.02	0.7
	C35	15	0.02	0.9

TABLE 3.6 – Configurations d'apprentissage pour SRN

•Apprentissage

Nous faisons apprendre au SRN 200 000 séquences RG avec 36 configurations d'apprentissage différentes, présentées table 3.6. Pour évaluer l'apprentissage, nous calculons l'erreur quadratique moyenne commise à chaque pas de temps. Puis nous lissons les résultats en calculant une moyenne à chaque pas de temps sur les 5000 dernières erreurs enregistrées.

Les tables 3.7 et 3.8 représentent le comportement du réseau doté de 3 unités cachées selon les configurations de C0 à C17. Seules les 2000 premières moyennes d'erreurs sont représentées, car c'est là que l'on observe les variations les plus importantes. On observe que pour toutes les valeurs du momentum inférieures à 0.5, les courbes représentant les moyennes des erreurs sont décroissantes. Cela implique que l'erreur diminue et donc que le réseau apprend à prédire la sortie attendue. Pour les valeurs de momentum de 0.7 et 0.9, l'erreur croit rapidement dès le début de l'apprentissage. Le réseau n'apprend donc pas.

Les tables 3.9 et 3.10 représentent le comportement du réseau doté de 15 unités cachées selon les configurations de C18 à C35. De même que pour les configurations précédentes, ici aussi seules les 2000 premières moyennes d'erreurs sont représentées, car c'est là que l'on observe les variations les plus importantes. Pour toutes les valeurs testées du momentum inférieures ou égales à 0.3, le réseau réussit à apprendre puisque sa courbe d'erreur moyenne décroit. Par contre, il s'avère qu'il n'arrive pas à apprendre une fois que la valeur du momentum dépasse 0.5. Pour l'ensemble des configurations de C27 à C35, le SRN affiche une erreur moyenne croissante, puis stagnante.

Sur les 36 configurations, celles qui présentent les meilleurs résultats sont les configurations C18, C19, C21, C22, C24 et C25. Ce sont les configurations pour lesquelles l'erreur représentée est la plus basse entre [0,6 ; 0,7]. Rappelons que l'erreur représentée est l'erreur quadratique moyenne lissée sur les 5000 derniers pas de temps (une erreur étant calculée à chaque pas de temps). Prenons le cas où l'erreur calculée $E = 0,6$, Cela implique que l'erreur se répartit sur les 7 unités de la couche de sortie du réseau. Chaque unité de sortie possède alors une erreur $E' = (0,6/7) = 0,085$. Or cette erreur étant au carré, cela induit pour chaque unité de sortie une erreur de $e' = \sqrt{E'} = \sqrt{0,085} = 0,29$. Cette valeur est importante pour des unités dont les valeurs d'activités sont comprises dans l'intervalle [0,1]. A ce stade, nous n'avons pas pu explorer les raisons de ces résultats et leur signification. Néanmoins, nous pensons que des travaux supplémentaires dans cette direction sont essentiels pour la bonne compréhension de cet algorithme.

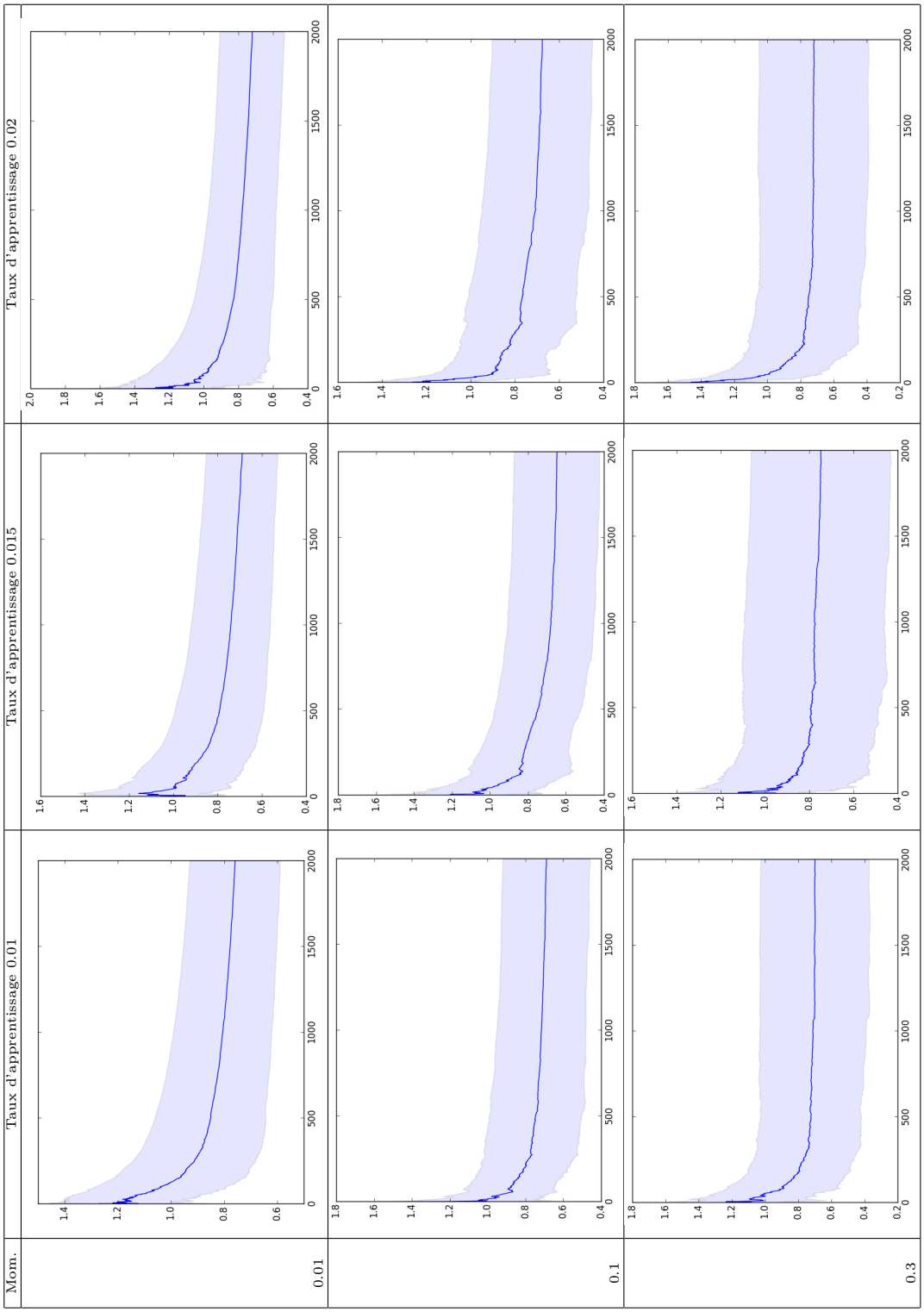


TABLE 3.7 – Evolution de la moyenne de l'erreur lors de l'apprentissage d'un SRN doté de 3 unités de la couche cachée selon les configurations d'apprentissage C0 à C8 (présentation des 2000 premières moyennes)

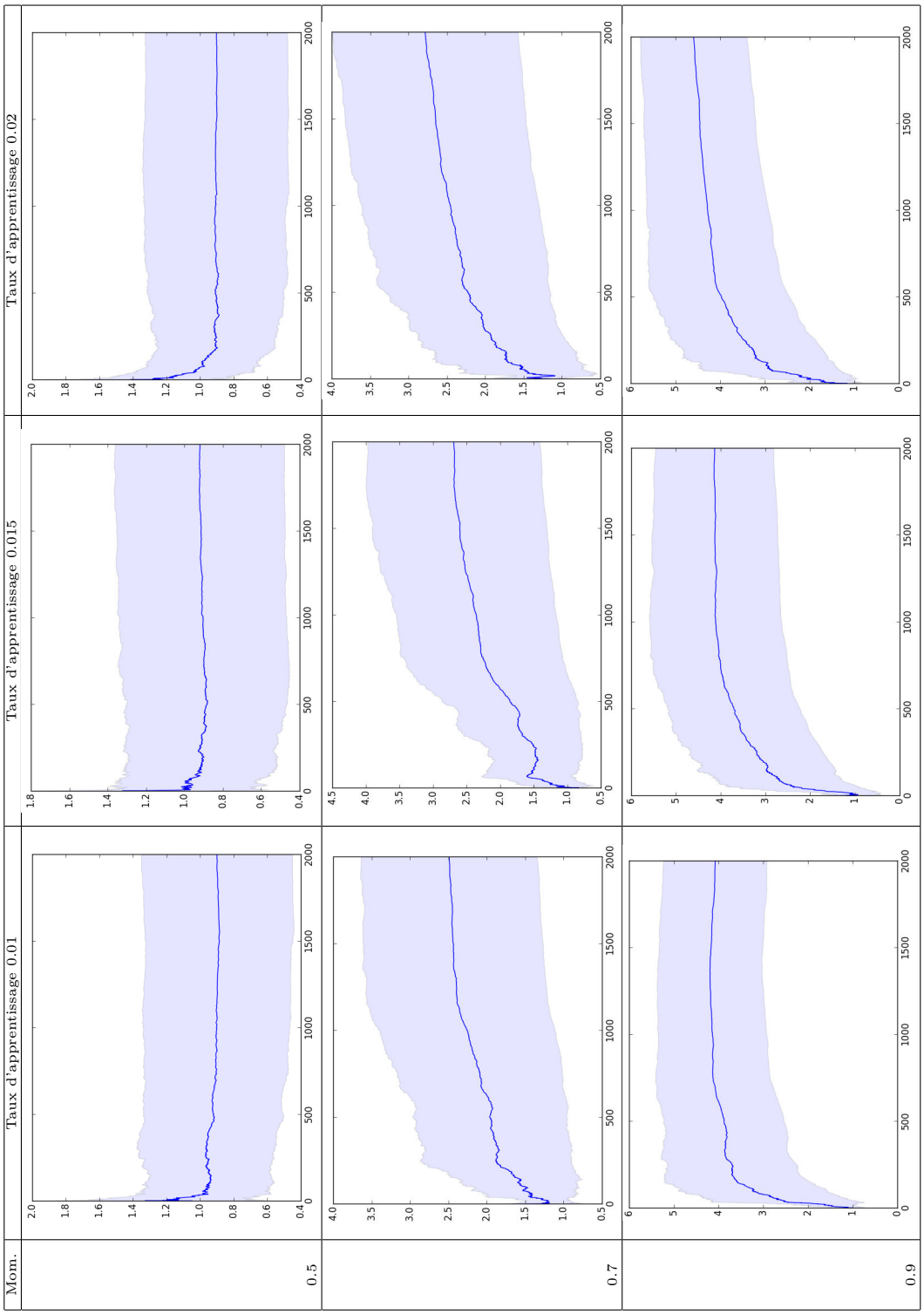


TABLE 3.8 – Evolution de la moyenne de l'erreur lors de l'apprentissage d'un SRN doté de 3 unités de la couche cachée selon les configurations d'apprentissage C9 à C17 (présentation des 2000 premières moyennes)

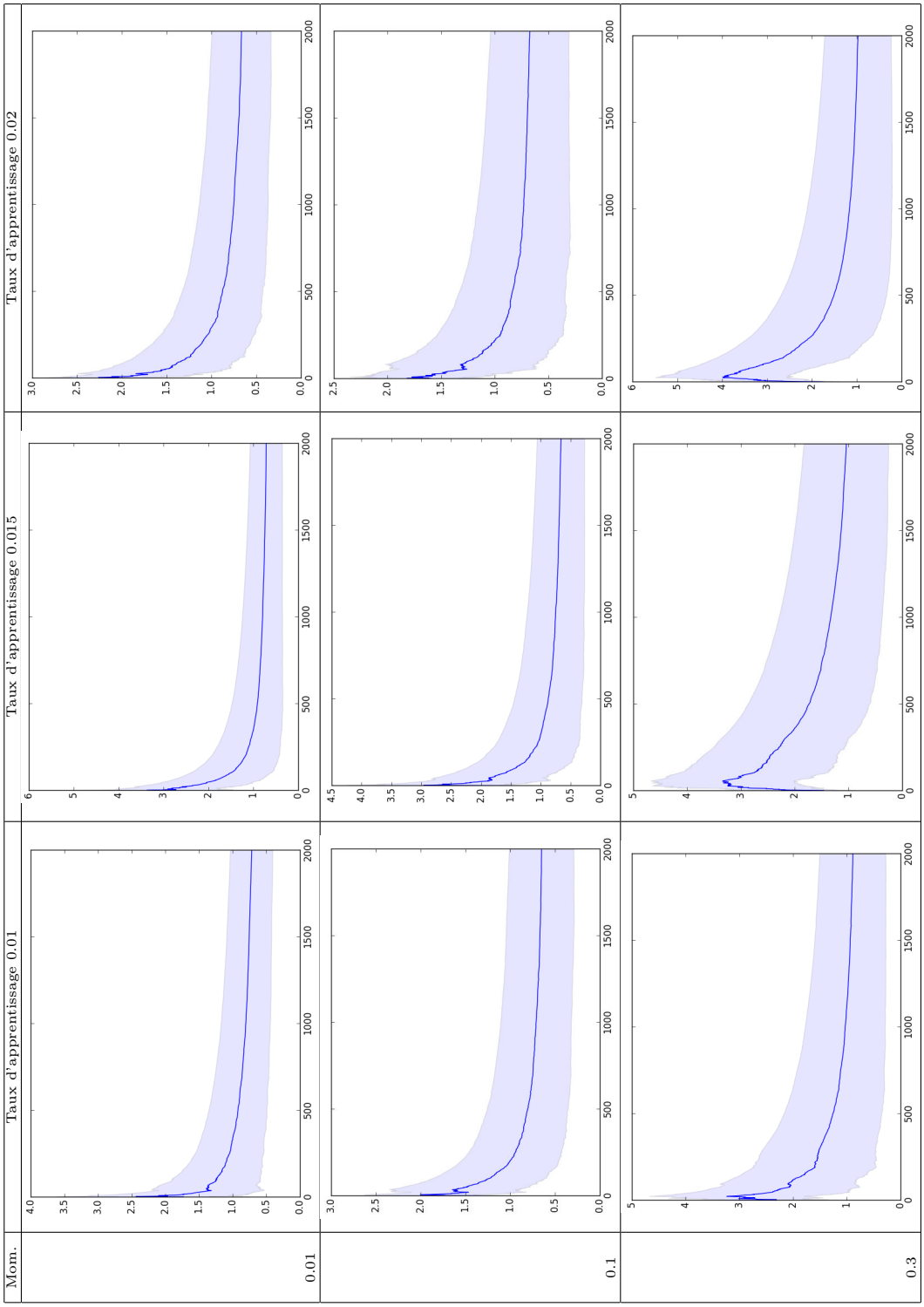


TABLE 3.9 – Evolution de la moyenne de l'erreur lors de l'apprentissage d'un SRN doté de 15 unités de la couche cachée selon les configurations d'apprentissage C18 à C26 (présentation des 2000 premières moyennes)

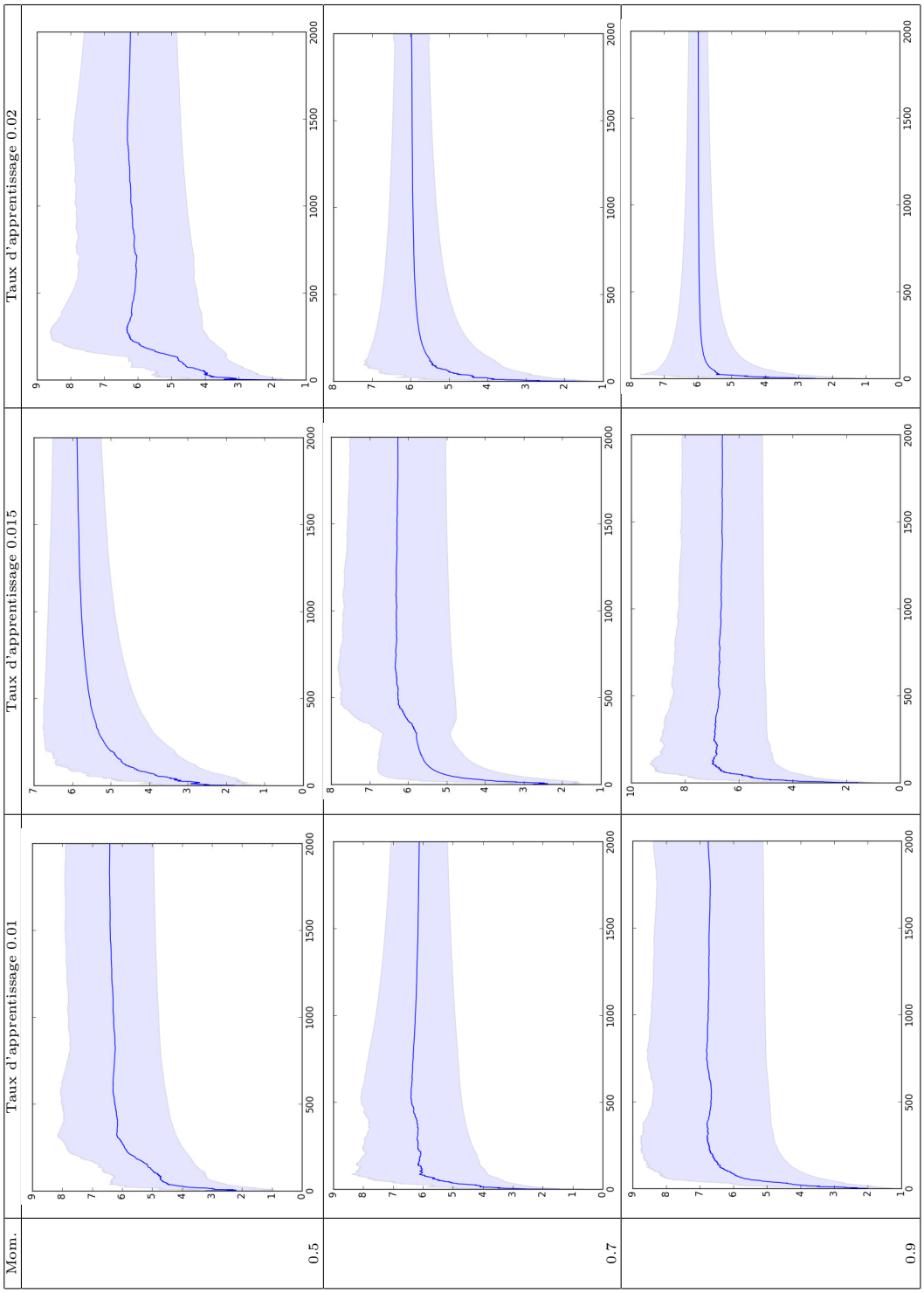


TABLE 3.10 – Evolution de la moyenne de l'erreur lors de l'apprentissage d'un SRN doté de 15 unités de la couche cachée selon les configurations d'apprentissage C27 à C35 (présentation des 2000 premières moyennes)

•Généralisation de l'apprentissage

Pour tester la performance du réseau et sa capacité à généraliser son apprentissage, nous réalisons 10 simulations consécutives pour chaque configuration d'apprentissage. A chaque simulation, comme dans l'article d'origine, deux corpus de tests sont générés : le premier comporte 20 000 séquences RG grammaticales, le second 130 000 séquences non grammaticales. Pour chacune des simulations, le réseau a été alimenté avec ces deux corpus et la prédiction réalisée par le réseau a été analysée à chaque pas de temps : Une prédiction est jugée correcte si, pour chaque symbole d'une séquence donnée, l'activation de son successeur est supérieure à 0,3 (le maximum étant 1) dans le pattern d'activité de la couche de sortie du réseau selon Servan-Schreiber *et al.* [1988]. Si ce critère n'est pas rempli, la présentation de la séquence au réseau est arrêtée et la chaîne est considérée rejetée car jugée non grammaticale. Une séquence "acceptée" est une séquence pour laquelle le réseau a su prédire correctement le symbole suivant à chaque pas de temps. Pour chaque simulation, deux mesures ont été réalisées : le pourcentage de séquences grammaticales acceptées et de séquences aléatoires acceptées. Un SRN performant devrait avoir les performances suivantes : un taux élevé (proche de 100%) de reconnaissances des séquences grammaticales et un taux faible (proche de 0%) de reconnaissances des séquences non grammaticales. Voici un exemple : Soit BTXSE une séquence obéissant à la grammaire de REBER. Au pas de temps t_0, le réseau reçoit le vecteur binaire représentant B en entrée. En fonction de l'apprentissage réalisé, le réseau fournit un vecteur de taille 7 composé de valeurs réelles, chacune d'entre elles indiquant l'activité d'une unité de la couche de sortie. Les unités dont l'activité est supérieure à 0.3, sont celles désignant les successeurs potentiels. Un réseau peut donc au pas de temps t, prédire plus d'un successeur. Si le symbole suivant, dans ce cas T, fait partie des successeurs potentiels prédits par le réseau, le processus d'analyse de la séquence peut continuer. Dans le cas inverse, la séquence est rejetée. Si la séquence est présentée dans son ensemble au réseau, cela implique qu'elle est acceptée. Les figures 3.5, 3.6, 3.7 et 3.8 présentent les performances moyennes sur 10 simulations du SRN selon les différentes configurations présentées dans la figure 3.6.

Résultat du SRN à 3 unités cachées :

La table 3.5 montre qu'un SRN de 3 unités cachées ayant appris avec un faible taux d'apprentissage et un momentum faible, présente de faibles performances. Aucune des configurations de C0 à C8, ne dépassent les 25% de séquences grammaticales reconnues, ce qui est relativement faible. A contrario, le réseau semble bien rejeter les séquences aléatoires. La table 3.7 affiche l'évolution de l'erreur du SRN pour ces différentes configurations. Celle-ci décroit, ce qui induit que le réseau réussi à faire un apprentissage.

La figure 3.6 montre les performances d'un SRN de 3 unités cachées uti-

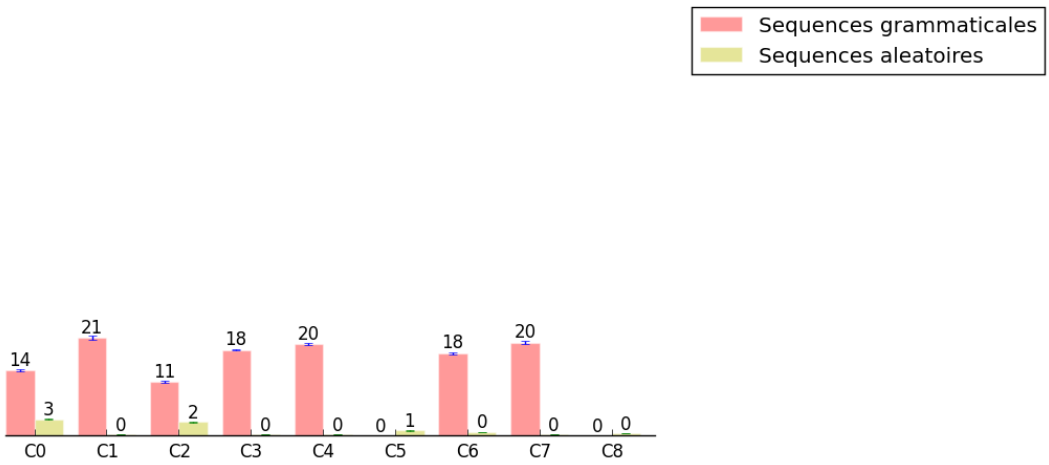


FIGURE 3.5 – Evolution des performances d'un SRN de 3 unités cachées pour les configurations C0 à C8. Moyennes obtenues sur 10 simulations.

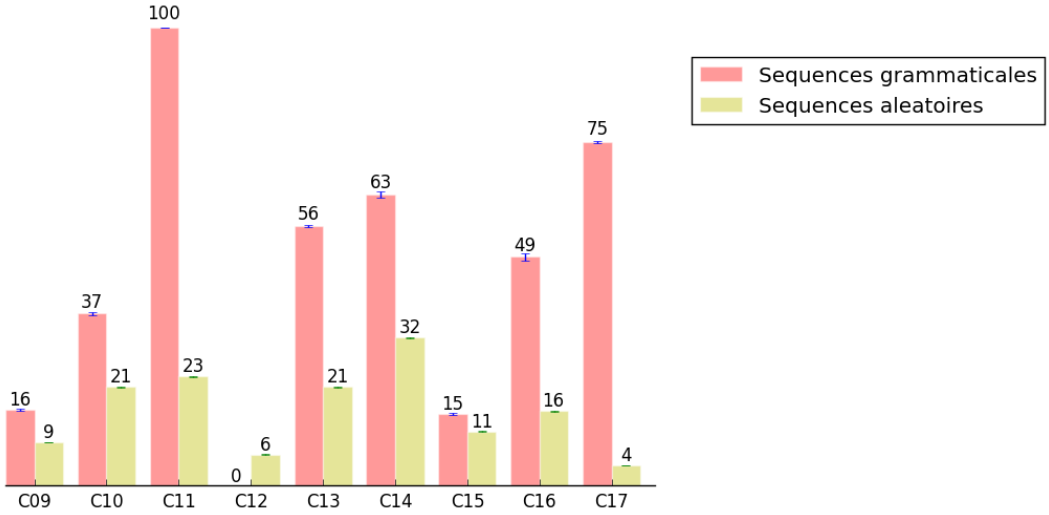


FIGURE 3.6 – Evolution des performances d'un SRN de 3 unités cachées pour les configurations C9 à C17. Moyennes obtenues sur 10 simulations.

lisant une configuration recommandée par Servan-Schreiber *et al.* [1988]. Si l'on observe les configurations des groupes G3 (C9 à C11), G4 (C12 à C14) et G5 (C15 à C17) (figure 3.6), on constate que l'augmentation du momentum entraine une augmentation du taux de reconnaissance des séquences grammaticales au sein de chaque groupe, tout en gardant un taux de reconnaissance de séquences aléatoires acceptable (inférieur à 40%). Les performances semblent meilleures que celles précédentes alors que le réseau n'apprend pas forcément bien (ses courbes d'erreurs sont croissantes pour un momentum supérieur à 0.5). Seules les configurations C9, C12 et C15 permettent au réseau d'apprendre. Sur l'ensemble des configurations testées pour un SRN de 3 unités cachées, la configuration C11 semble être celle qui présente les meilleurs résultats de généralisation, bien que la courbe d'erreur associée soit croissante.

Résultat du SRN à 15 unités cachées :

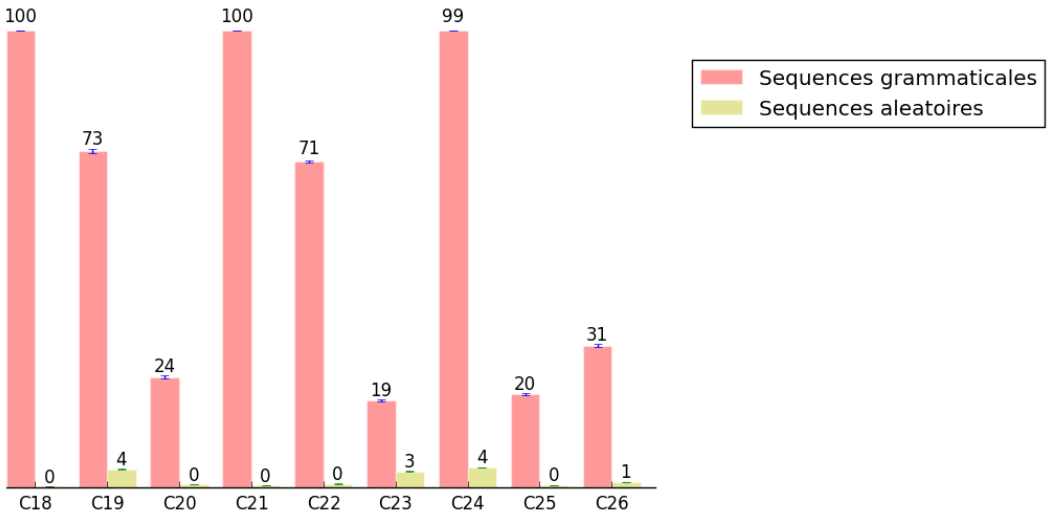


FIGURE 3.7 – Evolution des performances d'un SRN de 15 unités cachées pour les configurations C18 à C26. Moyennes obtenues sur 10 simulations.

La figure 3.7 montre les performances d'un SRN de 15 unités cachées. Dans cette figure, l'augmentation du momentum au sein de chacun des groupes G6 (C18 à 19), G7 (C21 à C26) et G8 (C24 à C26) induit une baisse des performances. Le taux de reconnaissance des séquences grammaticales baisse de plus de 75% en moyenne (entre la plus haute performance et la plus basse) sur les 3 groupes. A contrario, la variation du momentum, n'a pas d'impact significatif au niveau du taux de reconnaissance des séquences aléatoires. Les configurations C18, C21 et C24 sont avec le momentum le plus faible (0.01) et présentent les meilleures performances. La table 3.9 montrent que les courbes d'erreurs de ces configurations sont décroissantes. Les configurations C18, C21

et C24 sont donc celles où le réseau apprend bien, mais arrive aussi à généraliser son apprentissage.

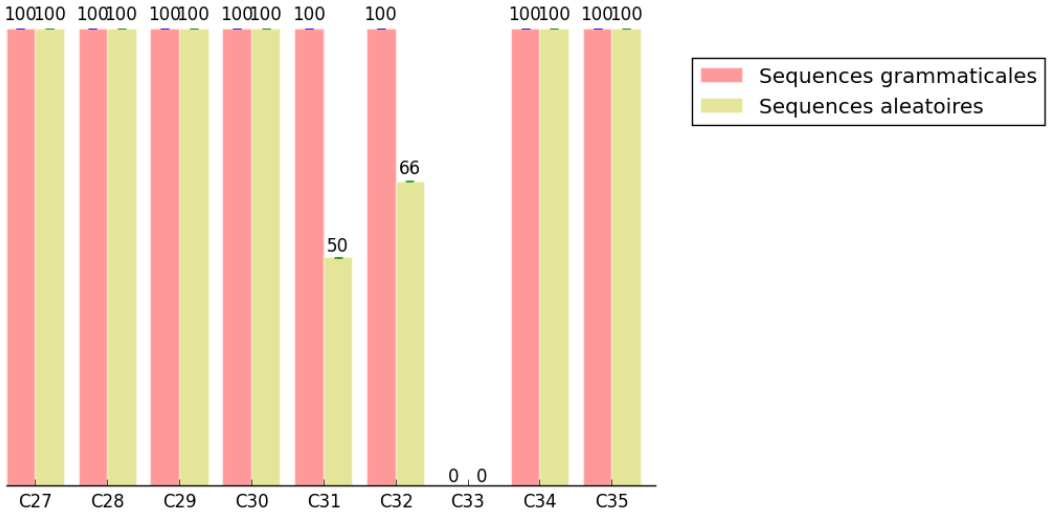


FIGURE 3.8 – Evolution des performances de Elman pour les configurations C27 à C35. Moyennes obtenues sur 10 simulations.

La figure 3.8 présente les performances d'un SRN de 15 unités cachées avec un taux d'apprentissage successivement de [0.01, 0.015, 0.02] et un momentum prenant les valeurs successives de [0.5, 0.7, 0.9], pour chaque taux d'apprentissage. Sur 10 simulations, le taux moyen de reconnaissance des séquences grammaticales et des séquences aléatoires est de 100%. Cela implique que le réseau ne sait pas distinguer entre ces deux types de séquences. L'étude plus précise de la prédiction a montré que le réseau prédisait à chaque pas de temps l'ensemble des symboles possibles comme successeurs potentiels, ce qui justifie son incapacité à classer des séquences. Au niveau de l'erreur du réseau durant son apprentissage, la table 3.10 montre qu'elle est croissante pour l'ensemble des configurations de C27 à C35. Le réseau ne réalise pas d'apprentissage pour ces configurations là.

•Résistance aux dépendances séquentielles sur le long terme

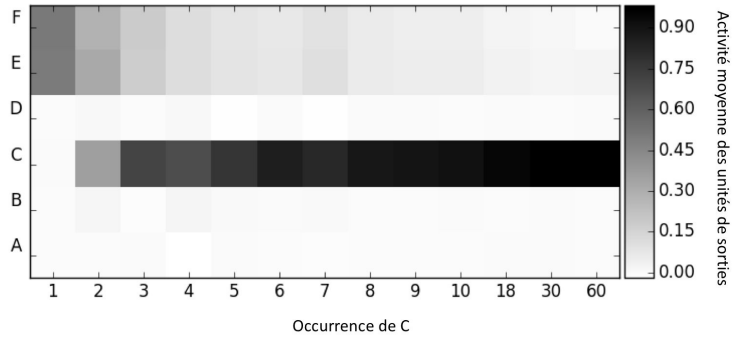


FIGURE 3.9 – Activité moyenne des unités de la couche de sortie pour un SRN de 3 unités cachées. Apprentissage réalisé avec : taux d'apprentissage=0.01 et momentum=0.01.

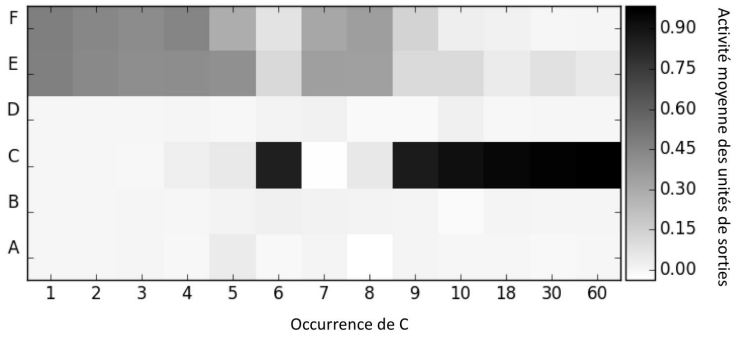


FIGURE 3.10 – Activité moyenne des unités de la couche de sortie pour un SRN de 15 unités cachées. Apprentissage réalisé avec : taux d'apprentissage=0.01 et momentum=0.01.

Nous avons réalisé un test supplémentaire afin de tester le comportement des deux SRN face à des dépendances séquentielles. Nous avons pour cela, utilisé deux séquences issues de la grammaire manuelle qui débutent et se terminent différemment, mais qui comportent le même nombre i de symboles C intermédiaires : AC^iF et BC^iE. Pour ce test, nous avons réalisé un apprentissage en faisant varier le taux d'apprentissage comme ce qui suit [0.01, 0.015, 0.02] et le momentum comme cela [0.01, 0.5, 0.7, 0.9]. Les couches d'entrées et de sorties se composent toutes deux de 6 unités pour les 6 symboles suivants : ['A', 'B', 'C', 'D', 'E', 'F']. Ces deux séquences ont été apprises sur 1000 itérations, soit 500 fois chacune, l'une alternant avec l'autre. Le nombre de C a varié selon les valeurs suivantes : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 30, 60]. Les courbes représentant les moyennes d'erreurs sont toutes décroissantes (la moyenne a

été calculée sur les 50 dernières erreurs à chaque fois) signifiant que le réseau apprend bien. Lors de la phase de test, les mêmes séquences ont été présentées au RNN. Les patterns de la couche de sortie au dernier pas de temps de chaque séquence, pour différentes occurrences de C, ont été enregistrés. Les figures 3.9 et 3.10 présentent l'activité moyenne des unités de la couche de sortie. Il s'agit de la somme des activités de chaque unité de la couche de sortie au dernier pas de temps de chaque séquence, divisé par le nombre de séquences. Chaque unité étant associé à un symbole, ces figures permettent de visualiser les tendances de prédiction du réseau au dernier pas de temps.

L'analyse de ces deux figures montrent que les réseaux se retrouvent rapidement (dès 2C et dès 6C pour des SRN de respectivement 3 unités cachées et 15 unités cachées) à prédire majoritairement le symbole C à la fin des deux séquences et cela indépendamment du début. Dans la figure 3.9, le réseau prédit de manière égale les symboles E et F pour les séquences comportant de 1 C. Mais au delà de ce palier, le réseau change progressivement de comportement et prédit le C comme dernier élément des séquences (indépendamment du type de séquence traitée) : en parallèle de l'atténuation de l'activité moyenne des unités prédisant E et F, on observe une augmentation progressive de l'activité moyenne de l'unité prédisant C. Dans la figure 3.10, le réseau prédit de manière égale les symboles E et F pour les séquences comportant de 1 à 4 C. Ensuite, pour les séquences de 5 à 8 C, il prédit de manière sensiblement égale (au dernier pas de temps) les symboles C, E et F. Puis à partir de 9 C répétés, le réseau ne prédit plus que des symboles C. Le changement progressif de prédiction observé pour les deux réseaux traduit un changement progressif dans la capacité de ce dernier à conserver son passé lointain. Le souvenir des premiers symboles (A et B) est progressivement écrasé par un passé récent, celui des symboles C les plus récents. Ce phénomène s'accentue lorsque le réseau doit traiter des séquences de plus en plus longues avec de plus en plus de symboles C. Par ailleurs dans le cas du réseau à 15 unités cachées, nous observons une variation ponctuelle des performances lorsque les séquences sont composées de 6C. Cette observation met en évidence une instabilité du réseau en terme de performances.

Discussion

L'étude des performances du SRN selon différentes configurations d'apprentissage met en évidence la sensibilité du réseau aux paramètres tels que le taux d'apprentissage et le momentum. Par ailleurs, nous avons montré que lorsque le réseau apprend (courbe d'erreur décroissante), cela n'implique pas qu'il soit capable de généraliser son apprentissage. Malgré le fait que nous nous sommes conformé aux paramètres selon la littérature existante, nous observons des résultats qui fluctuent. Le test utilisant la grammaire manuelle, quant à

lui, souligne d'une part la sensibilité du réseau au nombre d'unités de la couche cachée et d'autre part, montre que celui-ci finit par perdre rapidement sa capacité à distinguer le contexte (au bout de 2C pour un SRN de 3 unités cachées et 6C pour un SRN de 15 unités cachées). De ce test, il en ressort que l'augmentation du nombre d'unités de la couche cachée, permet de maintenir les performances du réseau et notamment sa capacité à conserver le souvenir du passé lointain. Néanmoins, cela laisse aussi penser qu'il existe un seuil au delà duquel le phénomène d'oublie a lieu, et cela quelque soit le nombre d'unités cachées du SRN. D'après Cleeremans et McClelland [1991], le SRN possède la particularité d'apprendre les patterns pertinents pour distinguer un contexte d'un autre. Or dans notre test, il est montré que lorsque celui ci est redondant, et qu'il remonte loin dans le passé, le réseau finit par l'oublier, les informations plus récentes venant écraser celles stockées et plus anciennes. Dans le cadre de la question d'extraction des règles à partir d'un RNN, il nous faut souligner une conséquence importante du phénomène d'oubli du passé lointain au profil du passé récent : celle de la perte des relations temporelles entre les états. Le passé devenant de plus en plus flou, cela implique qu'il sera de plus en plus difficile de reconstruire la ligne temporelle des séquences traitées (la succession des symboles).

Nous venons de présenter les performances du SRN selon les expériences de Servan-Schreiber *et al.* [1988]. Nous allons étudier dans la section suivante les performances du RNN-LSTM lorsqu'il est soumis aux mêmes contraintes d'apprentissage et aux mêmes tailles de corpus de données.

3.3.2 RNN-LSTM

Parmi l'ensemble des variations existantes du modèle LSTM, nous avons choisi de nous focaliser sur la version standard proposée par Gers *et al.* [1999] et que nous avons décrit section 2.3.2. Dans cet article, les auteurs proposent l'association des LSTM de Hochreiter et Schmidhuber [1997] et d'une porte de sortie afin de permettre l'apprentissage de séquences mais aussi de flux (séquences mises bout à bout) issus des variations de la grammaire de Reber (ERG et CERG). Ils ont ainsi montré l'efficacité mais aussi la solidité de ces nouveaux LSTM face à des corpus de données très volumineux (30 000 flux de taille 100 000). Nous présentons ici notre implémentation d'un RNN-LSTM inspiré des travaux de Gers et des paramètres proposés. Nous décrirons les expériences menées pour tester les performances du réseau.

Architecture du RNN-LSTM et apprentissage

Notre RNN-LSTM se compose de 3 couches : une couche d'entrée, une couche cachée et une couche de sortie. Les couches d'entrée et de sortie se

composent toutes deux de 7 unités, du fait que les grammaires utilisées comportent 7 symboles. La couche cachée se compose de 4 blocs LSTM chacun doté de 2 cellules LSTM et de 3 portes, soit 8 cellules et 12 portes en tout. L'ensemble des unités de la couche d'entrée sont connectées à l'ensemble des unités de la couche de sortie, à chacune des 8 cellules, et à chacune des portes des 4 blocs. L'ensemble des cellules ont des connexions vers chacune des unités de sorties, mais aussi vers l'entrée de toutes les cellules (une cellule a donc une connexion récurrente avec elle même) et enfin, vers chacune des portes de chaque bloc. Pour l'apprentissage, nous utilisons une combinaison du BPTT et RTRL, décrite par Gers *et al.* [1999] et présentée en section 2.3.2. Un pseudo-code décrivant l'apprentissage est fourni en annexe E.

Le nombre de poids paramétrés $nb_{W_{RNN-LSTM}}$ lors de l'apprentissage se calcule donc comme ce qui suit :

$$\begin{aligned} nb_{W_{RNN-LSTM}} = {} & nb_{entree} * nb_{bloc} * nb_{cellulesParBloc} + \\ & nb_{entree} * nb_{bloc} * nb_{porteParBloc} + nb_{entree} * nb_{sortie} + \\ & (nb_{bloc} * nb_{cellulesParBloc}) * (nb_{bloc} * nb_{cellulesParBloc}) + \\ & (nb_{bloc} * nb_{cellulesParBloc} * nb_{bloc} * nb_{porteParBloc}) + \\ & (nb_{bloc} * nb_{cellulesParBloc}) * nb_{sortie} + (nb_{bloc} * nb_{porteParBloc}) + nb_{sortie} \end{aligned} \quad (3.4)$$

Ainsi avec les paramètres de notre RNN-LSTM, nous obtenons un total de :

$$\begin{aligned} nb_{W_{RNN-LSTM}} = {} & 7 * 4 * 2 + 7 * 4 * 3 + 7 * 7 + (4 * 2) * (4 * 2) + \\ & (4 * 2 * 4 * 3) + (4 * 2) * 7 + (4 * 3) + 7 = 424 \end{aligned} \quad (3.5)$$

Les poids des biais des portes d'entrées et de sorties sont initialisés à -0.5 pour le premier bloc, puis sont décrémentés de -0.5 pour les suivants (-1, -1.5, -2). Pour les portes d'oubli des 4 blocs, c'est l'inverse qui s'opère : pour le premier bloc, le biais de la porte est initialisé à 0.5, puis s'incrémente de 0.5 pour les autres portes des autres blocs (1, 1.5, 2). Tous les autres poids sont initialisés, ainsi que les biais de la couche de sortie, de manière aléatoire entre [-0.2, 0.2]. Pour cette implémentation, pas de momentum, mais un taux d'apprentissage initialisé à 0.5 puis multiplié par 0.99 tous les 100 pas de temps. Lors de l'apprentissage, mais aussi lors de la prédiction, au début de chaque séquence, les états précédents s_{t-1}, ainsi que les entrées aux pas de temps précédents sont remis à zéro. Le contexte est ainsi réinitialisé au début de chaque séquence.

Reproduction des expériences de Servan-Schreiber : Apprentissage de séquences et généralisation

Nous avons reproduit les expériences menées sur SRN avec les mêmes tailles de corpus. Nous présentons ci-dessous les résultats de ces tests menés sur des séquences RG et menés sur les séquences ERG. Pour chaque grammaire, nous avons réalisé un apprentissage de 200 000 séquences avec un RNN-LSTM. Nous avons ensuite réalisé 10 simulations, pour lesquelles 2 corpus de séquences ont été générés à chaque fois : le premier de 20 000 séquences issues de la grammaire considérée et 130 000 séquences aléatoires utilisant les symboles de cette grammaire.

	RG	ERG
Corpus d'apprentissage		
Taille	200 000	200 000
Nombre d'échantillons	1 397 109	2 198 671
Longueur moyenne	7.98	11.99
Déviation standard	3.35	3.37
Corpus de séquences grammaticales de test		
Taille	20 000	20 000
Nombre d'échantillons moyen	140 193	219 411
Longueur moyenne	8.02	11.97
Déviation standard	3.47	3.35
Corpus de séquences aléatoires de test		
Taille	130 000	130 000
Nombre d'échantillons moyen	140 538	219 533
Longueur moyenne	8.00	7.01
Déviation standard	6.51	5.50

TABLE 3.11 – Caractéristiques des corpus utilisés. Les chiffres pour les corpus de tests sont des moyennes calculées sur 10 corpus.

Dans ce contexte de RNN-LSTM, une prédiction est considérée correcte si le carré de l'erreur quadratique moyenne de chacune des 7 unités de sortie est inférieur à 0.49. Nous avons réalisé cette expérience d'une part avec des séquences issues de la grammaire de Reber et des séquences issues de la grammaire de Reber embarquée. Dans ce cas aussi, nous avons mesuré le pourcentage de séquences grammaticales acceptées et le pourcentage de séquences aléatoires acceptées. Un RNN-LSTM ayant de bonnes performances présentera un taux élevé (proche de 100%) de séquences grammaticales et un taux faible (0%) de séquences aléatoires. Les figures 3.11 et 3.12 présentent les courbes des erreurs moyennes dans le cas de RNN-LSTM appliqués à des séquences RG et des séquences ERG. Tout comme les SRN, il s'agit de l'erreur quadratique

moyenne commise à chaque pas de temps, puis lissée en calculant une moyenne à chaque pas de temps sur les 5000 dernières erreurs enregistrées. Dans les deux cas, les erreurs décroissent ce qui témoigne de bonnes performances d'apprentissage.

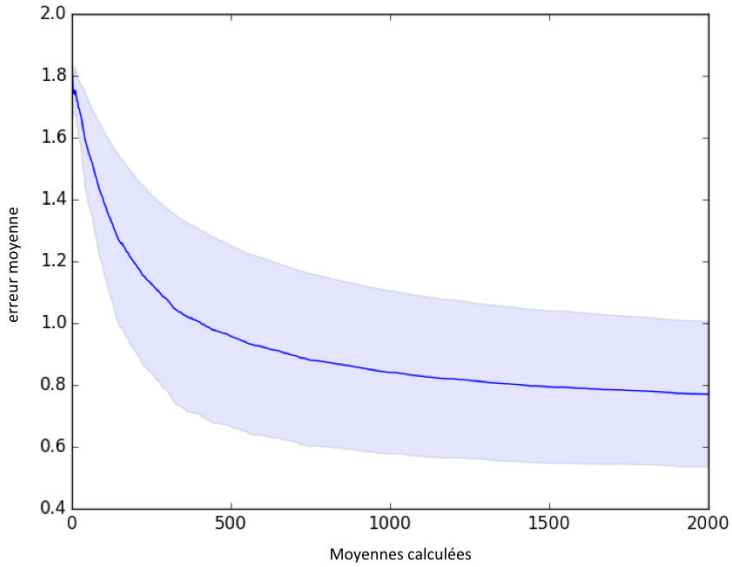


FIGURE 3.11 – Evolution de la moyenne de l'erreur lors de l'apprentissage du RNN-LSTM de 200 000 séquences issues de la grammaire de Reber

La figure 3.13 présente les résultats moyens sur 10 simulations pour la grammaire de Reber et la grammaire ERG. Le RNN-LSTM présente de bonnes performances avec un taux de séquences grammaticales acceptées à 99% pour les deux grammaires et un taux de séquences aléatoires acceptées faibles. Ces résultats alliés à ceux des figures 3.11 et 3.12, témoignent de la bonne capacité du RNN-LSTM non seulement pour apprendre, mais aussi pour généraliser son apprentissage. Une fois les règles encodées (suite à son apprentissage), le réseau arrive avec succès à reconnaitre les séquences grammaticales nouvelles qui lui sont soumises et à rejeter celles aléatoires, qui ne respectent pas les règles.

Reproduction des résultats de Gers : apprentissage de flux

Nous montrons dans cette partie les performances du modèle concernant l'apprentissage de flux continu. Nous définissons un flux, comme un ensemble de séquences ERG mises bout à bout. Gers définit une solution parfaite comme 10^6 prédictions successives sans erreur. Pour réaliser ce test, nous alternons apprentissage et prédiction : Nous faisons apprendre au réseau un flux de 100 000 symboles successifs au bout duquel nous gelons les poids. Ces poids sont utilisés pour une phase de tests portant sur 10 flux de 100 000 symboles. Le nombre

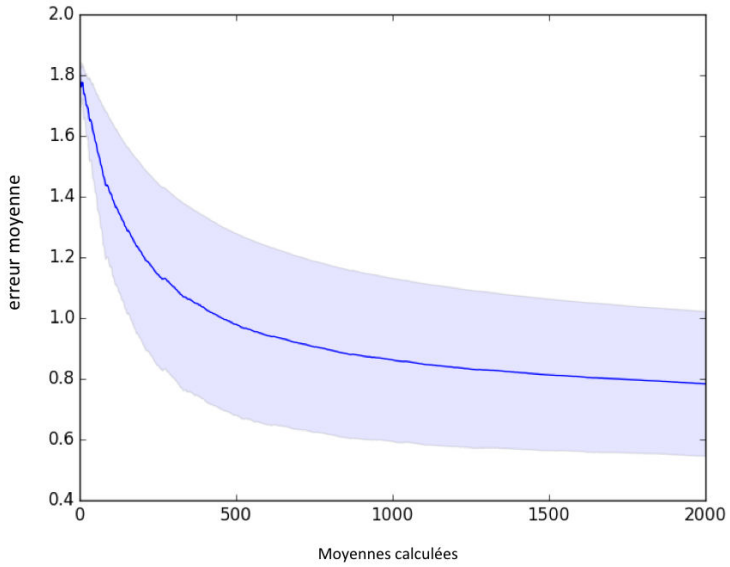


FIGURE 3.12 – Evolution de la moyenne de l'erreur lors de l'apprentissage du RNN-LSTM de 200 000 séquences issues de la grammaire de Reber Embarquée

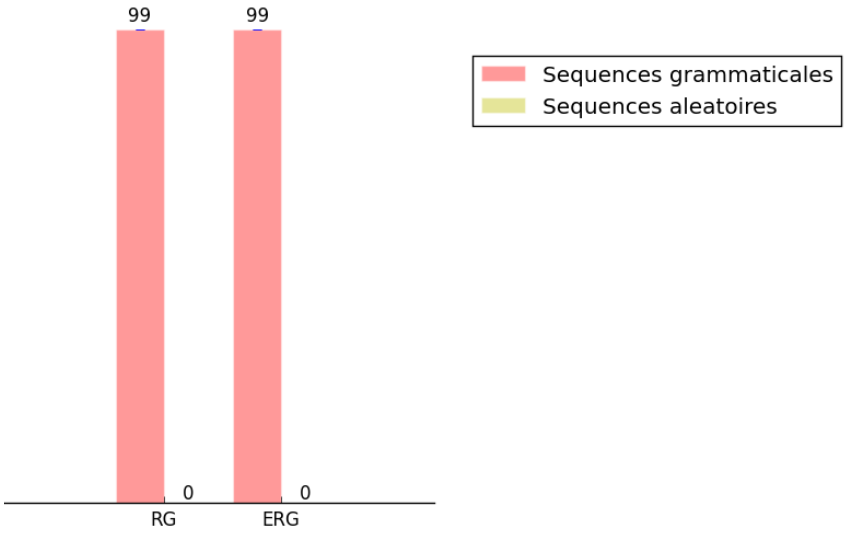


FIGURE 3.13 – Performances moyennes sur 10 simulations pour RNN-LSTM appliquées à la grammaire de Reber (RG) et la grammaire de Reber embarquée (ERG).

d'erreurs moyen de prédiction a été mesuré à chaque phase de tests. Nous avons réussi à tester 30 000 flux d'entrainement sans que le réseau ne présente d'erreur. Autrement dit, le réseau a présenté 100% de prédictions correctes sur l'ensemble des phases de tests selon le critère de Gers.

Ce test a permis de tester la résistance du réseau face aux flux de symboles, où il n'y a pas de début ou de fin explicite. Le réseau a su gérer l'ambiguïté des séquences ERG et lorsque celles-ci s'enchainent, le réseau a continué à les prendre en charge. Le réseau RNN-LSTM est par conséquent capable de réaliser un bon apprentissage de séquences simples, mais aussi plus ambiguës ou encore de flux de symboles.

Dépendances séquentielles : SRN vs LSTM

Nous avons montré la faiblesse de SRN lors de l'apprentissage de la grammaire manuelle. Lorsque l'information pertinente à conserver est trop loin dans le passé (i.e lorsque le nombre de C est important) le réseau n'arrive pas à réaliser la bonne prédiction. L'information sur le court terme vient écraser celle qui devait être conservée. Ce qui engendre une faible capacité à retenir des informations (un contexte) sur le long terme. L'utilisation de la grammaire RG a permis de tester les capacités des SRN et RNN-LSTM à apprendre et prédire selon un contexte où le début et la fin de séquences sont facilement identifiables. Les deux RNN ont montré de bonnes performances d'apprentissage et de généralisation, avec un bémol cependant pour le SRN dont les performances sont très variables selon les paramètres. La grammaire ERG, teste la capacité des réseaux à gérer l'ambiguïté au niveau des symboles de début et de fin de séquences. Ces dernières ne sont plus aux extrémités seulement, mais aussi à l'intérieur des séquences. Le modèle LSTM a montré de bonnes performances pour cela : sur 10 simulations, il a reconnu l'ensemble des séquences ERGs comme grammaticale (100% de séquences acceptées) et rejeté la majorité des séquences aléatoires (pourcentage d'acceptation faible de 3%). Enfin la grammaire CERG, supprime la notion de début et de fin. Les réseaux doivent apprendre un flux continu et être capables de faire les prédictions adéquates en fonction du contexte local et de celui plus loin dans le passé. Le RNN-LSTM a montré de très bonnes performances puisque aucune erreur de prédiction n'a été enregistrée sur l'ensemble de 30 000 flux de 100 000 symboles. En résumé, le RNN-LSTM a montré de meilleures performances que le SRN au niveau de l'apprentissage mais aussi de la généralisation de cet apprentissage. Par ailleurs, il a aussi montré de bonnes performances dans la prédiction des symboles successifs d'un flux, alors que le SRN a des performances fluctuantes et souvent faibles avec des séquences de Reber simples. La répétition de l'apprentissage et de la phase de test, 30 000 fois sur 100 000 flux différents composés de séquences ERG, a montré la solidité du RNN-LSTM à

maintenir l'information sur le long et court terme.

Discussion

Dans cette partie nous souhaitions comparer les performances des 2 RNN en termes d'apprentissage séquentiel et de généralisation. Nous avons montré que le SRN, malgré son apprentissage et ses performances pour la configuration C11, possède de nombreuses contraintes liées au choix des paramètres. Le nombre de neurones dans la couche cachée, le taux d'apprentissage mais aussi le momentum, peuvent induire une augmentation des erreurs et/ou une mauvaise généralisation. Et cela bien que nous nous soyons appuyé sur des données issues de la littérature. A contrario, les LSTM ont montré de bonnes performances pour l'apprentissage de séquences simples RG, de séquences plus ambiguës ERG, mais aussi de flux CERG. Nous observons aussi que l'apprentissage induit de bonnes performances de généralisation. Le modèle RNN-LSTM est donc plus performant pour l'apprentissage de séquences complexes porteuses de contexte local et plus étendu dans le temps que le modèle SRN. Le RNN-LSTM présentant des meilleures performances indique que sa représentation interne est de meilleure qualité.

Néanmoins, il nous faut soulever ici un point sur le RNN-LSTM : Pour nos simulations, notre protocole de test a nécessité l'apprentissage par le réseau de 200 000 séquences durant lesquelles le taux d'apprentissage a été mis à jour tous les 100 pas de temps (soit une fois tous les 100 symboles présentés). Pour la grammaire RG, nous avons mesuré que le taux d'apprentissage a été mis à jour 14 000 fois. Or d'après la figure 3.14 qui présente l'évolution du taux d'apprentissage durant les 1000 premières mises à jour, (10 000 premiers symboles présentés), la valeur du taux d'apprentissage décroit rapidement et avoisine la valeur nulle dès la 800ème mise à jour. Cela laisse entendre que le réseau au final n'apprend plus au delà ce seuil. Ce point sera discuté ultérieurement dans la section 3.5.

3.4 Extraction de règles à partir du RNN-LSTM

Dans cette section, nous allons présenter un nouveau modèle réalisé dans le but d'extraire des règles encodées dans un RNN-LSTM après l'apprentissage de plusieurs séquences régies par ces règles. Notre but est de trouver un moyen d'expliciter ce qui se passe dans un réseau de neurones qui apprend une grammaire. Nous allons dans un premier temps décrire les données que nous avons exploitées. Dans un second temps, nous expliciterons les résultats préliminaires obtenus grâce au clustering hiérarchique. Enfin, nous expliciterons notre procédé d'extraction des règles et les résultats obtenus avec des séquences issues de la grammaire de Reber, puis de la grammaire de Reber embarquée. Nous

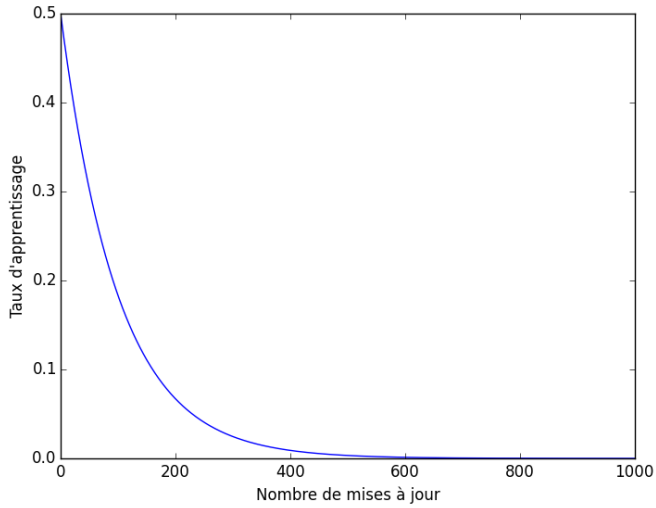


FIGURE 3.14 – Evolution du taux d'apprentissage lors de l'apprentissage de 10 000 symboles (soit 1000 mises à jour du taux d'apprentissage) issues de séquences RG avec le RNN-LSTM décrit section 3.3.2.

appliquerons, pour finir, nos travaux sur une grammaire électrique, extraite manuellement par nos soins à partir de l'étude de schémas électriques représentant des installations réelles. L'algorithme 3 décrit le processus d'extraction des règles sous la forme d'un graphe-automate (GA) doté d'étiquettes longues (figure 3.18) à partir des patterns d'activité d'un RNN-LSTM.

3.4.1 Patterns d'activités de la couche cachée

Lors d'un apprentissage de séquences dans un RNN, c'est au niveau de l'espace d'états des unités de la couche cachée, que se trouvent encodées les règles abstraites des séquences. Pour extraire celles-ci, nous avons utilisé le RNN-LSTM décrit précédemment dans ce chapitre. Nous avons d'abord réalisé un apprentissage sur les séquences issues des grammaires RG et ERG. Une fois cela fait, nous avons généré un corpus de test issu de la même grammaire, que nous avons fourni au réseau. A chaque pas de temps (autrement dit à chaque présentation de symbole), nous avons choisi de récupérer et stocker, en plus du symbole traité à l'instant t, les patterns d'activités de la couche cachée, soit les sorties des cellules LSTM. En nous focalisant sur cette information, parmi les multitudes que les unités LSTM contiennent, nous souhaitions transposer les approches connues d'extraction des règles à partir des patterns d'activités de la couche cachée des SRN sur des LSTM. Nous nous sommes ainsi constitué une liste de patterns d'activité, chacun doté d'un identifiant unique constitué du symbole courant et du pas de temps.

Algorithme 3 Algorithme d'extraction des règles sous la forme d'un graphe-automate (GA) doté d'étiquettes longues à partir des patterns d'activité d'un RNN-LSTM

Pré-conditions : # Apprentissage et test du réseau de neurones ——
RNN_LSTM.apprentissage(corpus_apprentissage)
# liste_etiquettes : liste des symboles présentés au réseau durant les tests
liste_patterns_activité, liste_etiquettes = RNN_LSTM.test(corpus_test)

Fonction extraction_règles (liste_patterns_activité, liste_etiquettes, k) :
# Clustering ——————————————————-
liste_clusters = k_means(k, liste_patterns_activité)

# Creation automate ——————————————————-
GA = {} # Dictionnaire
noeud_courant= -1
GA['noeuds'].ajouter(noeud_courant)
GA['arcs'] = [ ] # Liste de dictionnaires
Pour tout pattern h d'indice i de la liste_patterns_activité **faire**
  cluster_associé = liste_clusters[i]
  Si cluster_associé $\notin$ GA['noeuds'] **alors**
    GA['noeuds'].ajouter(cluster_associé)
  Fin Si
  arc= {} # Dictionnaire
  arc['id'] = (noeud_courant, cluster_associé)
  Si arc $\notin$ GA['arcs'] **alors**
    nouvel_arc = arc
    nouvel_arc['poids'] = 1
    nouvel_arc['etiquette'] = liste_etiquettes[i]
    GA['arcs'].ajouter(nouvel_arc)
  Sinon
    arc['poids'] = arc['poids'] +1
    arc['etiquette'] = arc['etiquette']+ liste_etiquettes[i]
    GA['arcs'].mise_à_jour(arc)
  Fin Si
  # Mise à jour de la valeur du noeud courant
  noeud_courant= cluster_associé
Fin Pour

return GA

Fin Fonction

3.4.2 Algorithme d'extraction des règles

Le clustering

Pour l'extraction des règles encodées, nous avons utilisé l'algorithme décrit section 2.4.1. Nous avons choisi d'exploiter l'algorithme k-means pour le clustering en faisant varier le nombre de clusters demandés. Cet algorithme consiste à partitionner les données en k groupes en minimisant la distance entre les données à l'intérieur de chaque partition. Le nombre de groupes ou clusters en lequel il faut partitionner les données est à la charge de l'opérateur humain. Comme il s'agit d'une étape critique du processus d'extraction des règles, nous avons choisi de faire varier k dans un intervalle de valeur [2, 20] (soit 18 valeurs différentes) et de générer un automate pour chaque valeur de k. Nous avons obtenu pour chaque grammaire 18 graphes-automates différents, un pour chaque valeur de k choisie. Le choix de la plage de valeur a été fixé en fonction du nombre de nœuds dans les grammaires étudiées. Au niveau technique, nous avons choisi d'utiliser la librairie Python scikit-learn dédiée au machine learning [Pedregosa *et al.*, 2011]. Pour chaque valeur de k et grâce à cette librairie, nous obtenons une liste d'entiers, de la taille de la liste de nos patterns d'activité cachée que nous étudions. Nous désignerons à partir d'ici la liste d'entier par "liste-cluster" et la liste des patterns d'activités par "liste-patterns". Ces deux listes sont de même longueur et puisque la seconde a été générée en fonction de la première, il existe une adéquation entre les deux indices. En effet, chaque entier de la liste-cluster, indique le numéro de cluster auquel est rattaché le pattern d'activité à la même position sur la liste-patterns.

Considérons deux exemples de listes : Une liste-patterns de 5 patterns : [h0, h1, h2, h3, h4] et une liste-cluster [0 3 2 2 0] issus d'un clustering utilisant k-means et pour lequel k=4. La lecture de ces deux listes consiste à dire que :
- les patterns h0 et h4 appartiennent au cluster 0
- le pattern h1 appartient au cluster 3
- les patterns h2 et h3 appartiennent au cluster 2

Grâce à cette lecture, il est possible d'associer chaque pattern à un cluster et donc un nœud dans le graphe-automate que nous souhaitons extraire. La figure 3.15 présente un exemple de génération d'automate utilisant les 5 patterns cités ci dessus.

La génération de l'automate

Nous avons commencé la génération de l'automate avec les éléments suivants à notre disposition : La liste-patterns et la liste-clusters. La liste-cluster, ne comporte que des entiers positifs. Nous avons donc choisi de commencer la génération de notre graphe-automate par l'ajout d'un nœud avec l'identifiant -1 comme point de départ (figure 3.15, label A). Pour faciliter l'analyse des

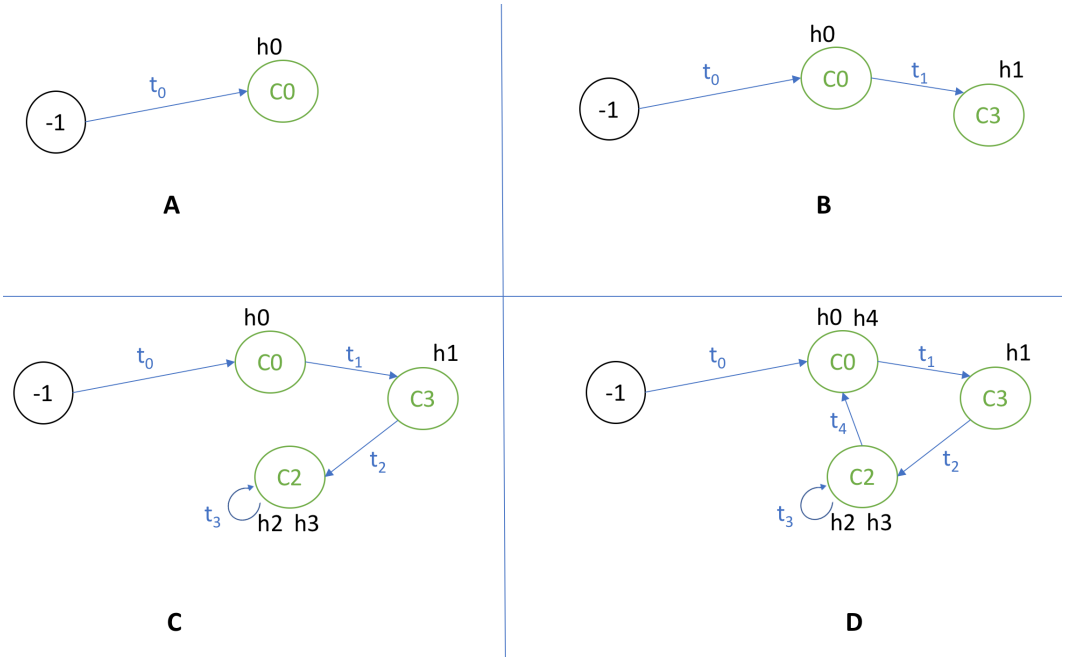


FIGURE 3.15 – Exemple de génération d'automate avec une liste-patterns [h0, h1, h2, h3, h4] et une liste-cluster [0 3 2 2 0].
A - Automate après le pas de temps t_0, création d'un nouveau nœud (état).
B - Automate après le pas de temps t_1, création d'un second nœud (état).
C - Automate après le pas de temps t_3 et t_4 qui appartiennent au même cluster, ce qui génère une boucle.
D - Automate après le pas de temps t_4 qui appartient à un cluster déjà visité, ce qui crée un arc entre le nœud courant (C2) et le nœud connu (C0).

graphes volumineux, mais aussi donner un point de repère pour la comparaison de différents graphe-automates, nous avons introduit cette idée, nouvelle par rapport aux approches connues. Les clusters ne pouvant avoir un identifiant négatif, nous avons choisit une valeur négative pour être sûr du caractère unique de cet identifiant.

Nous avons ensuite parcouru la liste-patterns et considéré chaque cluster associé : pour chaque premier pattern d'un nouveau cluster, nous ajoutons un nœud à notre graphe-automate (avec pour identifiant le numéro du cluster) et un arc orienté du nœud précédent vers le nouveau nœud (figure 3.15, label A, B, C). Si par contre, le pattern courant appartient à un cluster dont le nœud est déjà existant dans notre graphe-automate, alors nous ajoutons un arc dirigé entre le nœud précédent et le nœud courant (figure 3.15, label D, pas de temps t_4). Si deux patterns se suivant appartiennent au même cluster, une connexion récurrente est ajoutée sur le nœud représentant le cluster (figure 3.15, label C, pas de temps t_3).

Pour compléter cette génération, nous avons souhaité attribuer des étiquettes aux arcs, à l'image des étiquettes des arcs des grammaires de Reber et de ses variations, afin de mettre en lumière comment un RNN encode au niveau temporel les différents symboles qui lui sont présentés successivement. Pour cela, nous avons apporté deux ajouts à l'algorithme précédent : le premier consiste à modifier de manière incrémentale les étiquettes des arcs ajoutés. A chaque analyse d'un pattern de la liste-pattern, nous récupérons son identifiant alphanumérique à partir des données que nous avons au préalable préparées. La partie alphabétique désigne le symbole que le RNN traite à l'instant t et la partie numérique désigne le pas de temps. Une fois le cluster identifié et le nœud du graphe-automate créé (si besoin), il nous faut ajouter l'arc liant l'ancien nœud au nouveau, en lui associant cette fois ci son identifiant alphanumérique. La particularité du graphe-automate est que si un arc existe déjà entre deux nœud, il ne le créera pas une seconde fois. Ainsi dans une approche traditionnelle, il n'est pas possible de voir combien de fois une transition entre deux nœuds (états) a lieu, à moins de lui associer un poids ou une étiquette qui explicite cela. Nous avons donc choisi d'implémenter ces deux options : à chaque fois que l'algorithme de génération pointe sur un arc existant, l'étiquette de cet arc sera actualisée pour comporter le nouveau identifiant du pattern en cours et son poids sera incrémenter de 1. Nous obtenons à la fin de cette première évolution un graphe-automate, avec des étiquettes très longues, mais néanmoins qui montre l'organisation temporelle des patterns du RNN. La figure 3.17 présente l'exemple d'un automate généré sans étiquette et la figure 3.18 présente le même automate avec les étiquettes alphanumériques.

Le second ajout apporté à l'algorithme découle du premier : pour chaque

arc, nous avons réduit l'occurence de chaque symbole dans l'étiquette à une seule occurence, et nous avons enlevé la partie numérique. Nous souhaitions réduire la complexité d'analyse des graphes-automates et surtout pouvoir comparer les grammaires d'origines que nous avions utilisées (RG, ERG et CERG) avec les automates obtenus. Nous présentons et détaillons nos résultats dans la section suivante.

3.4.3 Application à la grammaire de REBER et à ses variations

Nous étudions dans cette section les patterns d'activité de la couche cachée de notre RNN-LSTM décrit précédemment.La figure 3.16 rappelle les grammaires RG et ERG utilisées dans cette section.

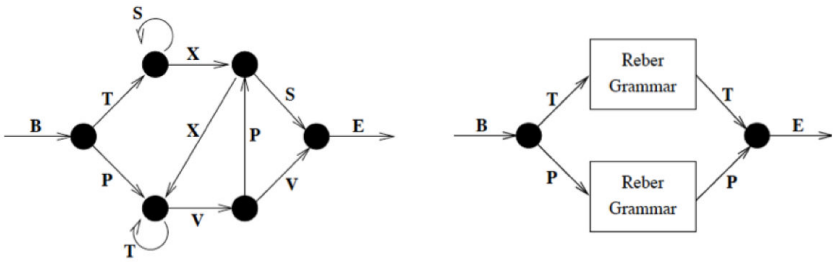


FIGURE 3.16 – La grammaire de Reber à gauche et la grammaire de Reber embarquée à droite

La grammaire de Reber (RG)

Nous avons commencé par réaliser un apprentissage sur 200 000 séquences. Ensuite, lors de la phase de test, sur les 10 simulations réalisées (à chaque fois sur 20 000 nouvelles séquences RG) nous avons récupéré les patterns d'activité de la couche cachée pour une seule simulation. Les résultats présentés dans les sections 3.17 à 3.19 représentent les graphes-automates extraits à partir de l'analyse des 50 premiers pas de temps lors de la prédiction qui portaient sur plusieurs occurrences de 2 séquences : BPVVE et BTXSE. En suivant les algorithmes décrits précédemment, nous obtenons le graphe-automate sans étiquette, représenté en figure 3.17, le graphe-automate avec les étiquettes longues (pour 15 clusters) représentés en figure 3.18 et le graphe-automate final en figure 3.19.

La comparaison du graphe-automate final avec la sous-partie de la grammaire de Reber responsable de l'encodage des 2 séquences montre de fortes similitudes entre les deux représentations. Soulignons ici que l'absence de boucles

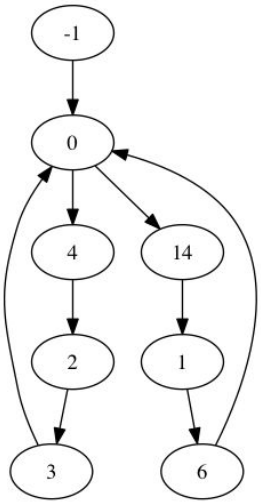


FIGURE 3.17 – Un graphe-automate sans étiquette obtenu avec un algorithme k-means pour le clustering, où k=15, et issu d'un RNN-LSTM ayant appris des séquences RG. Extraction réalisée sur les 50 premiers pas de temps.

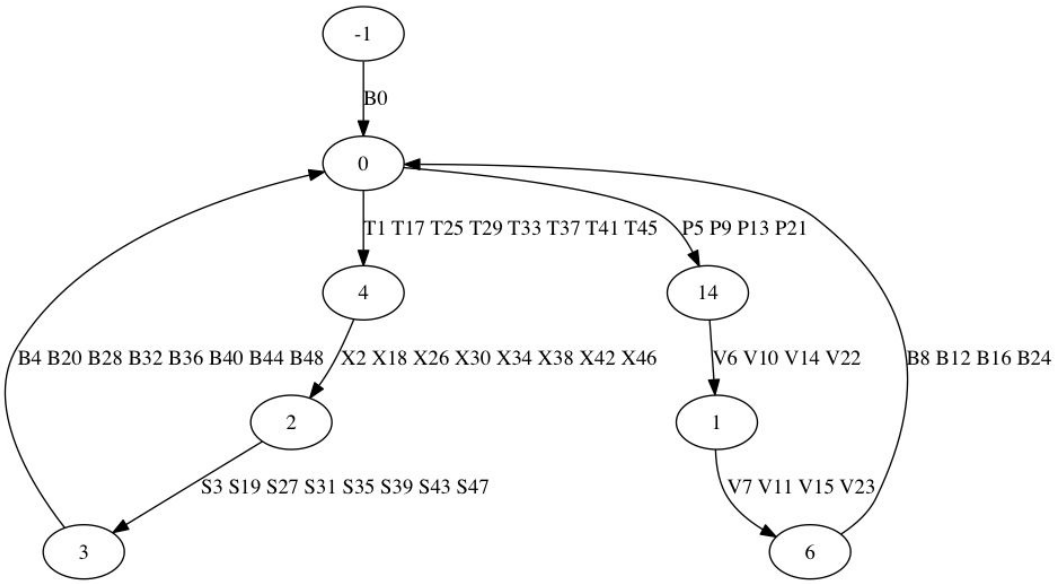


FIGURE 3.18 – Un graphe-automate avec étiquettes longues obtenu avec un algorithme k-means pour le clustering, où k=15 et issu d'un RNN-LSTM ayant appris des séquences RG. Extraction réalisée sur les 50 premiers pas de temps.

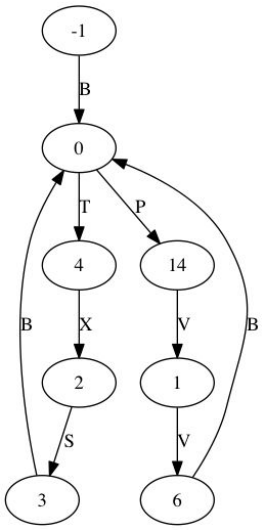


FIGURE 3.19 – Un graphe-automate final obtenu avec un algorithme k-means pour le clustering, où k=15 et issu d'un RNN-LSTM ayant appris des séquences RG. Extraction réalisée sur les 50 premiers pas de temps.

dans les graphes-automates résulte de l'absence de symboles répétés dans les séquences utilisées pour l'analyse. Nous avons ensuite voulu appliquer cela à des séquences plus longues et complexes : nous avons choisi d'appliquer cela aux séquences de la grammaire de Reber embarquée.

La grammaire de Reber Embarquée (ERG)

Un apprentissage sur 200 000 séquences ERG a été réalisé sur le même RNN-LSTM décrit ci dessus. Pour un rendu visuellement accessible, nous présentons l'automate extrait pour les 80 premiers pas de temps qui portaient sur plusieurs occurences des 4 séquences suivantes : BPBPVVEPE, BPBBTXSEPE, BPBPVVETE, BPBBTXSETE. Nous obtenons ainsi les 3 graphes-automates : le graphe-automate sans étiquette, représenté en figure 3.20, le graphe-automate avec les étiquettes longues (pour 19 clusters) représenté en figure 3.21 et le graphe-automate final en figure 3.22. L'étude du graphe-automate final obtenu pour la grammaire ERG, permet de retrouver les séquences utilisées lors de la phase de test de RNN-LSTM. On retrouve aussi par ailleurs les mêmes transitions que celle de la grammaire d'origine. Pour confirmer la validité de l'automate obtenu, nous avons procédé manuellement : nous avons généré des séquences que nous avons soumises à la grammaire d'origine ERG. La grammaire a validé les séquences dans 80% des cas.

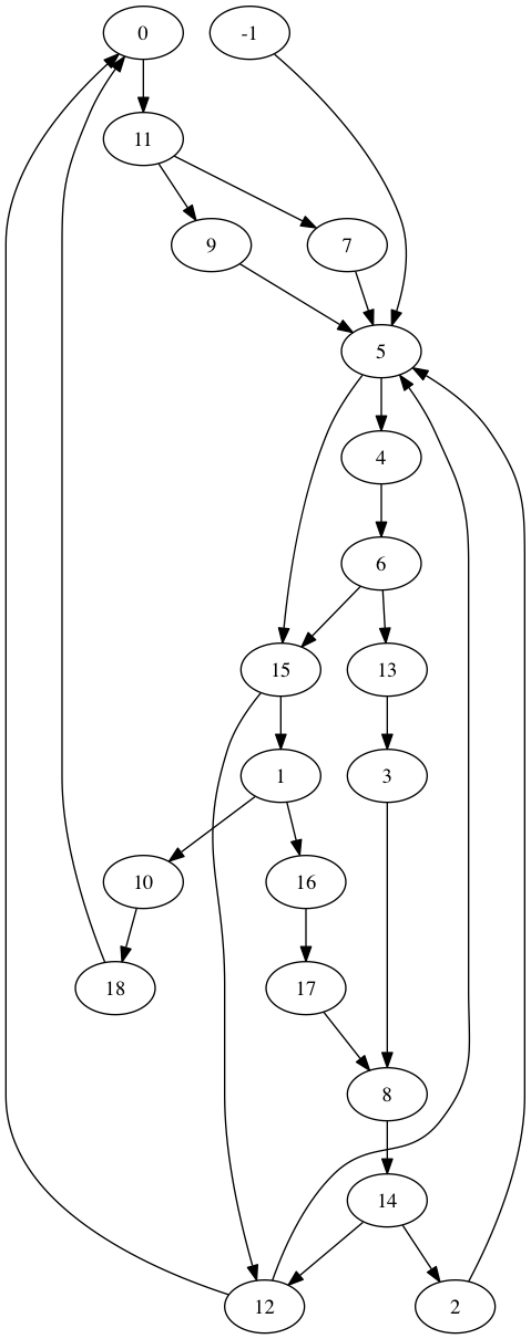


FIGURE 3.20 – Un graphe-automate sans étiquette obtenu avec un algorithme k-means pour le clustering, où k=19 et issu d'un RNN-LSTM ayant appris des séquences ERG. Extraction réalisée sur les 80 premiers pas de temps.

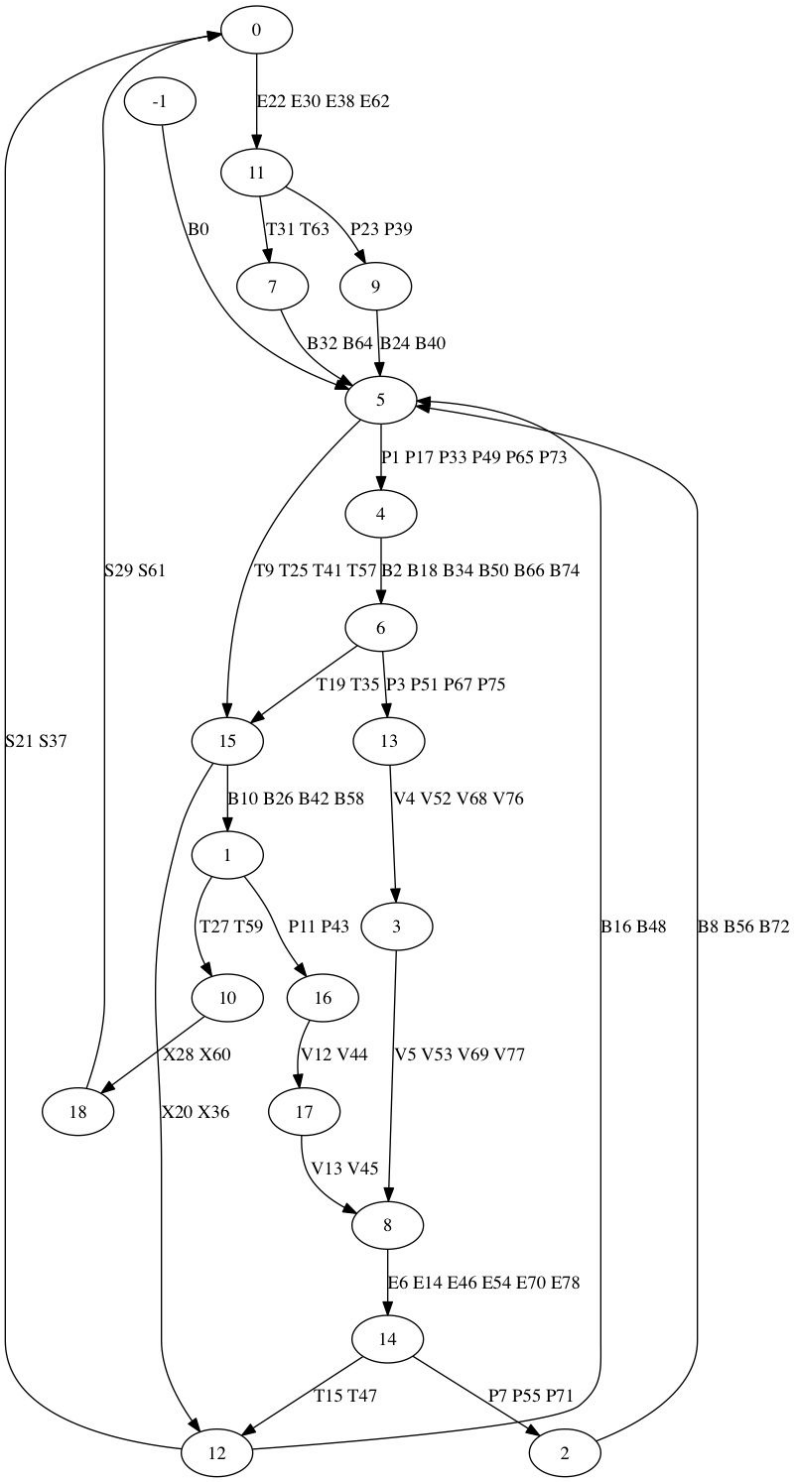


FIGURE 3.21 – Un graphe-automate avec étiquettes longues obtenu avec un algorithme k-means pour le clustering, où k=19 et issu d'un RNN-LSTM ayant appris des séquences ERG. Extraction réalisée sur les 80 premiers pas de temps.

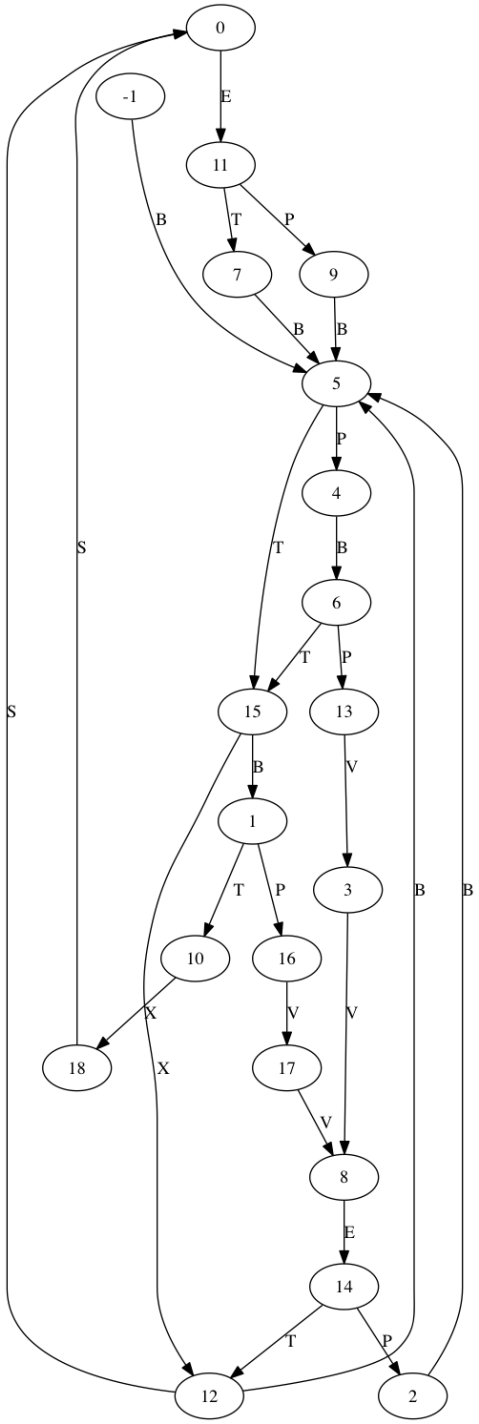


FIGURE 3.22 – Un graphe-automate final non-compressé obtenu avec un algorithme k-means pour le clustering, où k=19 et issu d'un RNN-LSTM ayant appris des séquences ERG. Extraction réalisée sur les 80 premiers pas de temps.

3.4.4 Application à une grammaire électrique (EL)

Dans cette section, nous allons aborder une des originalités de notre travail : les séquences de composants électriques. Jusqu'à présent les études expérimentales ont portée sur des grammaires connues mais artificielles. Nous allons, ici, appliquer notre nouvelle approche dans un nouveau contexte, industrielle et issue de données réelles. Nous avons souhaité montrer comment notre approche permet, en exploitant les séquences de composants électriques, de reconstruire en partie la base de données. Nous insistons surtout sur les règles métiers et les habitudes de représentations qui avaient été extraites via des échanges verbaux avec le client de l'entreprise Algo'Tech.

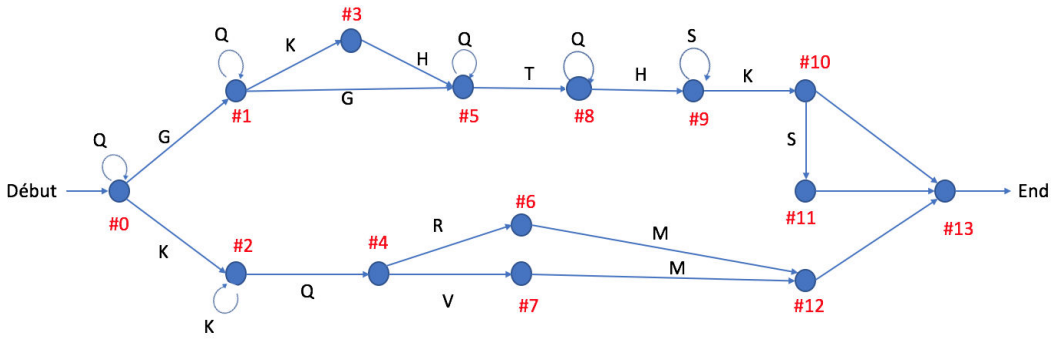


FIGURE 3.23 – Grammaire électrique issue de l'étude de 3 schémas électriques d'un même client

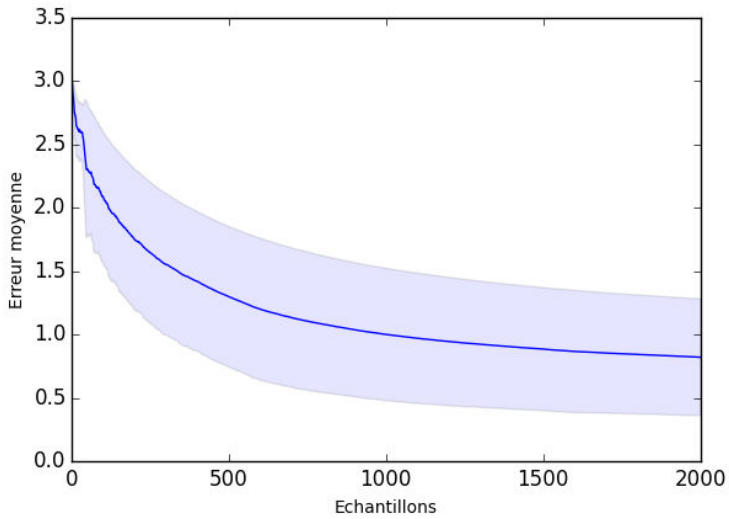


FIGURE 3.24 – Evolution de la moyenne de l'erreur lors de l'apprentissage du RNN-LSTM de 200 000 séquences issues de la grammaire électrique.

Nous présentons en figure 3.23 la grammaire d'un client que nous avons extrait manuellement par l'étude de schémas électriques. Nous avons, comme décrit précédemment dans le chapitre 1, mimer le comportement d'un expert Algotech en étudiant 3 schémas électriques : nous avons parcourus les fils de chaque schéma afin d'extraire les relations entre les composants électriques d'un folio à un autre. Ce travail chronophage, nous a ainsi permis d'extraire une grammaire pour chacun des schémas, que nous avons fusionné ensemble pour obtenir une grammaire électrique du client considéré. La validation de ce travail a nécessité la collaboration d'un expert Algo'Tech et entrainé plusieurs échanges concernant les symboles électriques, leur signification et les relations entre eux. La complexité et la charge de travail de ces travaux justifient notre choix premier d'utiliser des grammaires artificielles pour les étapes de notre approche décrites précédemment. Celle-ci est plus complexe que la grammaire de REBER. Elle comporte 15 nœuds et 25 transitions. A partir de cette grammaire, nous avons généré 200 000 séquences de composants électriques pour l'apprentissage et 20 000 pour le corpus de tests.

Nous avons fait apprendre au RNN-LSTM, et lors de la phase de test, nous avons enregistré l'ensemble des patterns d'activation de la couche cachée à chaque pas de temps. Pour 20 000 séquences, nous avions en moyenne 165 273 échantillons, soit autant de patterns d'activités. La figure 3.24 présente la courbe de moyenne d'erreur (les 2000 premières moyennes). Cette grammaire possédant 15 nœuds, nous avons réalisé un clustering k-mean pour des valeurs de k allant de [2 à 50]. Nous avons ensuite passé en revue les graphes-automates obtenus. Les figures 3.25, 3.26, 3.27 présente le graphe automate pour k= 41 car c'est celui qui visuellement se rapprochait le plus de la grammaire d'origine. Ces premiers résultats préliminaires montrent que le RNN-LSTM arrive à apprendre des séquences électriques et que l'extraction des règles est possible.

3.4.5 Discussion sur l'extraction de règles à partir de RNN-LSTM

Le travail d'extraction des règles présenté dans cette section fait partie des contributions majeures que nous avons réalisées. Dans le but de mettre en évidence les règles implicites dans des séquences apprises par les RNN, nous avons réussi à transposer une technique d'extraction jusque là réservée au SRN dans un nouveau contexte, celui des LSTM, proposant de ce fait une nouvelle méthode. Testée en premier lieu sur des grammaires artificielles que sont les grammaires de Reber et de Reber embarquée, nous avons pu transposer cette technique sur une problématique industrielle concrète, à savoir l'extraction des règles de connectivité électrique implicites à partir d'un RNN-LSTM ayant appris des séquences issues d'une grammaire électrique réelle élaborée à partir de l'étude de schémas électriques. Bien que notre approche ne soit qu'à ses débuts, le fait qu'elle puisse être appliquée à une problématique concrète laisse

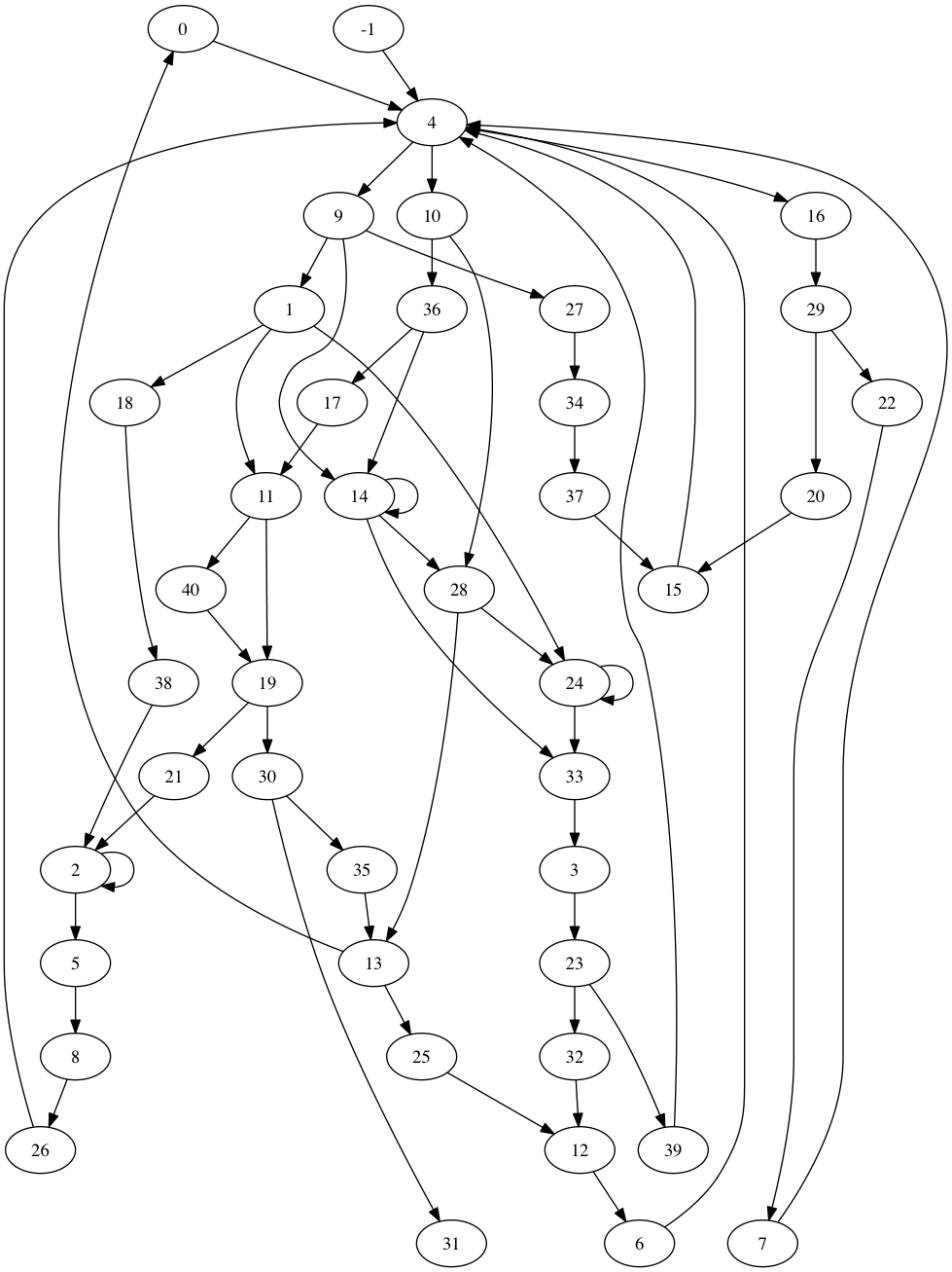


FIGURE 3.25 – Un graphe-automate sans étiquette obtenu avec un algorithme k-means pour le clustering, où k=41 et issu d'un RNN-LSTM ayant appris des séquences EL. Extraction réalisée sur les 79 premiers pas de temps.

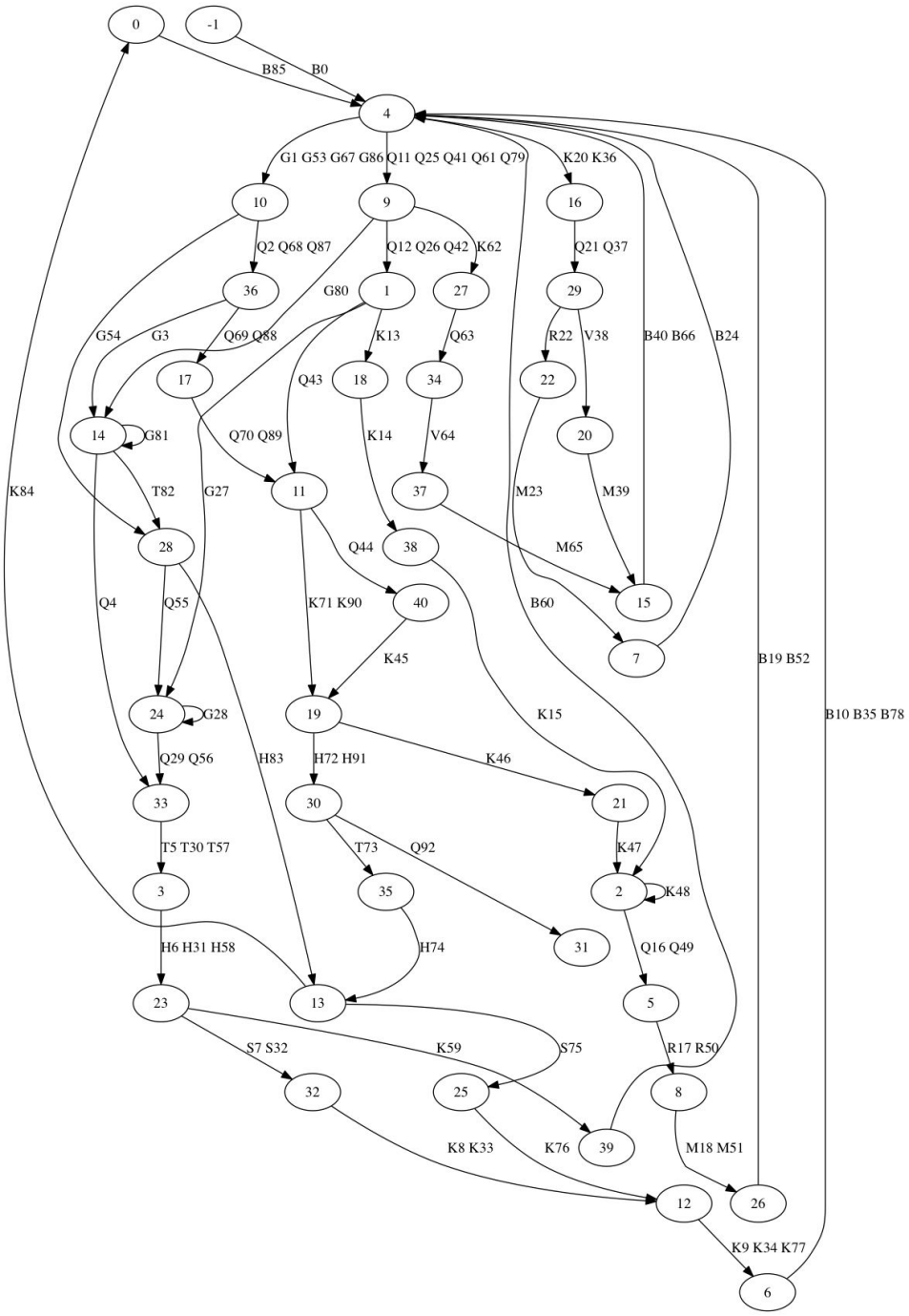


FIGURE 3.26 – Un graphe-automate avec étiquettes longues obtenu avec un algorithme k-means pour le clustering, où k=41 et issu d'un RNN-LSTM ayant appris des séquences EL. Extraction réalisée sur les 79 premiers pas de temps.

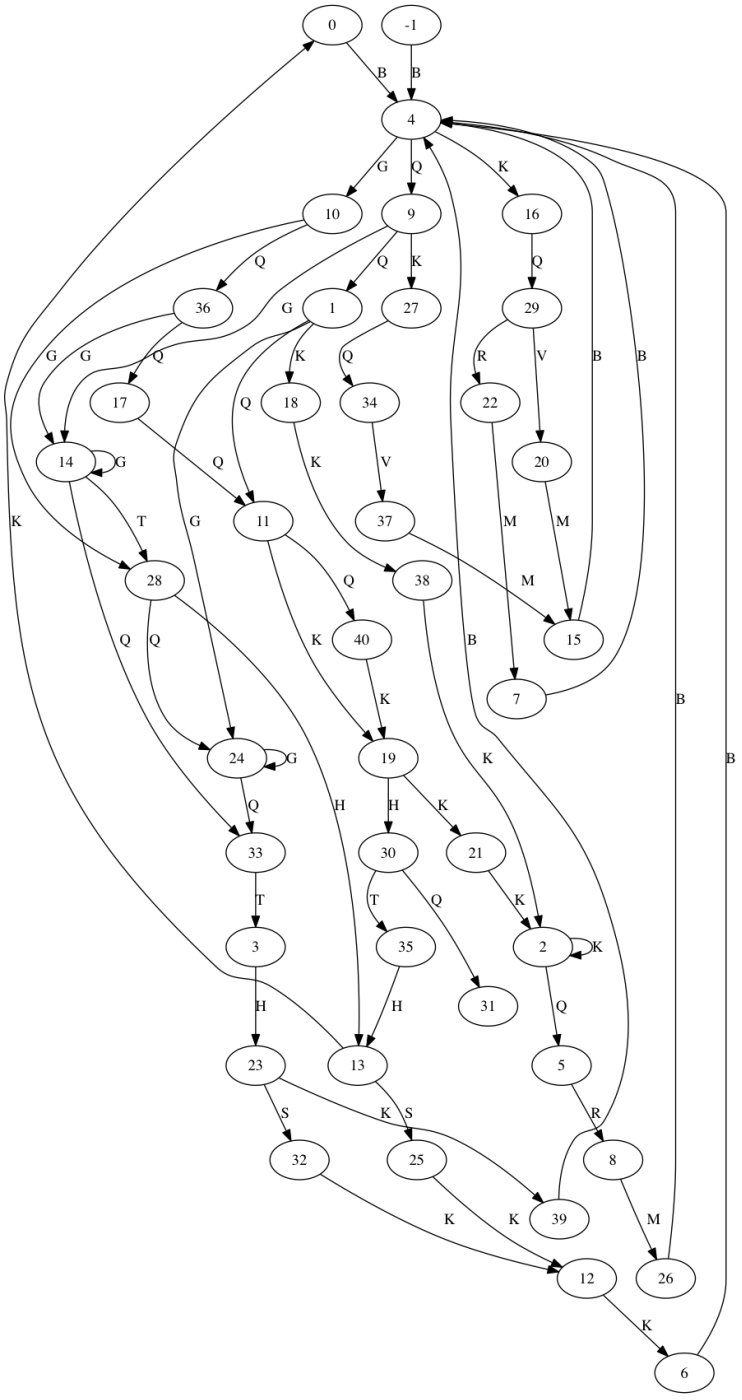


FIGURE 3.27 – Un graphe-automate final non-compressé obtenu avec un algorithme k-means pour le clustering, où k=41 et issu d'un RNN-LSTM ayant appris des séquences EL. Extraction réalisée sur les 79 premiers pas de temps.

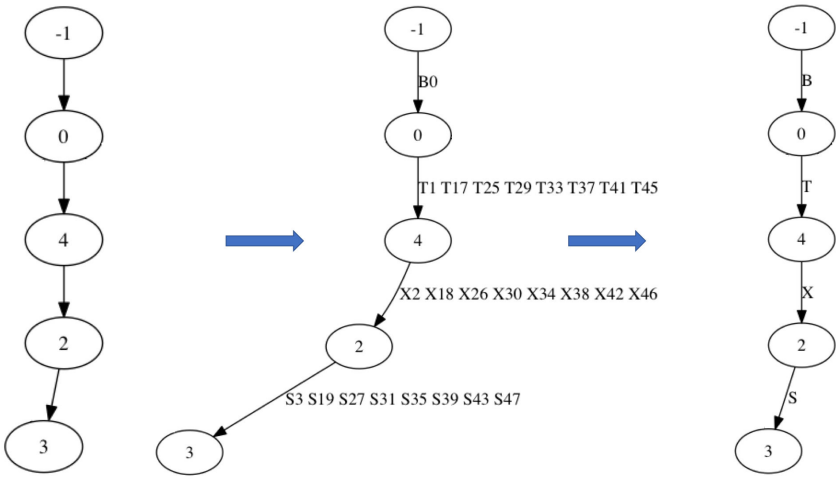


FIGURE 3.28 – Règles extraites à partir d'un RNN-LSTM ayant appris des séquences RG. Exemples issus, de gauche à droite, du graphe-automate sans étiquettes (figure 3.17), du graphe-automate doté d'étiquettes longues (figure 3.18) et du graphe-automate avec une étiquette simple (figure 3.19)

penser qu'elle peut donc être transposée dans les domaines techniques où l'on retrouve des séquences, ce qui offre des axes de développement intéressants et prometteurs.

L'algorithme que nous avons conçu au cours de ce travail de thèse correspond ainsi bien à nos attentes puisqu'il permet d'extraire des règles implicites en exploitant les patterns d'activité de la couche cachée d'un RNN-LSTM après la phase d'apprentissage de celui-ci. Grâce à cela, nous avons proposé 3 variantes de graphes-automates, proches des grammaires d'origine. La première, sans-étiquette, explicite l'agencement des clusters (états) de l'automate extrait. La seconde dotée d'étiquettes longues permet de suivre l'acheminement temporel des patterns et ainsi de lire l'automate. Les arcs de ce second graphe-automate comportant des poids, cela permet potentiellement d'analyser quelles sont les transitions (les règles de succession) les plus fortes. Dans le contexte de la question des RNN "boites noires", il semble important de souligner que cette représentation permet de mettre en lumière le fonctionnement interne d'un RNN-LSTM en explicitant la succession temporelle des patterns et leur prise en charge interne par le réseau de neurones récurrent. Enfin, le dernier graphe-automate ne comportant qu'une étiquette simple (un seul symbole) permet d'attribuer une étiquette unique à chaque arc. Il résume à lui seul notre approche puisqu'il propose une réprésentation explicite des connaissances implicites encodées par le RNN-LSTM. C'est la représentation qui se rapproche le plus des grammaires d'origines. La figure 3.29 présente un comparatif entre la grammaire de Reber et le graphe-automate extrait par notre

algorithme. Ce dernier ayant été réalisé pour 50 pas de temps, ne permet pas de représenter toute la grammaire mais présente de grandes similarités avec les règles de la grammaire d'origine.

Autre point qu'il nous faut souligner, l'insertion d'un nœud initial dans chacun de nos graphes-automates. Lorsqu'un graphe automate contient un faible nombre de cluster, il est simple de l'analyser. A contrario, lorsque le nombre est important, l'analyse peut s'avérer plus longue et énergivore (spécialement si l'on étudie le second type de graphe-automates avec des étiquettes longues). Pour faciliter cette étape, nous avons introduit un nœud doté de l'identifiant -1 (les clusters ne pouvant pas avoir un identifiant négatif dans notre approche) qui permet d'interpréter plus facilement les résultats en fournissant un point de départ à la génération des séquences et en offrant un même point de comparaison entre les différentes représentations. La figure 3.28 présente un exemple de règles extraites dans le cadre de la grammaire de Reber selon ces trois représentations avec le même point de départ, le nœud -1. A notre connaissance, de tels travaux d'extraction de règles à partir de LSTM dans le cas de langages réguliers n'avaient pas encore été réalisés.

Dans les cas présentés, lors de l'analyse des graphes-automates pour déterminer le nombre de cluster adéquat pour partionner l'espace d'état du RNN-LSTM, le nombre de graphes à tester (une soixantaine) bien qu'important est resté accessible à un opérateur humain. Cependant, nous sommes conscient que pour pouvoir garantir la transposition de ces travaux à d'autres domaines, des techniques plus automatiques de validation des graphes devraient être mise en place. La taille des graphes ne serait alors plus un problème potentiel. La taille peut aussi être réduite via des algorithmes de minimisation des graphes tel que l'algorithme de Hopcroft qui permet de calculer l'automate déterministe fini minimal, à partir d'un automate fini donné [Hopcroft, 1971]. Ce procédé, décrit comme un ingrédient à part entière du procédé d'extraction des règles par Jacobsson [2005], permettrait de comparer les graphes-automates obtenus plus facilement. Cela permettrait (éventuellement) aussi d'explorer la stabilité de la représentation interne des LSTM par exemple, si plusieurs paramètres de clustering permettent d'obtenir un même automate minimal.

L'un des points importants de notre approche actuelle est le choix du nombre de clusters k, que nous utilisons pour l'algorithme de clustering k-means. Dans les cas des grammaires RG, ERG et EL, au début de nos tests, le fait de connaitre la grammaire nous a aidé à fixer un intervalle de valeurs restreint pour faire varier le paramètre k (nombre de clusters). Mais cela n'a souvent pas été suffisant et nous avons dû réaliser des tests sur des intervalles plus larges ([2, 20] pour les grammaires RG et ERG et [2,50] pour la grammaire électrique). Pour trouver la valeur de k qui approxime le mieux la grammaire

d'origine nous avons dû manuellement passer en revue l'ensemble des graphes obtenus. En générant des séquences manuellement et en les soumettant aux grammaires d'origines, c'est ainsi que nous avons choisi les valeurs k=15 pour les figures 3.17, 3.18 et 3.19, et k=19 pour les figures 3.20, 3.21 et 3.22.

Nous sommes conscients que ce point représente l'une des faiblesses de notre approche. Le travail manuel de comparaison entre les graphes-automates obtenus et les grammaires d'origines afin de déterminer le paramètre k qui offre la meilleure représentation tend à devenir chronophage lorsque la plage de variation de k augmente. Sans compter qu'étant justement manuel, il ne peut être reproduit ou déclaré fiable. Le procédé actuel de sélection du paramètre k est donc en soit insatisfaisant. L'adoption d'autres techniques de clustering ou de méthodes plus automatisées pour le choix de ce paramètre k est donc un point incontournable dans la poursuite de nos travaux. L'exploration d'autres techniques de clustering permettrait ainsi, d'une part, de lever la contrainte de choix explicite (ce qui nous permettrait d'étendre notre approche à des domaines où la grammaire est inconnue, puisque ce paramètre serait alors fixé en fonction des données) et d'autre part, d'explorer l'adéquation entre notre approche et d'autre techniques de réseaux de neurones.

L'exploration pourrait aussi s'étendre à l'exploitation d'autres données du RNN-LSTM. Rappelons que le LSTM est un modèle connexionnistes qui possède une représentation explicite et interne du temps. L'une de ses caractéristiques principales est qu'il permet donc d'expliciter le temps. Par extension, il permet d'accéder explicitement à un ensemble de données tel que l'état interne de chaque cellule, le CEC, les valeurs des portes, etc. Cet ensemble de données représente ainsi un ensemble de travaux potentiels de recherches supplémentaires qui pourraient découler de ces travaux préliminaires.

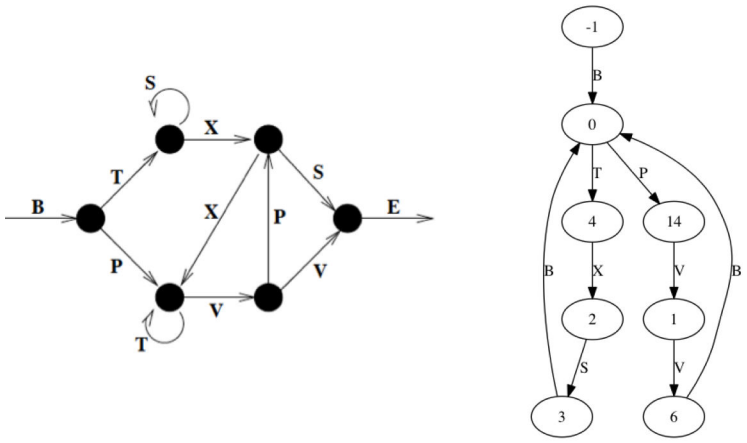


FIGURE 3.29 – Grammaire de Reber à droite et Graphe-automate extrait à gauche (pour 50 premiers pas de temps).

3.5 Discussion

A ce stade, nous considérons avoir répondu à la question initiale : à savoir comment extraire les règles implicites encodées dans des séquences porteuses de dépendances (sur le long et court terme) à partir d'un réseau de neurones récurrents ayant appris ces séquences.

La première partie de ce chapitre se concentre surtout sur l'explication et la mise en évidence de l'importance du choix du RNN pour l'apprentissage séquentiel. Nous avons exploité un premier RNN, le SRN en évaluant à la fois son apprentissage et sa capacité à généraliser sur des séquences issues de la grammaire de REBER. Nous avons ainsi montré qu'il s'agissait d'un RNN qui présente des faiblesses pour la rétention d'informations sur le long terme. En effet, l'information récente venant écraser celle plus ancienne, empêche de stocker un contexte sur de longues périodes de temps. Nous avons ensuite étudié les performances d'un second RNN, le RNN-LSTM, en terme d'apprentissage mais aussi de généralisation. Nous avons mis en évidence les bonnes performances de ce modèle, puisque après un apprentissage sur 200 000 séquences issues de la grammaire de Reber, le réseau a su reconnaitre 100% des séquences présentées lors de la phase de test, et cela sur 10 simulations de suite. Ces résultats encourageants ont aussi été reproduits avec des séquences issues de la grammaire de Reber embarquée, qui présentent plus d'ambiguïtés. Pour finir, nous avons testé ce modèle avec des flux continus de séquences ERG mises bout à bout. Sur 30 000 flux composés de 100 000 symboles successifs, le réseau a présenté 100% de prédictions correctes. Ces bonnes performances sont contre-balancées par l'apprentissage qui s'effectue durant les premiers 1000 pas de temps (figure 3.14) : la mise à jour du taux d'apprentissage tous les 100 pas

de temps, induit une diminution de la valeur de ce dernier jusqu'a atteindre une valeur quasi-nulle à partir de la 800ème mise à jour. Ce paramètre est donc à garder à l'esprit pour des implémentations futures afin d'arrêter l'apprentissage lorsque le réseau n'apprend plus.

En dehors des différences de performances, rappelons aussi la différence dans la représentation du temps dans le fonctionnement de ces deux RNN du fait de leur architecture. Le SRN possède une représentation interne et implicite du temps, alors que le LSTM possède une représentation explicite et interne du temps. Dans le premier cas, la représentation du passé est implicite mais fragile (puisque écrasé par un passé plus récent). A contrario, dans le second cas la représentation du passé est explicite et maintenue, lorsque nécessaire, sur plusieurs pas de temps sans altération. Les données sont donc plus accessible et préservée dans le cas des RNN-LSTM que dans le cas du SRN. L'ensemble de ces résultats nous ont ainsi amené à considérer le RNN-LSTM au lieu du SRN pour l'étude et l'extraction des règles implicites encodées durant l'apprentissage.

L'approche d'extraction des règles proposée dans la littérature était majoritairement réalisée sur les SRN. En la transposant sur des LSTM, nous avons proposé une nouvelle approche permettant d'exploiter un modèle connexionniste représentant le temps de manière explicite et en interne. Nous avons ainsi, dans la seconde partie de ce chapitre, mis en évidence la représentation implicite de la grammaire encodée par le RNN-LSTM. Nous avons utilisé l'algorithme du k-means utilisant un apprentissage non-supervisé par dessus le RNN-LSTM. En exploitant les patterns des unités de la couche cachée, cet algorithme réalise le partitionnement de l'espace d'état des unités selon un paramètre k choisit par l'opérateur humain. En d'autre terme, si l'on assimile le RNN-LSTM à un expert en apprentissage séquentiel, l'algorithme k-mean peut être assimilé à une entité observatrice extérieure qui analyse et interprète le comportement de l'expert afin d'expliciter les connaissances implicites de cet expert. Les études expérimentales portant sur cette nouvelle approche ont porté sur des séquences différentes de trois grammaires différentes : la grammaire de Reber, sa variante la grammaire de Reber embarquée, ainsi qu'une grammaire électrique représentant un cas industriel concret. A chacune de ses études, nous avons réussi à extraire trois graphes-automates proches des grammaires d'origine, représentant le fonctionnement interne du RNN. Chacune de ces représentations apporte un élément de réponse à la question des RNN "boites noires". Le premier graphe-automate informe sur l'agencement des états et les transitions existantes entre eux. Le second graphe-automate met en évidence l'acheminement temporel des patterns entre les différents états détectés. A partir du nœud initial -1, il est possible de suivre l'encodage interne du RNN-LSTM des patterns qu'il génère. Enfin, le troisième graphe-automate,

résumé des deux précédents, est la représentation la plus proche des grammaires d'origine.

Nos travaux ont donc permis de réaliser le passage de séquences dotées de règles implicites à des grammaires (graphes-automates) présentant explicitement ces mêmes règles. Bien que préliminaires, ces travaux ont débouché sur des résultats riches en terme d'informations, de potentiel d'exploration au niveau technique (comme nous l'avons discuté précédemment section 3.4.5), et de potentiel d'exploration dans d'autres domaines.

Conclusion générale

Table des matières

Résumé des travaux

Dans ces travaux, nous nous sommes penché sur la problématique d'extraction de l'expertise et surtout des connaissances implicites. Ces notions étant transversales à de nombreux domaines, nous avons puisé nos connaissances dans différents axes d'études : la cognition, le knowledge management et l'IA avec les réseaux de neurones récurrents. Nous avons ciblé le domaine électrique et notamment les connaissances encodées dans les séquences de composants électriques car il s'agit d'une part du domaine d'application de notre partenaire, l'entreprise Algo'tech, et d'autre part, c'est un domaine où l'organisation des connaissances rejoint celle chez l'humain, à savoir explicite et implicite. Plus précisément, nous nous sommes intéressé à la question de l'apprentissage séquentiel des RNN. Nous avons souhaité mettre en évidence la possibilité d'extraire les règles encodées implicitement par un RNN-LSTM après son apprentissage de séquences afin d'amener dans le domaine de l'explicite ce qui a été appris implicitement.

C'est dans cette optique, et suite aux interrogations soulevées en introduction, que nous avons produit les contributions suivantes :

Notre première contribution applicative, au niveau du premier chapitre, a été de proposer un procédé d'extraction et de représentation des connaissances

techniques dans les schémas électriques. Nous avons développé une approche qui vise à construire une représentation selon différents niveaux d'abstractions à partir de ces documents techniques. Pour cela nous avons mené tout un travail de fouille et de préparation des données en vue de mettre en évidence la structure hiérarchique (des concepts métiers aux composants électriques), la structure modulaire (organisation en concepts métiers) et la structure séquentielle (séquences de concepts, séquences de folios, séquences de composants électriques) des connaissances contenues dans les schémas électriques. Nous avons ainsi mis en évidence l'existence des règles métiers implicites contenues dans les séquences de composants électriques, que nous avons ciblées pour la suite de notre étude.

Pour notre seconde contribution nous avons étudié et comparé deux modèles de réseaux de neurones pour l'apprentissage séquentiel : un SRN et un RNN-LSTM. Nous nous sommes essentiellement intéressé à leurs performances en termes d'apprentissage séquentiel et de généralisation. Nous avons montré que le SRN, malgré son apprentissage et ses performances, est très sensible aux variations de paramètres. Nous avons pour ce modèle proposé 36 configurations différentes dans lesquelles nous faisions varier le nombre de neurones de la couche cachée, le taux d'apprentissage et le momentum. Nous avons ainsi mis en exergue qu'un bon apprentissage du SRN ne s'accompagne pas systématiquement de bonnes capacités de généralisation et inversement. Et cela bien que nous nous soyons appuyé sur des données issues de la littérature. Par ailleurs, nous avons aussi mis en évidence le phénomène d'écrasement du passé lointain à la faveur du passé récent qui a lieu dans le SRN, ce qui représente une grande limite à son utilisation pour l'étude de séquences longues. A contrario, les LSTM ont montré de bonnes performances pour l'apprentissage et la généralisation des séquences RG (dotées d'un début et d'une fin explicites), des séquences ERG (où le réseau doit apprendre à détecter les ambiguïtés entre les symboles de début et de fin situés à plusieurs endroits dans une même séquence), mais aussi de flux CERG (succession de séquences ERG mises bout à bout). Au niveau des dépendances séquentielles, le fait que le RNN-LSTM ait réussi à gérer les flux de symboles continus prouve sa solidité et son adéquation pour la gestion de séquences longues : le passé est conservé sur de long laps de temps sans être altéré ou oublié. Le modèle RNN-LSTM est par conséquent plus performant pour l'apprentissage de séquences complexes porteuses de contexte, local et étendu dans le temps, que le modèle SRN.

Notre troisième contribution, que l'on qualifiera de contribution algorithmique, a été de proposer un procédé d'extraction des règles encodées par un RNN-LSTM après apprentissage, procédé qui n'était jusque là appliqué que sur des SRN. Nous avons ainsi décrit les différentes étapes d'une nouvelle méthode qui visent à l'extraction et la représentation de manière explicite des règles im-

plicites présentes dans des séquences à partir d'unités LSTM. Cette approche a été réalisée sur la grammaire RG et la grammaire ERG en présentant des premiers résultats encourageants. Enfin nous avons testé notre approche sur une grammaire électrique, obtenue à partir de l'analyse de schémas électriques d'un client, ce qui a permis de montrer que nos travaux sont aussi transposables à des cas concrets. A chaque fois nous avons utilisé un algorithme de clustering k-mean, en complément du RNN-LSTM, afin de partitionner l'espace d'état des unités LSTM de la couche cachée. Ce partionnement, étape essentielle du processus d'extraction, a nécessité plusieurs itérations, selon différents paramètres k, afin d'obtenir 3 graphes-automates proches des grammaires d'origine : Un premier sans-étiquettes explicitant l'agencement des clusters et les transitions entre eux, un second doté d'étiquettes longues permettant de suivre l'acheminement temporel des patterns entre les clusters dans l'espace d'états et un troisième, qui attribue une étiquette simple (un symbole) à chaque transition et qui se rapproche le plus des grammaires d'origines.

En résumé, à travers nos trois approches, nous avons proposé une solution générique d'extraction des connaissances implicites à partir de schémas électriques.

Résultats

3.5.1 Au niveau industriel

Au niveau industriel, pour l'entreprise Algo'Tech, deux résultats ont été obtenus et centralisés dans le prototype réalisé et présenté dans le chapitre 1. Le premier résultat concerne l'extraction des concepts métiers. Nous avons élaboré une méthode qui permet d'analyser les schémas électriques d'un client et d'en proposer trois représentations synthétiques qui tiennent sur une page (plutôt que des dizaines ou centaines de folios) : un premier graphe représentant les folios électriques et les liaisons entre eux, un second graphe qui explicite l'organisation d'un schéma folio par folio en fonction des concepts métiers (présents dans ce schéma) et un troisième graphe qui représente l'agencement des concepts métier dans un schéma. Un quatrième graphe permet de représenter les concepts métiers présents dans l'ensemble des schémas d'un client et offre ainsi une vue synthétique du métier d'un client.

Le second résultat est l'extraction de l'empreinte implicite par des techniques de data mining, ou autrement dit, la mise en évidence d'habitudes de travail au niveau des séquences de composants électriques. Nous avons, lors de cette analyse, comparé des folios d'un même client mais réalisés à des années différentes. Nous avons ainsi observé dans plusieurs cas que des folios nommés

différemment comportaient la même séquence de composants électriques. Rappelons que dans le contexte du premier chapitre, les séquences sont constituées de composants électriques présents dans les folios et ordonnés selon leurs coordonnées (x, y) croissantes. En se basant sur ces travaux, nous avons mis en évidence les habitudes implicites de travail (en rapport avec les séquences de composants électriques) sous la forme d'arbres de clusters et de dendogrammes.

C'est ainsi l'alliance de ces deux approches, extraction de concepts métiers et extraction d'empreinte implicite présente dans les séquences, que nous avons réussi à extraire les corrélations en termes de concepts et de séquences de composants électriques d'un client donné.

Le prototype réalisé en Python est actuellement en test chez Algo'Tech. Au niveau humain, nous avons souhaité évaluer l'impact chez les collaborateurs d'Algo'Tech face à un outil d'extraction des connaissances et des règles de câblage électrique. Malgré l'accueil positif du prototype au sein de l'entreprise, il est trop tôt actuellement pour évaluer s'il y aura un impact significatif dans la méthode de travail des collaborateurs Algo'Tech.

3.5.2 Au niveau machine learning

Au niveau machine learning, nos travaux ont permis de montrer qu'il était possible d'extraire les règles apprises par les LSTM. L'extraction des règles n'était jusque là réalisée, à notre connaissance, que sur des SRN. En adaptant cette méthode sur les patterns d'activité de la couche cachée du RNN-LSTM, nous avons établi que les sorties des cellules des unités LSTM, étaient aussi porteuses d'une information relative à la représentation implicite des régularités encodées par le réseau. Nous avons testé cette nouvelle approche dans trois contextes différents et montré que l'on obtenait toujours des résultats intéressants. Dans les 3 contextes (RG, ERG et EL), nous avons réussi à expliciter les règles apprises implicitement par les LSTM et cela sous trois représentations distinctes. Parmi elles, le second graphe-automate, représentation dotée d'étiquettes longues sur ses transitions, permet d'observer le comportement interne d'un RNN en suivant la prise en charge temporelle des patterns dans l'espace d'états.

Ce dernier point permet de rebondir sur la question de l'interprétabilité des réseaux de neurones récurrents en général et de leur fonctionnement, à savoir la question des RNN "boites noires". Si l'on reprend une métaphore faite précédemment, l'utilisation d'un algorithme de clustering par dessus le RNN-LSTM peut être assimilé à observer un expert en cours de "raisonnement" et interpréter son comportement pour en extraire des règles. Ces dernières étant

les règles implicites qui ont une incidence sur le "raisonnement" de l'expert. L'agencement temporel des patterns à partir du graphe-automate doté d'étiquettes longues, permet d'observer ce "raisonnement" en tant que succession d'éléments répartis dans l'espace d'état. Sans avoir la prétention d'avoir résolu la question, nous pensons que ce point mérite néanmoins d'être souligné pour la richesse d'information qu'il permet et le potentiel de perspectives de recherche qu'il engendre. Nous y reviendrons par la suite.

Une autre originalité de notre travail réside dans l'application de celui-ci dans un nouveau domaine, celui des séquences de composants électriques. Nous avons extrait et créé manuellement une grammaire de connectivité de composants électriques à partir de schémas électriques représentant une installation réelle. Cette grammaire a ensuite été utilisée pour générer des séquences de composants électriques. Soumises à notre approche d'extraction de règles, ces séquences ont permis l'extraction d'un automate, certes complexe mais présentant des similarités avec la grammaire d'origine. Ce travail préliminaire montre qu'il existe tout un potentiel qui découle de cette approche dans le domaine de l'extraction de l'expertise à partir des schémas électriques, et plus globalement à partir de schémas techniques (contenant des séquences).

Enfin il semble important de souligner que le fait de se focaliser sur l'étude des séquences dans ces documents plutôt que l'ensemble des documents, permet de se libérer de la nécessité d'avoir de gros volumes (en terme de documents) pour pouvoir appliquer des techniques populaires de deep learning ou de data mining. En se focalisant sur les séquences, il est possible d'avoir des résultats exploitables pour un industriel et cela juste avec une centaine de schémas techniques.

3.5.3 Au niveau gestion des connaissances implicites

L'un des apports majeurs de nos travaux est la mise en évidence de la pertinence d'une approche centrée autour de l'humain pour l'extraction des connaissances implicites (i.e l'expertise) contenues dans les schémas techniques. En se basant sur l'organisation cognitive des connaissances d'un individu (explicite et implicite ; séquentielle et hiérarchique), deux approches d'extraction des connaissances ont été proposées. Dans le chapitre 1, nous avons mis en évidence l'existence d'une structure hiérarchique et séquentielle dans les schémas électriques en nous inspirant du processus de planification chez l'humain et de la mémoire sémantique. Dans la seconde partie du chapitre 2, nous avons aussi montré, grâce à notre connaissance de la mémoire implicite, qu'il était possible de modéliser l'apprentissage implicite en utilisant un RNN-LSTM. L'exploitation des séquences par les réseaux de neurones récurrents alliée à une approche d'extraction des règles nous a permis d'expliciter les règles implicites présentes

dans les séquences. Dans une dimension plus globale, nous avons montré qu'une approche orientée autour du processus humain de génération des connaissances permet de mieux comprendre et d'exploiter les données et cela même dans un domaine très technique.

Perspectives

3.5.4 Amélioration de l'approche actuelle

Concernant le prototype du logiciel que nous avons réalisé, il s'agit d'une preuve de concept pour laquelle nous nous sommes essentiellement focalisé sur le coté fonctionnel. Ayant montré l'apport de nos travaux dans le domaine du schéma électrique, nous pensons que la réalisation d'une version plus aboutie dotée d'une interface ergonomique permettrait une meilleure prise en main en interne à Algo'Tech. Par ailleurs, cela permettrait aussi de favoriser la réalisation de tests sur d'autres clients travaillant dans le même domaine que ceux déjà analysés et d'autres clients travaillant dans des domaines différents. Cela permettrait de valider l'approche d'extraction des concepts métiers dans différents domaines techniques.

Au niveau des réseaux de neurones et de la comparaison entre le SRN et le RNN-LSTM, nous souhaiterions interroger la notion de performance en ajoutant d'autres mesures de performances. Dans chacune de nos implémentations, nous nous sommes basés sur des paramètres issus de la littérature pour l'architecture des réseaux et pour déterminer les critères d'évaluation des performances qui leur sont applicables. Si nous modifions ces critères, aurions-nous des grammaires différentes ? Nous pourrions déterminer s'il existe une dépendance entre les critères des performances des réseaux et les grammaires extraites et si oui ses modalités. Le pourcentage de prédiction correcte selon la grammaire d'origine est un critère que nous mesurons actuellement. En testant cela sur des séquences grammaticales non bruitées, il serait possible de comparer la justesse de la prédiction de chaque réseau. Nous tenons particulièrement à souligner les limites des deux approches implémentées pour les SRN et le RNN-LSTM qui représentent autant de perspectives de travail dans un axe purement apprentissage. La compréhension des critères de succès d'une bonne prédiction (0.3 pour le SRN et 0.49 pour le RNN-LSTM) ainsi que l'étude plus approfondie des performances et de l'erreur des réseaux lors de l'apprentissage nous semblent des axes essentiels de poursuite de nos travaux.

Cette question des paramètres desquels dépend la grammaire, se pose aussi dans le procédé d'extraction des règles que nous avons établi. Dans notre méthode, nous avons choisi de débuter notre exploration de l'espace d'état d'un RNN-LSTM en exploitant et analysant les sorties des cellules des unités LSTM

car nous souhaitions transposer l'approche appliquée au SRN sur les LSTM. Nous avons ainsi mis en évidence la représentation implicite du réseau en accédant à sa mémoire implicite. Le LSTM étant un modèle connexionniste qui représente le temps de manière interne mais surtout explicite, il permet, par son architecture, d'accéder à un ensemble d'information explicitement : les valeurs du CEC à chaque pas de temps, les valeurs d'activité des portes, les valeurs des entrées en cellules, ainsi que les informations relatives aux pas de temps précédents (CEC et sortie des cellules). Appliquer notre approche à chacune de ses données, mais aussi à l'alliance des données entre elles, permettrait potentiellement, d'obtenir des grammaires différentes. Ces représentations différentes nous permettrait, par exemple, de mettre en évidence les connaissances localisées au niveau du CEC. Soulignons que cette question des paramètres desquels dépend la grammaire se pose aussi si nous appliquons notre approche sur des variations des LSTM standards, tels les GRU ou les LSTM dotés de connexions peepholes.

Dans le cadre de notre approche d'extraction des règles à partir du RNN-LSTM, une prochaine étape de validation serait d'appliquer cet algorithme sur des séquences de composants électriques de différents clients et de soumettre la grammaire extraite non seulement aux experts Algo'Tech, mais aussi aux experts clients. Nous pourrions comparer nos résultats obtenus aux bases de données contenant les métiers des clients et valider ou ajuster ainsi nos travaux. Elargir à d'autres clients d'autres domaines, permettrait aussi de confirmer la transposabilité de notre approche face à d'autres domaines et surtout sa solidité.

Dans une dimension plus technique, nous allons rappeler ici les axes d'amélioration discutés précédemment pour l'approche d'extraction des règles (section 3.4.5). Nous pensons que l'ajout d'un algorithme de minimisation des automates permettrait d'explorer de nouvelles possibilités pour l'étude des règles extraites. Comparer les représentations compressées des automates permettrait d'évaluer quelles sont les règles qui ont tendance à rester présentes, et surtout de réaliser un comparatif entre les différentes techniques de clustering de l'espace d'états. Cela nous permettrait aussi de réduire les automates en terme de complexité : nous obtiendrons un automate réduit pour chaque automate que nous avons. Il serait possible de comparer les différentes versions réduites et voir s'il existe une stabilité dans la représentation interne du RNN-LSTM. L'ajout d'un algorithme d'analyse qui soumettrait des séquences générés à partir d'un automate à la grammaire d'origine, permettrait de discerner plus facilement les automates (et par extension les paramètres k du k-mean par exemple) qui sont adéquats ou non pour le clustering. Enfin, concernant ce dernier, nous souhaiterions tester des approches différentes plus proches des données. Nous pensons notamment à l'algorithme Growing When

Required (GWR) [Marsland *et al.*, 2002] et l'algorithme Growing Neural Gas [Fritzke, 1995] qui consistent à utiliser des approches incrémentales où les réseaux s'adaptent aux données. En théorie, ces approches permettent de réaliser un clustering de manière automatique sans qu'un opérateur humain fasse de choix explicite. L'automate extrait serait plus proche et fidèle aux données. Nous pourrions alors comparer nos automates actuels à ceux résultant de ces approches, qui correspondent plus à un fonctionnement cognitif (i.e sans l'intervention d'un intervenant extérieur).

Dans nos travaux nous avons mis en place l'attribution de poids aux transitions dans le second graphe-automate. Nous pensons que de telles données permettraient d'étudier les variations de comportement dans les habitudes d'une entité (société ou personne) en comparant différents graphes-automates réalisés à différentes périodes par exemple. Dans les contextes où la grammaire est inconnue, cette information serait probablement pertinente pour l'analyse des transitions entre les états.

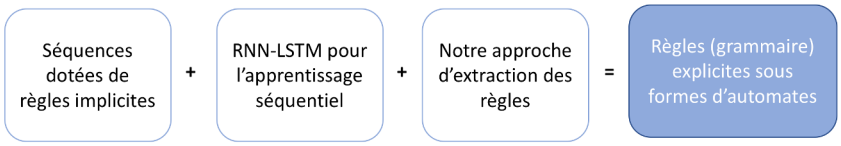
Dans les contextes de séquences très longues, il arrive souvent qu'un évènement ne se répète que sur des pas de temps tellement longs qu'une analyse humaine soit compliquée. Nous pensons qu'appliquer notre approche sur de telles séquences, et notamment exploiter le graphe-automate doté d'étiquettes longues permettrait de mettre en évidence les transitions (les règles implicites) qui n'ont lieu que très rarement. Cette mesure de l'occurence de certaines règles, rejoint l'idée abordée précédemment ci-dessus (i.e concernant les grammaires inconnues) car cela permettrait d'apporter des éléments d'analyses supplémentaires.

Enfin, une dernière perspective au niveau technique qui nous intéresse est l'ajout de couches cachées supplémentaires et l'extraction des automates pour chaque couche. En effet, dans nos travaux nous avons réalisé un RNN-LSTM d'une seule couche cachée, majoritairement en raison de notre souhait de comparer les LSTM et des neurones artificiels standards. Chung *et al.* [2016] a montré, lors de traitement de phrases en anglais par un RNN-LSTM doté de 3 couches cachées, que chaque couche encodait un niveau d'informations différent : la première couche apprenant chaque lettre (et espace), la seconde couche apprenant les mots et la troisième couche apprenant des portions de phrases. Appliqué à notre contexte de séquences électriques, serait-il possible d'extraire une grammaire de connectivité de composants électriques pour la première couche, une grammaire de connectivité des folios pour la seconde couche et une autre liée à des portions d'installations réelles pour la dernière couche ? Par ailleurs, nous avons choisi d'appliquer un RNN pour le clustering par dessus le RNN-LSTM. Mais il aurait été aussi envisageable d'en ajouter un troisième et d'observer un processus d'extraction pour ce réseau aussi. Le

nombre de couche et le nombre d'algorithme que nous pouvons ajouter pour l'analyse sont deux questions potentielles à explorer.

3.5.5 Application à d'autres domaines

Nos travaux actuels peuvent se résumer à l'équation suivante :



En faisant varier les séquences nous pourrions appliquer théoriquement cette équation dans différents domaines. Dans notre domaine d'application, bien que nous ayons travaillé sur des séquences issues de grammaires artificielles ou issues de schémas électriques de clients, nous étions toujours contraint par un cadre déterminé avec des contraintes et une combinatoire limitée. Si nous transposions nos travaux, dans des domaines où les cadres sont moins figés et dans lesquelles les codes implicites ne sont pas connus, comme chez l'humain par exemple, quelles grammaires obtiendrons-nous ? Ces dernières seraient-elles spécifique à un individu, ou à un groupe d'individu ? Dans le domaine de suivi d'activité notamment, si les séquences sont celles des déplacements d'une personne chez elle sur toute une journée (plusieurs laps de temps), pourrions-nous en suivant notre approche, extraire une grammaire des déplacements de cet individu (par mois, par jour, par semaine par exemple) ? et pourrions-nous observer l'évolution de celle-ci (apparition de nouveau nœuds, de nouvelles transitions, renforcement de certaines par l'analyse des poids des transitions et fragilisation d'autres, etc.) ? Dans le cas des personnes âgées, si cela était possible, cela permettrait potentiellement, non seulement de suivre l'activité de ces personnes et de s'assurer régulièrement qu'elles ont un bon rythme de vie (sorties en extérieur, déplacement en cuisine, courses, etc.), mais cela permettrait théoriquement aussi d'étudier les évolutions des habitudes des individus lors de l'apparition de certaines pathologies, notamment celles neurodégénératives par exemple.

Dans le domaine de l'interprétabilité des RNN, nous avions en introduction cité le cas de la DARPA. Nous pensons qu'en appliquant notre approche dans un domaine dont la grammaire est connue, nous pourrions grâce à l'automate extrait (et surtout à la représentation avec des étiquettes longues) déterminer les paramètres qui fournissent une grammaire qui approxime le mieux la grammaire d'origine, ce qui permet dans un premier temps d'observer le

comportement du RNN dans ce cas précis. Mais ensuite en faisant varier les paramètres (architecture du réseau, nombre d'unités, nombre de couches, etc.), nous pourrions observer les variations de comportement et surtout l'impact au niveau de la représentation implicite des règles. Cela pourrait s'avérer utile dans le cas des algorithmes de Deep Learning, où certes, plus il y a de couches plus l'algorithme est puissant, mais il devient aussi de plus en plus obscur et inaccessible. Une telle approche permettrait potentiellement de visualiser les successions d'éléments qui induisent les résultats de ces algorithmes. Décortiquer le "raisonnement" des algorithmes et apporter plus d'explications aux décisions prises par les algorithmes de Deep Learning permettraient, entre autre, d'améliorer la confiance des utilisateurs dans ces outils et peut-être limiter les dérives liées à leurs utilisations.

Annexe A

Manuel utilisateur prototype

Extraction des connaissances

Manuel d'utilisation

Algo'Tech Informatique - Technopole Izarbel - Hôtel d'entreprises
64210 Bidart
Tel : 05.59.01.59.60 - Fax : 05.59.01.59.69
Email : infos@algotech.fr - Web : www.algotech-informatique.com

Table des matières

I- Application à installer en amont du projet

Graphviz est un outil de visualisation et de génération des graphes.

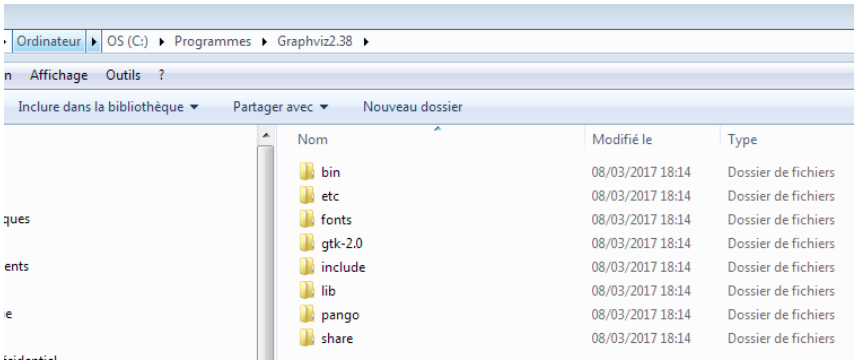
Son absence bloque la génération des graphes lors de l'étape 3, mais n'est pas préjudiciable à l'étape 1 du projet, soit l'analyse.

Il est cependant conseillé de réaliser les étapes ci-dessous pour une bonne exécution du projet.

1. Graphviz

- Décompresser le fichier« KnoweldgeExtraction_Install.zip »
- Aller dans le répertoire « outils »
- Double cliquer sur « graphviz-2.38.msi »
- Procéder à l'installation en cliquant sur « Suivant » jusqu'à là fin.
- Si l'application c'est bien installé, un nouveau répertoire est crée au lien suivant :

C:\Program Files\Graphviz2.38

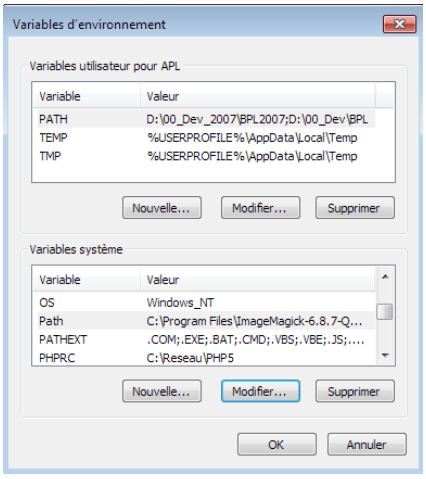


2. Modifier la variable d'environnement PATH

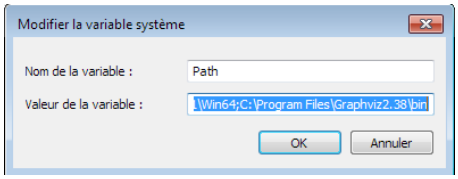
Notre application « KnowledgeExtraction.exe » lance un appel à l'exécutable dot.exe du répertoire C:\Program Files\Graphviz2.38\bin.

Il faut donc modifier la variable d'environnement PATH afin d'inclure ce nouveau chemin.

- Cliquer sur le bouton rond « Démarrer »
- Positionner la souris sur Ordinateur et cliquer droit
- Cliquer sur « Propriétés »
- Dans la fenêtre « Système » qui s'ouvre (aussi localiser à Panneau de configuration\Système et sécurité\Système), sur la colonne gauche, cliquer sur « Paramètres systèmes avancés »
- Dans la fenêtre « Propriétés système », cliquer sur « Variables d'environnement » afin d'ouvrir une nouvelle fenêtre « Variables d'environnement »



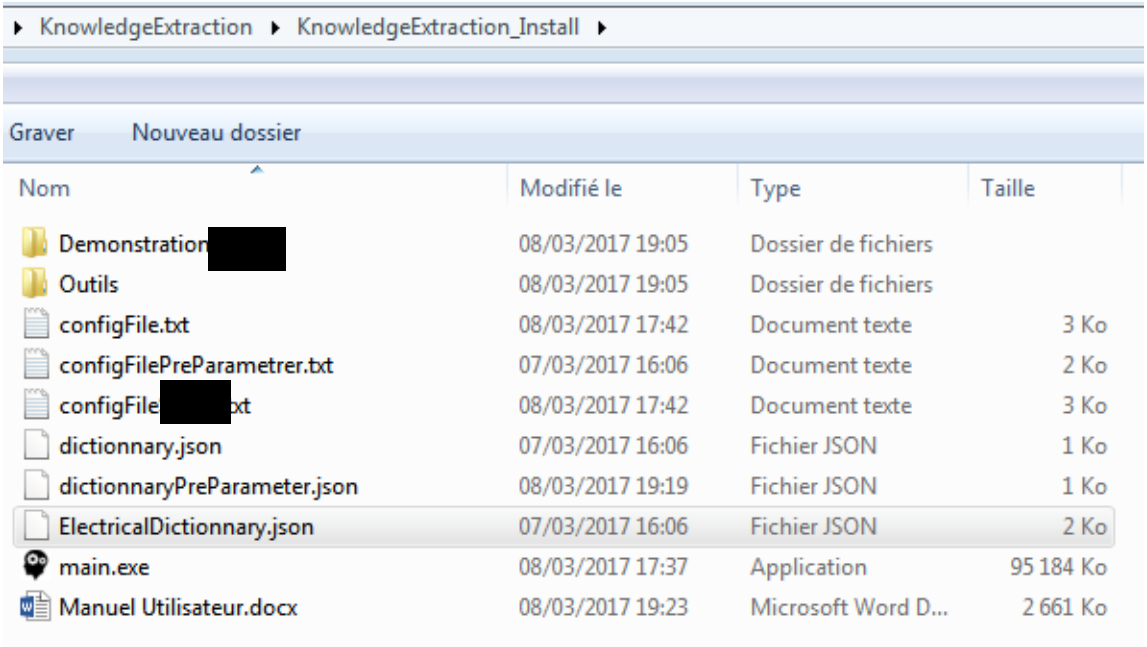
- Dans la section du bas nommé « Variables systèmes », rechercher la variable « Path », se positionner dessus puis cliquer sur le bouton « Modifier »
- Dans la popup « Modifier la variable système » qui s'ouvre, au niveau du champ « Valeur de la variable », se positionner à la fin et ajouter : « **;**C:\Program Files\Graphviz2.38\bin »
- Attention à ne pas oublier le point virgule avant le nouveau chemin.



- Cliquer sur OK, fermez tout et redémarrer votre ordinateur pour qu'il prenne en compte la modification.

II- Presentation du dossier de installation

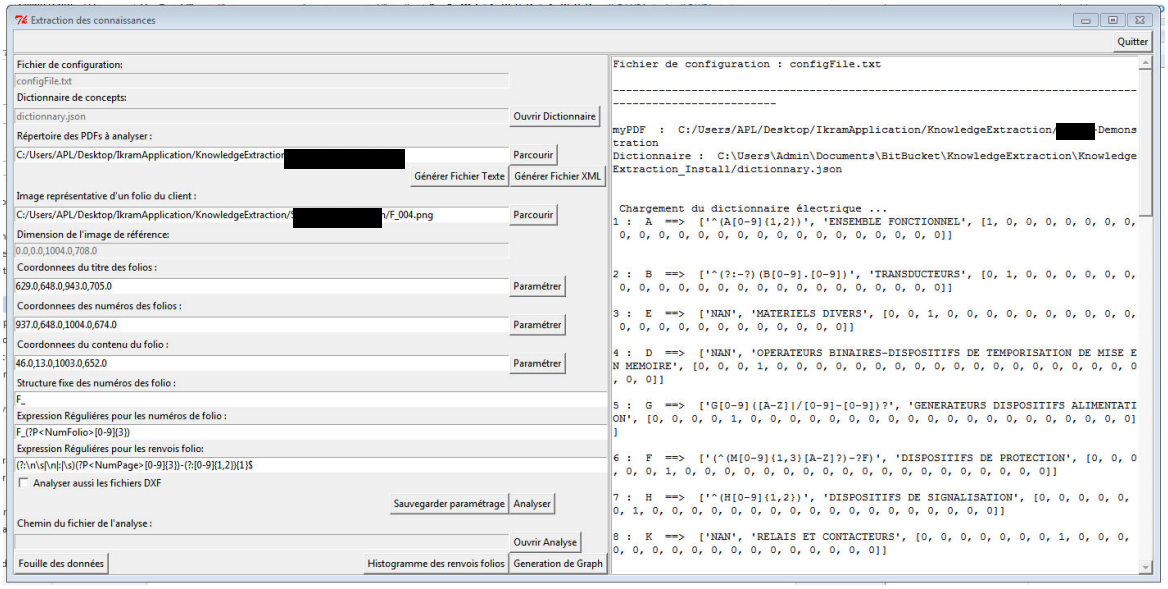
Une fois le fichier« KnoweldgeExtraction_Install.zip » décompressé, on obtient le répertoire suivant :



Dans ce dossier, il y a 4 types de fichiers qui sont important :

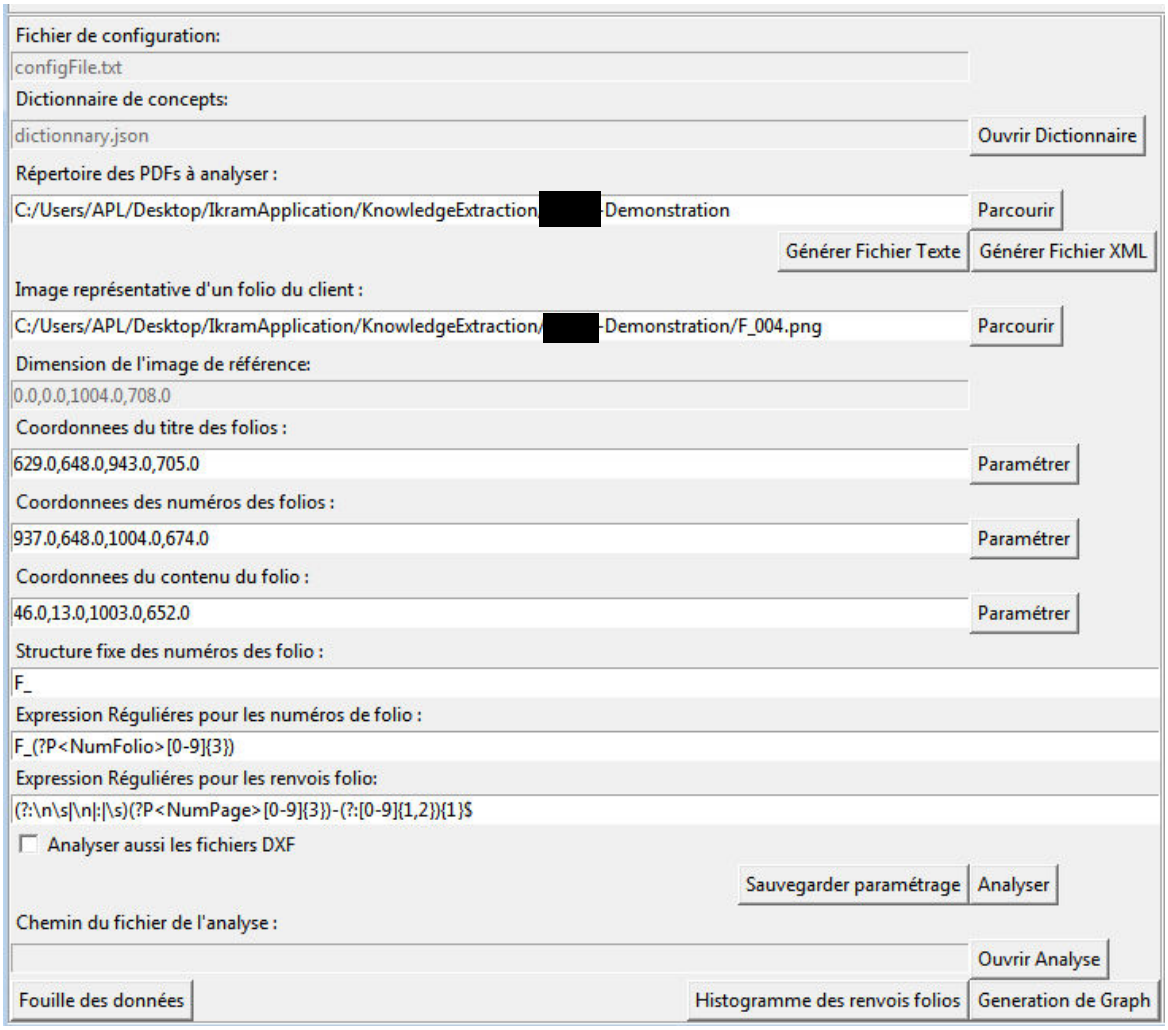
1) **Le répertoire « Démonstration_____ »** contient un exemple d'analyse sur 3 fichiers _____
2) **Le répertoire « Outil »** contient le programme Graphviz à installer (voir section II-Application à installer en amont du projet)
3) **Le fichier de paramétrage « configFile.txt »** : l'utilisateur peut le consulter mais il est déconseillé de le modifier directement. Il est nécessaire pour le fonctionnement de l'application. IL est possible de faire des sauvegardes en ajoutant un préfixe ou suffixe pour un client, mais seul le fichier nommé exactement « configFile.txt » qui sera pris en compte.
4) **Le fichier de paramétrage « confiFilePreParametrer.txt »** : est un fichier générique pour démarrer un nouveau client.
5) **Le fichier de paramétrage « configFile_____.txt »** : est un fichier de paramétrage _____ pour l'exemple.
6) **Le fichier « dictionnary.json »** : contient les concepts qui sont extrait pour un nouveau client, ce fichier est pré-rempli pour l'exemple _____ .
7) **Le fichier « dictionnaryPreParameter.json »** : est le fichier générique qui contient les concepts qui seront extrait pour un nouveau client. Ce fichier sera modifié à chaque client. Il ne contient par défaut qu'une paire d'accolade.
8) **L'application « main.exe »** est le programme central à paramétrer et lancer.
9) **ManuelUtilisateur.docx** est le manuel utilisateur ici présent.

III- Présentation générale de l'interface



A. Partie Gauche

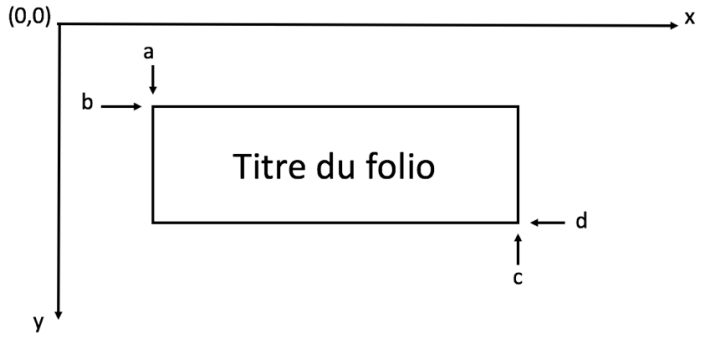
La partie gauche de l'application contient l'ensemble des champs à renseigner pour permettre à l'analyse d'avoir lieu.



3. Les champs

- **« Fichier de configuration »** rappel le nom du fichier de configuration utilisé pour charger et sauvegarder les informations. Son nom est fixé dans le code. Il est donc conseillé d'avoir un fichier par client (configFileNom du Client.txt) qu'il faudra renommé «configFile.txt » à chaque fois que l'on souhaitera basculer sur ce client là.
- **« Répertoire des PDFs à analyser »** contient le chemin du PDF (ou du répertoire contenant les pdfs à traiter) qui sera traité. Ces fichiers peuvent être n'importe où sur le disque, il faut juste veiller à ce que le chemin soit toujours d'actualité et sans espace.
- **« Image représentative d'un folio du client** » contient le chemin vers une image d'un folio du client. Cette image servira pour le paramétrage et est essentielle.

- **« Coordonnées du titre des folios »** contient le rectangle de coordonnées du titre sur les folios:
    - Les coordonnées sont de la forme « a, b, c, d »
        - **a** : coordonnée horizontale du point en haut à gauche du rectangle
        - **b** : coordonnée verticale du point en haut à gauche du rectangle
        - **c** : coordonnée horizontale du point en bas à droite du rectangle
        - **d** : coordonnée verticale du point en bas à droite du rectangle



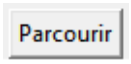
Ces coordonnées sont déduites par le processus de sélection des données sur l'image de référence lors de la phase de paramétrage. Cela sera expliqué ultérieurement (section V.A-Etape Préalable à l'utilisation à chaque nouveau client)

- **« Coordonnées des numéros des folios »** contient le rectangle de coordonnées du numéro de folio sur les folios
- **« Coordonnées du contenu des folios »** contient le rectangle de coordonnées de la partie schéma sur les folios
- **« Structure fixe des numéros de folios »** permet d'expliciter quelle est la partie fixe d'un numéro de folio
- **« Expression régulières pour les numéros de folio »** désignent l'expression générique qui va permettre de rechercher les numéros de folio dans chaque folio
- **« Expression régulières pour les renvois folio »** désignent l'expression générique qui va permettre de rechercher les renvois folio dans chaque folio
- **« Analyser aussi les fichiers DXF »** : Case à cocher qui permet d'inclure ou non les fichiers DXF dans l'analyse

Remarque :
Les expressions régulières constituent un système très puissant et très rapide pour faire des recherches dans des chaînes de caractères (des phrases, par exemple). Dans notre cas, l'application extrait le texte des PDFs afin de pouvoir leur appliquer les expressions régulières
Les expressions régulières seront détaillé ultérieurement (section V.A.2.Identifier les repères, les lister et établir leurs expressions régulières)

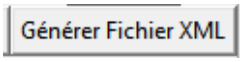
- **« Chemin du fichier généré »** contiendra le chemin et le nom du fichier généré à la fin de l'analyse.

4. Les boutons

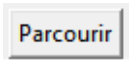


Ce bouton situé au niveau du champs « Répertoire des PDFs à analyser » permet de localiser le fichier PDF à traiter

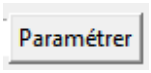
Ce bouton permet de générer un fichier texte contenant le texte extrait du PDF. Le fichier généré porte le même nom que le PDF d'origine et sera généré dans le même répertoire que lui



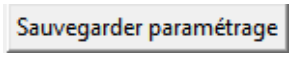
Ce bouton permet de générer un fichier XML contenant le texte extrait du PDF
Le fichier généré porte le même nom que le PDF d'origine et sera généré dans le même répertoire que lui



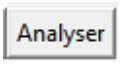
Ce bouton situé au niveau du champs « Image représentative d'un folio d'un client» permet de localiser l'image faite à partir d'un folio d'un client et qui représente l'organisation classique d'un folio du client



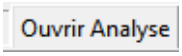
Ce bouton permet d'ouvrir une fenêtre supplémentaire qui affiche l'image représentative, sur laquelle il est possible de localiser l'information à extraire. Chaque bouton permet de récupérer l'information relative au champ à coté duquel il est placé.



Ce bouton permet la sauvegarde des paramètres dans le fichier configFile.txt si l'utilisateur souhaite les conserver.

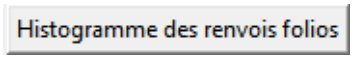


Ce bouton permet de lancer l'analyse du fichier et le processus d'extraction des connaissances. Un fichier Excel est créé à l'issue de cette étape.
A chaque lancement d'une nouvelle analyse, il vaut mieux s'assurer que le fichier excel a été fermé ou renommer, car l'application écrase le fichier du jour.

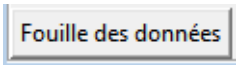


Une fois l'analyse réalisée, ce bouton permet l'ouverture du fichier excel créé

Ce bouton permet de lancer la génération des graphes, spécifiques à chaque schéma, mais aussi globaux, offrant ainsi une autre représentation des schémas.



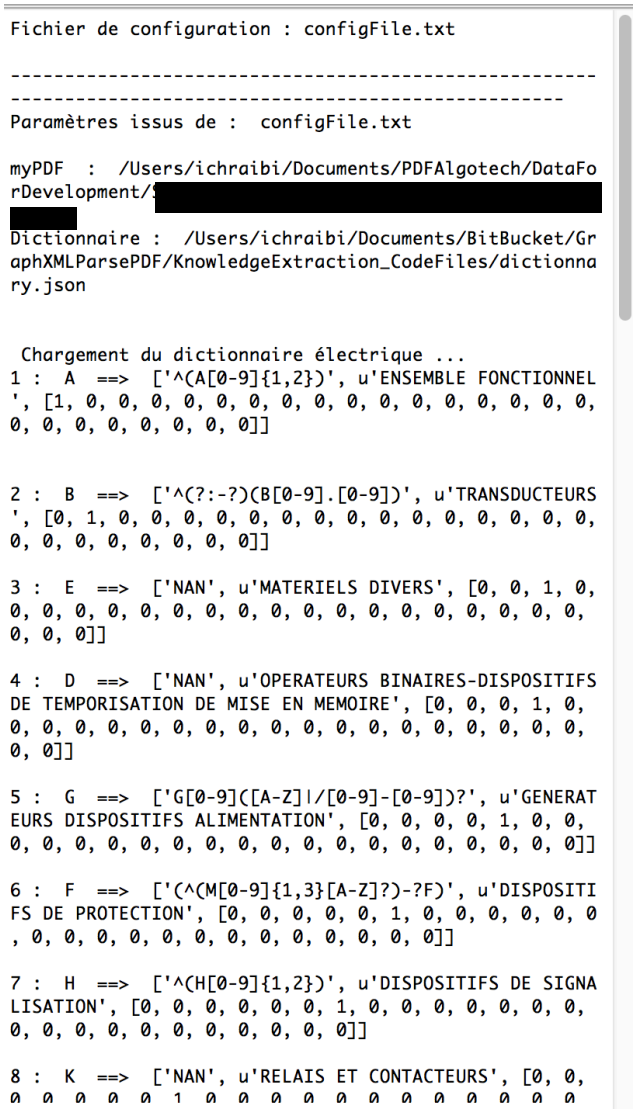
Ce bouton permet de générer un histogramme des renvois folios par concepts.



Ce bouton permet de lancer la fouille des données sur l'ensemble des données extraites des PDFs

B. Partie Droite

Ce bouton permet de quitter l'application



Le champ textuel ci-dessus, affiche les sorties standards et erreur de l'application. Cette partie récapitule les paramètres à chaque nouvelle analyse lancée et ainsi permet de garder une trace des actions réalisés lors d'une session.

Les erreurs affichées contiennent souvent des données techniques permettant une correction plus rapide, de ce fait, elles peuvent sembler peu intelligibles.

IV- Etape 1 : Paramétrage de l'application

A. Etape préalable à l'utilisation de l'application à chaque nouveau client :

1. Création d'une image représentative

Pour permettre à l'application de savoir où rechercher les informations à extraire, il faut lui donner les coordonnées des données à extraire.

Ce paramétrage nécessite d'ouvrir les schémas des clients, et de trouver un folio « représentatif » du client. Ce folio doit contenir au niveau du cartouche un titre, un numéro de folio, et une partie circuit.

L'utilisateur peut alors faire une capture d'écran du folio et enregistrer l'image au format png avec les PDFs du client.

2. Identifier les repères, les lister et établir leurs expressions régulières

Lors de la phase de fouille des données, l'application extrait les repères des composants électriques sur l'ensemble des PDF.

Pour cela, il faut lui passer en paramètre les expressions régulières qu'elle doit rechercher : les différents masques des différents composants électriques qu'elle doit extraire.

L'utilisateur doit donc au préalable, sélectionner un PDF ou plusieurs (cela est laissé à l'appréciation de l'utilisateur) parmi ceux du client représentatif et lister les repères qu'il rencontre.

Dans le fichier ElectricalDictionnary.json, sont recensé les 25 catégories de composants électriques.

Les repères doivent être regroupés selon ces 25 catégories lorsqu'elles sont présentes afin de faciliter l'analyse des similitudes.

A partir de ces suites de repères, l'utilisateur peut alors en déduire les expressions régulières adéquates et compléter le fichier de configuration.

Voici un exemple de paramétrage des repères dans le cas _____ pour les moteurs :

Etape 1 : recensé tous les repères sur un ou plusieurs Pdfs

"M" : ["MOTEURS"],

M1V M1L M51 M52 M53 M54 M70D M70R M71 M72 M73 M77

Etape 2 : recensé les structures

Lettre+chiffre+lettre

Lettre+Chiffre

Etape 3 : construire une expression régulière adéquate : "^(M[A-Z]?[0-9]{1,3}[A-Z]?)"

Indication sur l'expression régulière de _____ :	
^M	^ : désigne que la chaine débute par un M M : partie fixe de la chaine recherchée
[A-Z]?	[A-Z] : Il peut y avoir une lettre entre A et Z ?: désigne que la lettre peut être présente ou absente (il s'agit d'une option) ex : MR ou M
[0-9] {1,3}	Un chiffre entre 0 et 9 répété 1 à 3 fois Ex : 1, 12, 123
[A-Z]?	[A-Z] : Il peut y avoir une lettre entre A et Z ?: désigne que la lettre peut être présente ou absente (il s'agit d'une option)

B. Fichier de configuration : configFile.txt

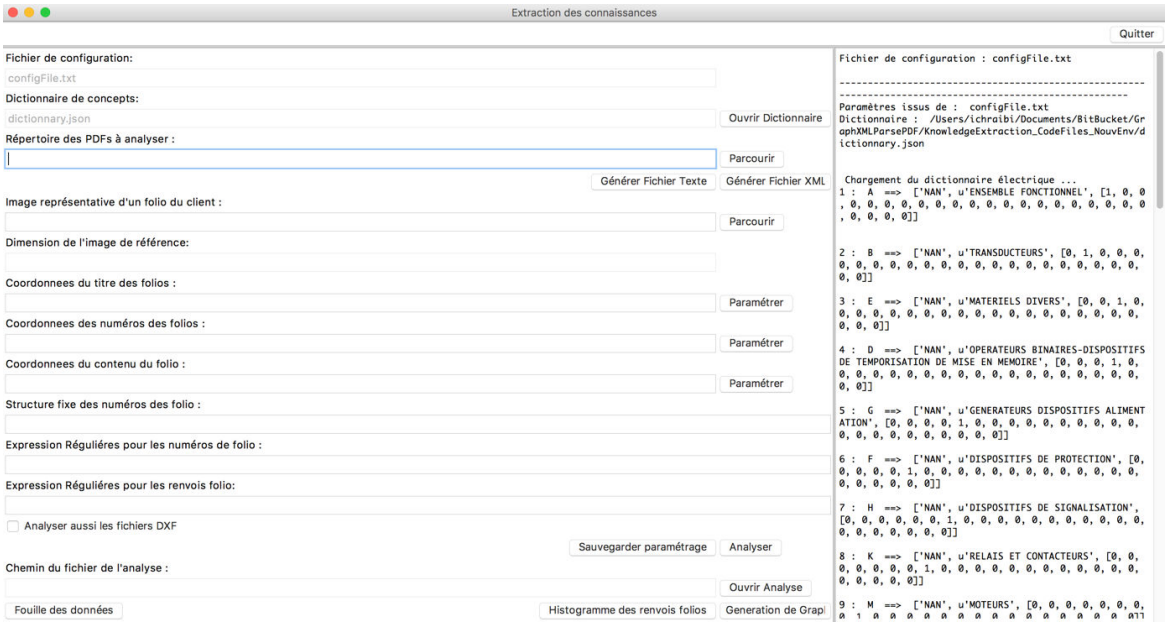
Voici un fichier de configuration vide à l'initialisation d'un projet :

```
#Fichier de configuration pour le parsage des documents PDF
#Attention, il ne faut pas d'espace avant ou après le signe "=" dans chaque
expressions
#les Saut à la ligne et espace entre chaque expression sont à conserver pour que les
expressions régulières fonctionnent
#Il est possible de mettre une ligne en commentaire en la précédant du symbole « # »

#Nom du client :


#Chemin du fichier PDF à transformer en graph (utiliser des "/" plutôt que des "\")
myPDF=""

FolioImg=""

#Coordonnées de la BBOX de l'image original
ImgBBOX=""


#Coordonnées de la BBOX pour extraire les titres des Folios
CbbtTitreFolio=""

#Coordonnées de la BBOX pour extraire les numéros de folios
CbbtNumFolio=""

#Coordonnées du contenu du folio (tout ce qui est dans le cadre, hors du cartouche)
FolioContentCoord=""

#-----------------------------------------------------------------------
#Expression pour filtrer les folios qui seront mis dans le graphe
FolioStructName=""

#Expression pour rechercher les numéros de folios dans le texte extrait de chaque
page du PDF
expReNumFolio=""

#Expression pour rechercher les renvois folios dans le texte extrait de chaque page
du PDF
expReIndicationsFolios=""


#Variable booléenne qui indique si l'on doit ou non traité les fichiers DXF
ProcessDXF=""


#-----------------------------------------------------------------------
#Expressions régulières pour chercher les repères des composants électriques dans les
textes des folios
A=""
B=""
D=""
E=""
F=""
G=""
H=""
K=""
KA=""

KM=""

L=""
```

A l’exception des expressions régulières pour recherche les repères des composants électriques dans les textes des folios, l’ensemble des champs, peuvent être paramétré à partir de l’interface du prototype.

Des indications importantes sont données en haut du document concernant sa forme et son contenu.

Voici l’interface au démarrage d’un nouveau client :



L'application :

- Charge les informations du fichier configFile.txt
- Localise le fichier dictionnary.json qui contiendra les concepts globaux du client
- Charge les données du dictionnaire électrique ElectricalDictionnary.json (document non accessible depuis l'interface) et qui liste pour chacune des 25 familles de composants électriques, un vecteur binaire l'identifiant.

1. Localiser le répertoire de PDFs ou le fichier à analyser

Quel que soit l'élément que l'on souhaite analyser, un PDF ou un ensemble de PDF, il faut toujours sélectionner un PDF dans le répertoire souhaité.

Ensuite si l'on souhaite analyser en une seule fois, tous les PDFs qui sont dans ce répertoire, il suffit de modifier directement à la main le chemin comme ci-dessous :

En effet, l'application ne permet pas de sélectionner un répertoire directement, mais que des PDFs.

2. Génération des fichiers textes et XMLs

Une fois la localisation réalisée, il est recommandé de lancer la génération des fichiers textes associés puis celles des fichiers XML.

Cette partie permettra de générer pour chaque PDF du client, un document texte et xml nommé comme le pdf.

Le document txt, permettra de vérifier que l'application est capable d'exploiter les PDFs en question.

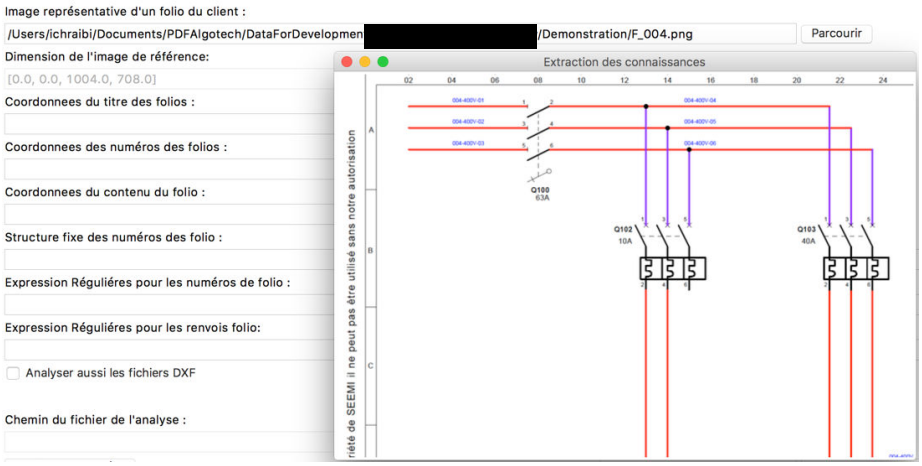
Le document xml permettra de gagner du temps, sur la phase d'analyse, qui les nécessitera dans tous les cas.

3. Image de référence et coordonnées à extraire

Préalablement, l'utilisateur a préparé une image png de référence représentant un folio du client courant.

a) Localisation de l'image de référence

Il s'agit de la première étape pour permettre l'extraction des coordonnées des différentes informations nécessaires à l'analyse



b) Extraction des coordonnées des titres des folios

Remarque :

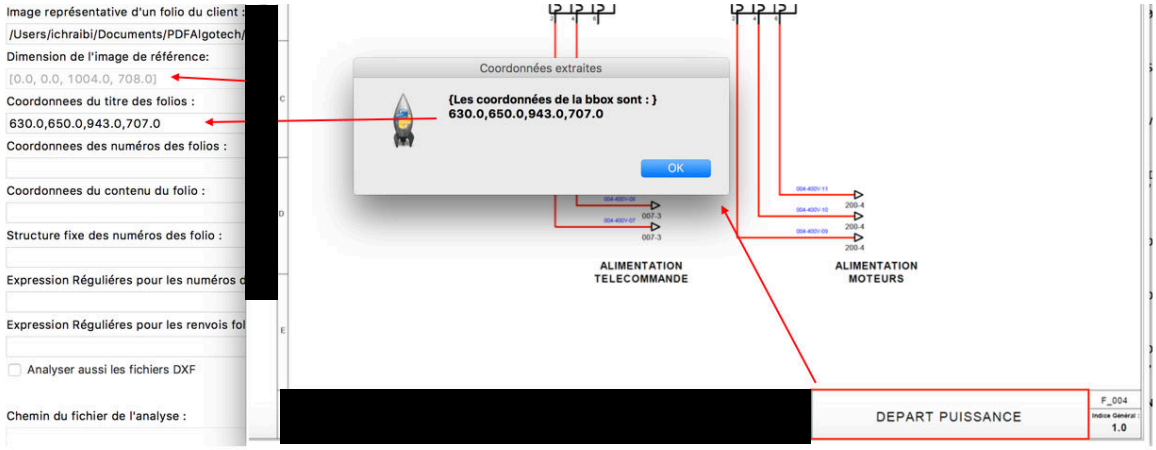
Pour l'extraction de l'image, il vaut mieux avoir la même taille d'image de référence. Il faut donc veiller à ne pas redimensionner l'image entre deux phases d'extraction des coordonnées, car sinon les dimensions de l'image de référence ne seront pas les mêmes.

L'utilisateur doit cliquer sur le bouton « Paramétrer », il verra ensuite l'image de référence se charger devant lui.

Il lui faut tracer un rectangle avec la souris autour de l'information qu'il souhaite extraire. Lors de cette étape, l'utilisateur doit veiller à bien englober la donnée.

A chaque rectangle tracé, un message apparait explicitant les coordonnées extraites.

Ces coordonnées sont aussi affichées dans le champ associé au niveau de l'interface principale de l'application.

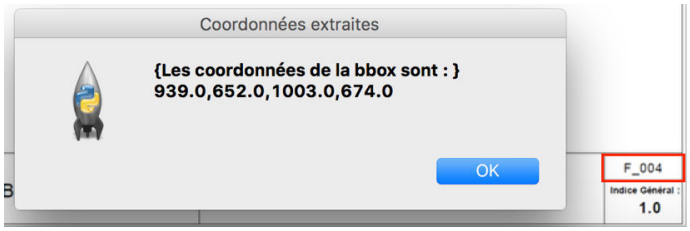


L'utilisateur peut cliquer et créer autant de rectangle qu'il le souhaite sur la figure. Seul le dernier rectangle dessiné sera pris en compte.

Il est a noté que les dimensions de l'image sont aussi chargées à chaque extraction. Ces coordonnées sont essentielles pour le calcul des nouvelles coordonnées à chaque fois que la page est de taille différente.

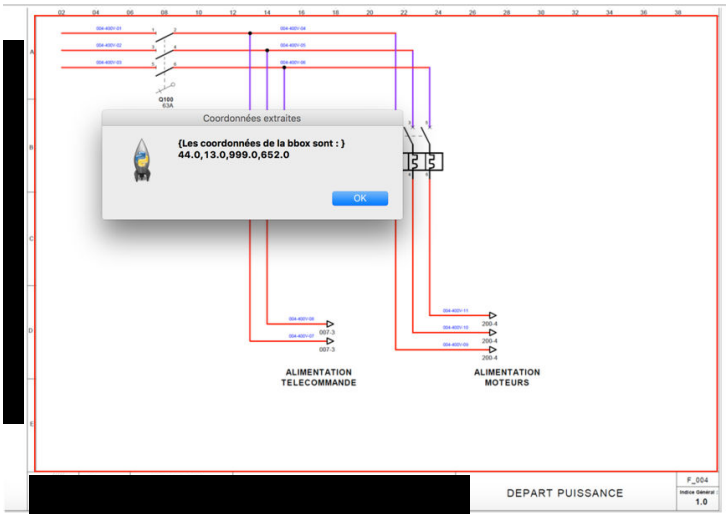
c) Extraction des coordonnées des numéros de folios

L'extraction de ces coordonnées suit le même procédé que l'extraction des coordonnées des titres des folios



d) Extraction des coordonnées de la partie schéma des numéros de folios

L'extraction de ces coordonnées suit le même procédé que l'extraction des coordonnées des titres des folios



4. Partie fixe d'un numéro de folio

Remarque :
Cette étape nécessite d'ouvrir deux fichiers textes relatif a deux pdfs distincts d'un même client, afin de chercher deux ou trois numéros de folios.
Cela permet d'une part de vérifier que cette information est bien extraite par l'application, et d'autre part de vérifier qu'elle est toujours extraite de la même manière.

Si les folios sont Folio+numéro, alors la partie fixe est 'Folio'. Dans le cas du client _____ , la partie fixe est :

Structure fixe des numéros des folio :

```
F_
```

5. Expressions régulières pour les numéros de folios

Remarque :
Nous avons choisi des expressions régulières pour le langage de programmation **Python**. Il faut donc faire attention à ce paramètre lors de recherche Google éventuelles car les regex s'adaptent selon le langage utilisé.

Les expressions régulières sont des expressions génériques qui permettent de rechercher dans un ensemble de texte des éléments bien particulier.

Plus une expression régulière se veut précise, plus elle est complexe. Il est recommandé d'utiliser les deux liens donnés en annexe (section IX.A.Expressions régulières) pour tester et en apprendre plus sur les expressions régulières, aussi appelés regex.

Remarque :
Pour les expressions régulières, en général, il vaut mieux aussi contrôler que l'extraction se fait bien en se basant sur les fichiers textes.
Il est alors possible d'utiliser le texte extrait sur un site de test de regex en ligne afin de vérifier que son expression retourne le résultat escompté.

Voici quelques indications pour rechercher des numéros de folio en se basant sur le cas _____:

Expression Réguliéres pour les numéros de folio :

```
F\_(?P<NumFolio>[0-9]{3})
```

Indication sur l'expression régulière de _____:	
F_	La partie fixe permet de préciser à la regex de ne chercher que les textes qui commencent par cette partie.
(?P<NumFolio> XXX)	Cette expression crée une balise « NumFolio »,

	indiqué par « ?P » et entouré de parenthèses, qui sera ensuite exploiter dans le code. Il faut donc inclure dans chaque regex créé : (?P<NumFolio>) XXX correspond à la regex, soit la partie variable que l'on cherche à extraire
[0-9]	Désigne que l'on cherche un chiffre entre 0 et 9
[0-9] {3}	Désigne que l'on cherche un chiffre entre 0 et 9 qui est répété 3 fois

Cette expression va permettre d'extraire les textes suivants : F_200, F_201, etc.

En fournissant à l'application l'indication que « NumFolio » est 200, 201, …

6. Expressions régulières pour les renvois folios

Expression Réguliéres pour les renvois folio:

```
(?:\n\s|\n|:|\s)(?P<NumPage>[0-9]{3})-(?:[0-9]{1,2}){1}$
```

Indication sur l'expression régulière de _____:	
(?:\n\s|\n|:|\s)	? : désigne que ce qui se trouve entre parenthèse sert juste pour la recherche, mais qu'on en souhaite pas le retrouver dans le résultat final \n\s|\n|:|\s implique que l'on cherche tout ce qui **commence par** ○ Soit un retour chariot suivi d'un espace(\n\s) ○ Soit un retour chariot seul(\n) ○ Soit du caractère : ○ Soit d'un espace blanc (\s) NB : la barre verticale « | » implique un OU
(?P<NumPage> XXX)	Cette expression crée une balise « NumPage », indiqué par « ?P » et entouré de parenthèses, qui sera ensuite exploiter dans le code. Il faut donc l'inclure dans chaque regex créér par la suite. XXX correspond à la regex, soit la partie variable que l'on cherche à extraire C'est tout se qui se trouve entre les parenthèses de la balise, qui sera exploité par l'application. Ce qu'il y a autour sert à préciser la regex.
[0-9] {3}	La partie fixe permet de préciser à la regex de ne chercher que les textes qui commencent par cette partie.
-(?:[0-9]{1,2}){1}$	? : désigne que ce qui se trouve entre parenthèse sert juste pour la recherche, mais qu'on en souhaite pas le retrouver dans le résultat final [0-9]{1,2} : désigne un chiffre entre 0 et 9 qui peut

	être présent 1 à 2 fois. {1} : implique que l'on cherche des textes où ([0-9]{1,2}) est présent une fois $ désigne que le texte recherché se situe en fin de ligne

Cette expression complexe va permettre d'extraire les textes suivants : 200-12, 201-40, etc.

En fournissant à l'application l'indication que « NumPage » est 200, 201, …

7. Analyse des fichiers DXF

Si pour chaque PDF, le client fournit un fichier DXF unique, il est alors possible de cocher la case à cocher «Analyser aussi les fichiers DXF ».

Le programme créera alors des onglets en plus relatif à l'analyse supplémentaire des fichiers DXF.

8. Sauvegarde des paramètres

A chaque sauvegarde des paramètres, un message s'affiche pour confirmer la bonne réalisation de l'opération.

De plus, dans la partie droite de l'écran, il est possible de trouver un récapitulatif des nouveaux paramètres.

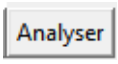
Cette affichage permet, s'il l'on a changé plusieurs fois de paramètres, d'avoir un rappel des différents paramètres testés.


```
Sauvegarde des nouveaux paramétres....
-------------------------------------
Nouveau paramétrage :
Répertoire ou PDF (myPDF) :  /Users/ichraibi/Documents
/PDFAlgotech/DataForDevelopment/Seemi/GenererAvecElecV
iew/Demonstration
Image de réference (FolioImg) :  /Users/ichraibi/Docum
ents/PDFAlgotech/DataForDevelopment/[redacted]
[redacted]/Demonstration/F_004.png
Dimension de l'imahe de référence (ImgBBOX) :
Coordonnées des titres de folios (TitleCoord) :  627.0
,647.0,942.0,708.0
Coordonnées des numéros de folios (RefFolioCoord) :  9
38.0,647.0,1005.0,675.0
Coordonnées de la partie schéma des folios (FolioConte
ntCoord) :  45.0,13.0,1003.0,653.0
Partie fixe des numéros des folios (FolioStructName) :
 F_
Expression régulière pour l'extraction des numéros de
folios (expReNumFolio) : F\_(?P<NumFolio>[0-9]{3})
Expression régulière pour l'extraction des renvois fol
ios (expReIndicationsFolios) :  (?:\n\s|\n|:|\s)(?P<Nu
mPage>[0-9]{3})-(?:[0-9]{1,2}){1}$
Inclure le traitement des fichiers DXF (ProcessDXF) :
FAUX
Parametres sauvegardés!
```

V- Etape 2 : Analyse et extraction des données dans un fichier Excel

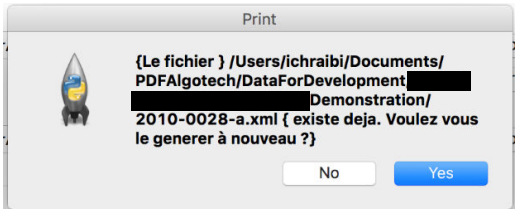
A. Lancement de l'analyse

1. Analyse des titres issus des PDFs

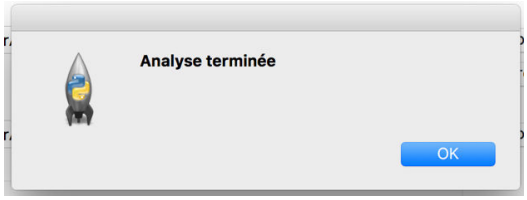
Pour lancer l'analyse, il faut cliquer le bouton 

.

Le prototype va alors vérifier que pour chaque PDF, un fichier xml existe. Si c'est le cas, l'application demande si l'on veut regénérer le fichier xml.



Si le fichier xml correspondant n'existe pas, l'application se chargera de le générer puis de l'exploiter.

Un message avertit de la fin de l'analyse : 

Et dans la partie droite de l'écran, l'utilisateur retrouve des logs qui indiquent l'avancé de l'analyse, ainsi que les fichiers créés.

```
Lancement de l'analyse....
Repertoire ou fichier PDF :  /Users/ichraibi/Documents
/PDFAlgotech/DataForDevelopment/[redacted]
[redacted]Demonstration

Fichier Excel :  /Users/ichraibi/Documents/PDFAlgotech
/DataForDevelopment/[redacted]/Demonstr
ation/20170214_DataExtraction.xlsx

Fichier excel créé :  /Users/ichraibi/Documents/PDFAlg
otech/DataForDevelopment/[redacted]/Dem
onstration/20170214_DataExtraction.xlsx


=>Lancement de l'analyse du PDF  :  /Users/ichraibi/Do
cuments/PDFAlgotech/DataForDevelopment/[redacted]
[redacted]/Demonstration/2010-0028-a.pdf
Creation du fichier  xml
Chemin du fichier  :  /Users/ichraibi/Documents/PDFAlg
otech/DataForDevelopment[redacted]/Dem
onstration/2010-0028-a.xml
=>Analyse en cours.....
XMLFile :  /Users/ichraibi/Documents/PDFAlgotech/DataF
orDevelopment[redacted]/Demonstration/
2010-0028-a.xml

Liste des PDF analysés :  ['2010-0028-a.pdf']


=>Lancement de l'analyse du PDF  :  /Users/ichraibi/Do
cuments/PDFAlgotech/DataForDevelopment/[redacted]
[redacted]/Demonstration/2010-0043-a.pdf
Creation du fichier  xml
Chemin du fichier  :  /Users/ichraibi/Documents/PDFAlg
otech/DataForDevelopment/[redacted]/Dem
onstration/2010-0043-a.xml
=>Analyse en cours.....
XMLFile :  /Users/ichraibi/Documents/PDFAlgotech/DataF
orDevelopment/[redacted]/Demonstration/
2010-0043-a.xml

Liste des PDF analysés :  ['2010-0028-a.pdf', '2010-00
43-a.pdf']
```

Au niveau de l'interface, le champ « Chemin du fichier de l'analyse » est désormais renseigné :

2. Analyse des fichiers DXF

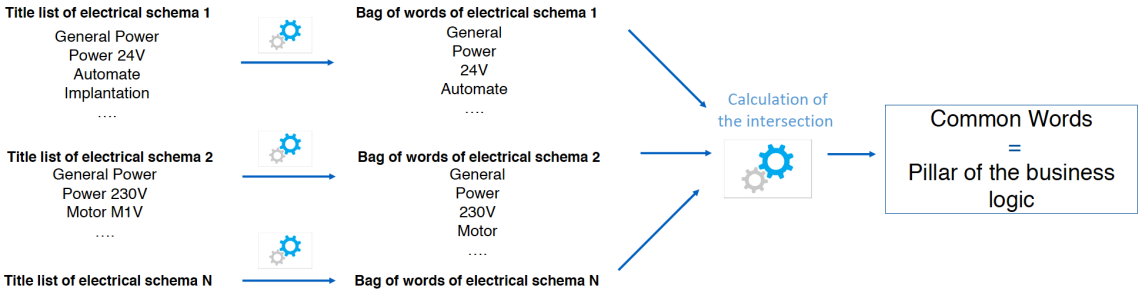
L'analyse des fichiers DXF nécessitent d'avoir pour chaque PDF, un seul fichier DXF associé complet.

L'analyse prend alors plus de temps, mais génère en sortie un fichier avec plus d'informations.

Voir section VI.C.2 Analyse avec les fichiers DXFs

B. Fonctionnement de l'analyse

1. Génération de sac de mots à partir des titres



Chaque titre de chaque pdf est décomposé en sac de mots.

Les sacs de mots de chaque schéma électrique sont utilisés pour calculer l'intersection entre eux.

Les mots communs résultants les concepts globaux détectés d'un client donné. Ces mots sont listé dans le fichier dictionnary.json qui est mis à jour à chaque analyse.

Les titres par folios et les sacs de mots sont aussi listés dans les premiers onglets du fichier excel d'analyse « DataExtraction.xls »

C. Fichier DataExtraction.xls

1. Analyse de base

Lorsque l'application réalise une analyse, elle produit en sortie un fichier excel en date du jour qui comporte plusieurs onglets :

Onglet	Information
Configuration	Reprend et liste les paramètres utilisés pour générer le fichier
List_of_Titles	Liste des titres par pdf
Bags_of_words	Sac de mots issus des titres de chaque PDF
Cleaned_Bags_Of_Words	Sac de mots filtré : - Pas de mots <3 lettres - Pas de caractères numériques ou non alphabétique
BOW_Analysis	Anayse des sac de mots : calcul d'union et d'intersection entre les différents sac de mots
FoliosIndication_Analysis	Comptabilisation des renvois folios distincts par PDF et par page
NF_FoliosIndication_Analysis	Comptabilisation des renvois folios par PDF et par page quel que soit le nombre de répétition
CommonWords_Analysis	Comptabilisation du nombre d'occurrence des mots communs sur le panel de pdf étudié

2. Analyse avec les fichiers DXFs

Lorsque les fichiers DXF sont fournit, l'analyse est alors plus longue, mais elle permet de générer des onglets supplémentaires.

Onglet	Information
EleCompLabel_Analysis	Liste des repères des composants électriques par PDF et trié par ordre alphabétique
ElecCompFamily_Analysis	Liste des repères des composants électriques par famille de composants électriques
NomFichierPDF	Liste l'ensemble des blocs INSERT (symboles de composants électriques) du fichier DXF ainsi que l'ensemble des attributs
Seq_NomFichierPDF	Liste par folio du PDF la séquence de repères ordonnée selon les cordonnées électriques et aussi la séquence de famille de composants qui en découle

Remarque : Les deux derniers onglets décrits sont spécifiques à un PDF. Il y en aura autant de créer que de PDF traités.

D. Fichier dictionnary.json

A la fin de l'analyse, le programme a réussi à extraire l'ensemble des concepts globaux en se basant sur les mots composant les titres des folios.

Ces concepts globaux, sont listés dans un fichier « dictionnary.json » que l'utilisateur peut alors visualiser, mais aussi compléter.

Le fichier se présente sous la forme :

{

conceptGlobal1 : [« mot associé 1 », « mot associé 2 », « mot associé 3 »]

conceptGlobal2 : [« mot associé 4 », « mot associé 5 », « mot associé 6 »]

}

Un exemple du dictionnaire pour _____ (généré automatiquement et complété manuellement) est disponible en annexe sous le titre « dictionnary_____.json »

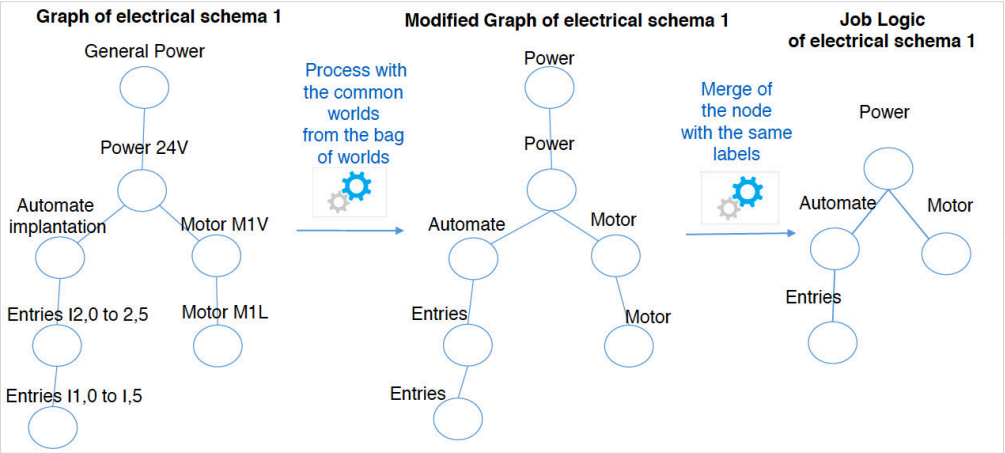
Une fois le dictionnaire modifié et le fichier d'analyse excel sauvegardé sous un autre nom, une phase d'analyse peut être généré à nouveau pour voir les résultats de l'extraction.

Il est **fortement conseillé de consulté et compléter ce fichier dictionnary.json en parallèle du fichier excel DataExtraction avant de lancer la génération des graphes**. Car cette étape suivante se base sur ce dictionnaire de concepts.

VI- Etape 3 : La génération de graphes

A. Fonctionnement de l'algorithme

1. Obtention de la logique métier de chaque pdf

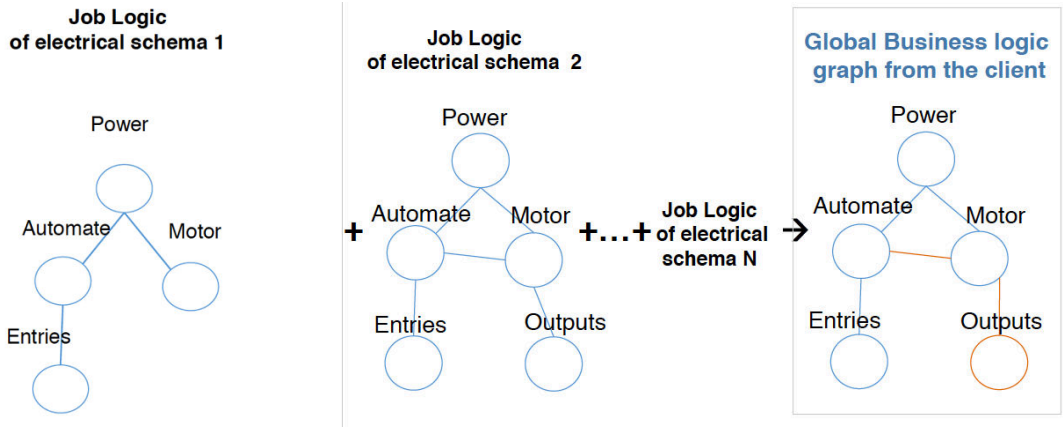


La première étape du traitement a été de générer un graphe à partir de chaque PDF en se basant sur les numéros de folios, et les renvois folios, les labels des nœuds étant les titres des folios.

Ensuite chaque titre (label de nœud) est remplacé par le mot commun associé en se basant sur le fichier dictionnary.json.

Les nœuds portant un même label sont alors fusionnés afin d'obtenir une représentation simplifiée de la connaissance contenu dans le PDF d'origine.

2. Fusion des graphes de logique métier de chaque pdfs



La seconde étape du traitement a été de récupéré pour chaque PDF, le graphe de la logique métier et de les concaténer ensemble.

En partant de l'hypothèse que chaque graphe est une instanciation de la connaissance globale (une partie) alors, leur concaténation, permettra de reconstruire la connaissance globale.

L'utilisateur obtiendra ainsi le graphe des connaissances globale du client construit directement de ses pdfs.

B. Images produites

Pour un schéma PDF, 2 graphes sont créée :

NomPDF**_original.png**	Un graph au format png avec tout les folios, même ceux sans lien
NomPDF**_Filtered.json**	Un fichier json qui décrit le graph original sans les nœuds sans lien
NomPDF**_JLG.png**	Un graph graph de la logique métier avec des nœuds standardisés JLG = Job Logic Graph (graph de la logique métier
NomPDF**_JLG_Filtered**.json	Un fichier json qui décrit le graph standardisé sans les nœuds sans lien

Si l'analyse porte sur plus de 2 PDFs, alors le graphe des concepts métier est aussi généré sous le nom DateDuJour**_SemanticGraph_.png.**

Ce dernier regroupe tous les concepts métiers utilisés par le client sur l'ensemble des PDFs analysés ainsi que les relations entre eux.

VII- Etape 4 : Le clustering hiérarchique

1. Processus et fichiers générés

Le clustering hiérarchique est une méthode d'analyse qui vise à construire une hiérarchie de cluster en utilisant une distance et une méthode de mesure pour créer ces clusters.

Un cluster étant un ensemble dont les composants partagent des caractéristiques communes.

Grace à la phase d'analyse, le programme a extrait les composants électriques sur chaque folio ainsi que leur coordonnées x, y.

C'est selon un x et un y croissant que les composants électriques ont été ordonnées au sein de chaque folio.

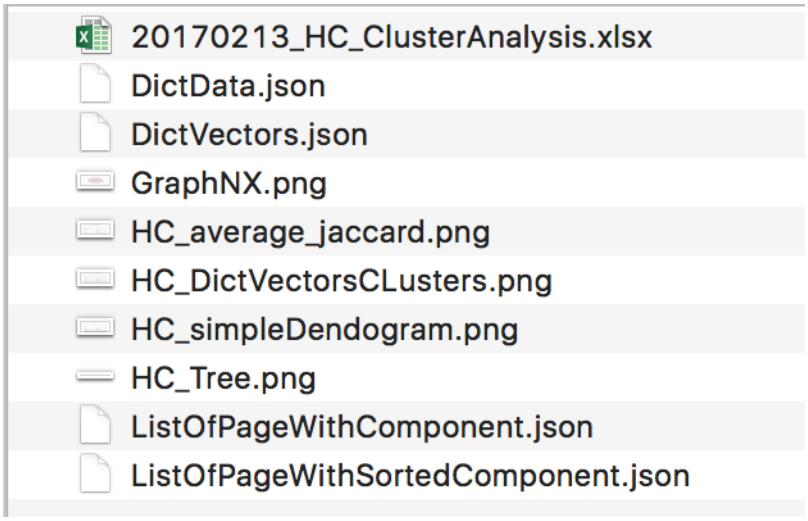
Pour cette partie, deux jeux de données ont été constitués :

- Le premier jeu de données : regroupe tous les folios ensemble tout PDF confondus
- Le second jeu de données regroupe les folios tout PDF confondus, mais par concept métier d'appartenance.

Chaque folio, qui correspond à une séquence de composant électrique est ensuite traduit en un vecteur binaire de longueur 25 (car nous avons recensé 25 familles de composants électriques) : chaque unité désignant la présence (1) ou l'absence (0) d'une famille de composant au sein d'un folio.

Ces deux ensemble de données sont ensuite analysée. Dans le répertoire des fichiers analysés, plusieurs éléments sont créés :

- Un répertoire GLOBAL_HC qui correspond au résultat du hierarchical clustering sur le premier jeux de données



Ce répertoire comporte alors plusieurs fichiers :

- Le fichier d'analyse excel en date du jour
- DictData.json : qui contient le détail des données du premier set. Pour chaque folio, on y trouve :
    - Le vecteur booléen associé
    - La séquence de catégorie de composant ex : [Q, Q]
    - Le titre du folio
    - Son PDF d'origine
    - La séquence de repère tel qu'elle apparait dans le folio ex : [Q100, Q102]
    - Le numéro de page dans le PDF
    - Le vecteur numérique : celui qui comptabilise le nombre de composants de chaque famille dans le folio
- DictVectord.json : qui contient des détails des données du premier set (mais moins de données que DictData.json). Pour chaque folio, on y trouve :
    - La séquence de catégorie de composant ex : [Q, Q]
    - Le titre du folio
    - Le vecteur booléen associé
    - Le vecteur numérique associé
    - La séquence de repère tel qu'elle apparait dans le folio ex : [Q100, Q102]
- HC_DictVectorsClusters.png : Dendogram (représentation graphique des résultats du clustering hiérarchique) avec les numéros de clusters explicitement marqué dessus
- HC_average_jaccard.png : Dendogram avec les distances entre cluster calculer explicitement marqué dessus.
- HC_simpleDendogram.png : Dendogram simple sans annotation sur lequel on trouve les noms des folios sur l'axe des abscisses (l'axe des x)
- HC_Tree.png : est un arbre hiérarchique de cluster dans lequel les nœuds sont des folios. Chaque nœud comporte :
    - Le numéro de cluster
    - Le nom du folio
    - La séquence de catégorie de composant ex : [Q, Q]
    - La séquence de repères ex : [Q100, Q100]
- ListOfPageWithComponent.json : Fichier json qui comporte pour chaque cluster :
    - La séquence de catégorie de composant

    - Le titre du folio
    - La séquence de repère
- ListOfPageWithSortedComponent.json : comporte pour chaque folio traité
    - Son titre
    - La séquence ordonnée de catégorie de famille selon leur coordonnées x et y croissantes

2. Fichier Excel HC_ClusterAnalysis.xlsx

Ce fichier comporte un onglet important : « CLusterAnalysis »

Cet onglet regroupe l'analyse des clusters en explicitant à chaque ligne :

- les clusters calculés 2 à 2
- leurs vecteurs binaires associés,
- leur titre du folio
- Leurs séquences de repères mais aussi de famille de composants
- Et surtout les distances calculées selon différentes mesures : Jacquard (celle utilisé par défaut pour l'instant) Hamming, Euclidien,
- Et enfin pour chaque ligne, le nombre total de clusters qui ont été fusionné ensemble.

VIII- Annexe et Liens supplémentaires

A. Expressions régulières

- Pour en savoir plus sur les expressions régulières : http://apprendre-python.com/page-expressions-regulieres-regular-python
- Pour tester ses propres expressions régulières sans utiliser le prototype : https://regex101.com/

B. Annexe : dictionnary.json de _____


```
{
  "ECHANGES": ["ECHANGES"],
  "INDICE": ["INDICE"],
  "CARNET": ["CARNET"],
  "PROTECTIONS": ["PROTECTIONS"],
  "ELECTRIQUE": ["ELECTRIQUE", "ELECTRIQUES"],
  "DATE": ["DATE"],
  "VARIATEUR": ["VARIATEUR", "VARIATEURS", "VAR.", "RESEAU", "RESEAUX"],
  "RACCORDEMENT": ["RACCORDEMENT"],
  "INSTALLATION": ["INSTALLATION", "INSTALLATIONS"],
  "SECURITE": ["SECURITE", "LIGNE", "LIGNES", "SIGNALISATION",
"SIGNALISATIONS", "RADAR", "RADARS"],
  "LOCAL": ["LOCAL"],
  "NORME": ["NORME"],
  "ETUDIE": ["ETUDIE"],
  "GENERALE": ["GENERALE"],
  "AUTOMATE": ["AUTOMATE", "AUTOMATES", "STATION", "STATIONS", "SORTIE",
```

```
"SORTIES", "Q0.0", "Q0.3", "Q1.0",
    "Q1.3", "Q2.0", "Q2.3", "Q3.0", "Q3.3", "Q4.0", "Q4.3", "Q5.0", "Q5.3",
"Q6.0", "Q6.3", "Q7.0", "Q7.3",
    "Q8.0", "Q8.3", "Q9.0", "Q9.3", "Q10.0", "Q10.3", "Q15.0", "Q15.3",
"Q16.0", "Q16.3", "Q17.0", "Q17.3",
    "Q18.0", "Q18.3", "Q19.0", "Q19.3", "Q20.0", "Q20.3", "Q21.0", "Q21.3",
"Q22.0", "Q22.3", "Q23.0", "Q23.3",
    "Q24.0", "Q24.3", "Q25.0", "Q25.3", "ENTREE", "ENTREES", "I0.0",
"I0.3", "I1.0", "I1.3", "I2.0", "I2.3", "I3.0",
    "I3.3", "I4.0", "I4.3", "I5.0", "I5.3", "I6.0", "I6.3", "I7.0", "I7.3",
"I8.0", "I8.3", "I9.0", "I9.3", "I10.0",
    "I10.3", "I15.0", "I15.3", "I16.0", "I16.3", "I17.0", "I17.3", "I18.0",
"I18.3", "I19.0", "I19.3", "I20.0", "I20.3",
    "I21.0", "I21.3", "I22.0", "I22.3", "I23.0", "I23.3", "I24.0", "I24.3",
"I25.0", "I25.3"],
  "BORNE": ["BORNE", "BORNES", "BORNIERS", "BORNIER", "LISTE", "LISTES"],
  "OBSERVATION": ["OBSERVATION"],
  "CABLE": ["CABLE", "CABLES"],
  "ELEMENT": ["ELEMENT", "ELEMENTS"],
  "FILTRE": ["FILTRE"],
  "PUPITRE": ["IMPLANTATION", "IMPLANTATIONS", "PORTE", "PORTES",
"ARMOIRE", "ARMOIRES"],
  "CARACTERISTIQUES": ["CARACTERISTIQUE", "CARACTERISTIQUES"],
  "STOCK": ["STOCK"],
  "ALIMENTATION": ["ALIMENTATION", "ALIMENTATIONS", "DISTRIBUTION",
"PUISSANCE", "PUISSANCES"],
  "BARRES": ["BARRES"],
  "EQUIPEMENT": ["EQUIPEMENT", "EQUIPEMENTS", "MOTEUR", "MOTEURS",
"MOVIMOTS", "MOVIMOT", "MOVI",
    "MOVIDRIVES", "MOVIDRIVE", "CONVOYEUR", "CONVOYEURS", "CODEUR",
"CODEURS", "VENTILATEUR", "VENTILATEURS"],
  "MASSES": ["MASSES"],
  "BARRAGE": ["BARRAGE", "BARRAGES"],
  "COMPOSANT": ["COMPOSANT", "COMPOSANTS"],
  "NAVETTE": ["NAVETTES", "NAVETTE"]
}
```

Annexe B

Dictionnaire des mots communs

```
{
        "ECHANGES": ["ECHANGES"],
        "ELECTRIQUE": ["ELECTRIQUE", "ELECTRIQUES"],
        "CARNET": ["CARNET"],
        "PUPITRE": ["IMPLANTATION", "IMPLANTATIONS", "PORTE",
"PORTES", "ARMOIRE", "ARMOIRES"],
        "INSTALLATION": ["INSTALLATION", "INSTALLATIONS"],
        "SECURITE": ["SECURITE", "LIGNE", "LIGNES", "SIGNALISATION",
"SIGNALISATIONS", "RADAR", "RADARS"],
        "VARIATEUR": ["VARIATEUR", "VARIATEURS", "VAR.", "RESEAU",
"RESEAUX"],
        "NORME": ["NORME"],
        "GENERALE": ["GENERALE"],
        "AUTOMATE": ["AUTOMATE", "AUTOMATES", "STATION", "STATIONS",
"SORTIE", "SORTIES", "Q0.0", "Q0.3", "Q1.0", "Q1.3", "Q2.0", "Q2.3",
"Q3.0", "Q3.3", "Q4.0", "Q4.3", "Q5.0", "Q5.3", "Q6.0", "Q6.3",
"Q7.0", "Q7.3", "Q8.0", "Q8.3", "Q9.0", "Q9.3", "Q10.0", "Q10.3",
"Q15.0", "Q15.3", "Q16.0", "Q16.3", "Q17.0", "Q17.3", "Q18.0",
"Q18.3", "Q19.0", "Q19.3", "Q20.0", "Q20.3", "Q21.0", "Q21.3",
"Q22.0", "Q22.3", "Q23.0", "Q23.3", "Q24.0", "Q24.3", "Q25.0",
"Q25.3", "ENTREE", "ENTREES", "I0.0", "I0.3", "I1.0", "I1.3",
"I2.0", "I2.3", "I3.0", "I3.3", "I4.0", "I4.3", "I5.0", "I5.3",
"I6.0", "I6.3", "I7.0", "I7.3", "I8.0", "I8.3", "I9.0", "I9.3",
"I10.0", "I10.3", "I15.0", "I15.3", "I16.0", "I16.3", "I17.0",
"I17.3", "I18.0", "I18.3", "I19.0", "I19.3", "I20.0", "I20.3",
"I21.0", "I21.3", "I22.0", "I22.3", "I23.0", "I23.3", "I24.0",
"I24.3", "I25.0", "I25.3"],
        "BORNE": ["BORNE", "BORNES", "BORNIERS", "BORNIER", "LISTE",
"LISTES"],
        "CABLE": ["CABLE", "CABLES"],
        "ELEMENT": ["ELEMENT", "ELEMENTS"],
        "FILTRE": ["FILTRE"],
        "RACCORDEMENT": ["RACCORDEMENT"],
        "CARACTERISTIQUES": ["CARACTERISTIQUE", "CARACTERISTIQUES"],
        "STOCK": ["STOCK"],
        "ALIMENTATION": ["ALIMENTATION", "ALIMENTATIONS",
"DISTRIBUTION", "PUISSANCE", "PUISSANCES"],
        "EQUIPEMENT": ["EQUIPEMENT", "EQUIPEMENTS", "MOTEUR",
"MOTEURS", "MOVIMOTS", "MOVIMOT", "MOVI", "MOVIDRIVES", "MOVIDRIVE",
"CONVOYEUR", "CONVOYEURS", "CODEUR", "CODEURS", "VENTILATEUR",
"VENTILATEURS"],
        "MASSES": ["MASSES"],
        "BARRAGE": ["BARRAGE", "BARRAGES"],
        "COMPOSANT": ["COMPOSANT", "COMPOSANTS"],
        "NAVETTE": ["NAVETTES", "NAVETTE"]
}
```

Annexe C

Dictionnaire des catégories de composants électriques

```
{
"A" : ["ENSEMBLE FONCTIONNEL"],
"B" : ["TRANSDUCTEURS"],
"C" :["CONDENSATEURS"],
"D" : ["OPERATEURS BINAIRES-DISPOSITIFS DE TEMPORISATION DE MISE EN
MEMOIRE"],
"E" : ["MATERIELS DIVERS"],
"F" : ["DISPOSITIFS DE PROTECTION"],
"G" : ["GENERATEURS DISPOSITIFS ALIMENTATION"],
"H" : ["DISPOSITIFS DE SIGNALISATION"],
"K" : ["RELAIS ET CONTACTEURS"],
"KA" : ["CONTACTEURS AUXILIAIRES RELAIS D AUTOMATISME"],
"KM" : ["CONTACTEURS PRINCIPAUX"],
"L" : ["INDUCTANCES"],
"M" : ["MOTEURS"],
"N" : ["SOUS-ENSEMBLES HORS SERIE"],
"P" : ["INSTRUMENT DE MESURE ET D ESSAI"],
"Q" : ["APPAREILS MECANIQUE DE CONNEXION POUR CIRCUITS DE
PUISSANCE"],
"R"  : ["RESISTANCES"],
"S" : ["APPAREILS MECANIQUES DE CONNEXION POUR CIRCUIT DE
COMMANDE"],
"T" : ["TRANSFORMATEURS"],
"U" : ["MODULATEURS-CONVERTISSEURS"],
"V" : ["TUBES ELECTRONIQUES- SEMI CONDUCTEURS"],
"W" : ["VOIES DE TRANSMISSION-GUIDES D ONDES-ANTENNES"],
"X" : ["BORNES-FICHES-SOCLES"],
"Y" : ["APPAREILS MECANIQUES ACTIONNES ELECTRIQUEMENT"],
"Z" : ["CHARGES CORRECTIVES-TRANSFORMATEURS DIFFRENTIELS-FILTES
CORRECTEURS-LIMITEURS"]
}
```

Annexe D

Algorithme d'apprentissage de SRN

Algorithme 4 Algorithme d'apprentissage du SRN

Pour tout séquence de taille n **faire**

Pour $t = 0$ à $(n - 1)$ **faire**
# Etape 1 : propagation avant
entrée_réseau = séquence[t]
sortie_attendue = séquence[$t + 1$]
sortie_obtenue = SRN.propagation_avant(entrée_réseau)
Erreur_SRN = sortie_attendue - sortie_obtenue
# Etape 2 : propagation arrière
# Etape 2a : calcul de l'erreur pour chaque unité de la couche de sortie
# σ' : dérivée de la fonction d'activation σ des unités du SRN
Pour tout unité k de la couche de sortie **faire**
δ_k = Erreur_SRN[k] * σ '(sortie_obtenue[k])
Fin Pour
Etape 2b :Calcul de l'erreur au niveau de la couche cachée pour chaque unité j et au niveau de la couche d'entrée
Pour tout couche du réseau, de la couche cachée à la couche d'entrée **faire**

Pour tout unité j de la couche courante c **faire**
σ' : dérivée de la fonction d'activation σ des unités du SRN
$w_{(k,j)}$: poids provenant de j et allant vers k
y_j : somme des entrées de l'unité j
$\delta_j = \sigma$ '(y_j) * $\sum_k (w_{(k,j)}$ * $(\delta_k))$
Fin Pour
Fin Pour
Etape 2c : Mise à jour des poids
Pour tout poids $w_{(j,i)}$ **faire**
x_{ji} : entrée associée au lien entre l'unité i vers l'unité j
Variation du poids = Taux d'apprentissage * δ_j * x_{ji} + momentum * (Variation du poids(t-1))
Nouveau poids = Ancien poids + Variation du poids
Fin Pour
Fin Pour
Fin Pour

Annexe E

Algorithme d'apprentissage de RNN-LSTM d'après Gers et Schmidhuber [2001]

Voici ci dessous notre pseudo-code présentant l'apprentissage dans un RNN-LSTM largement inspiré des travaux de Gers et Schmidhuber [2001]

Algorithme 5 Algorithme d'apprentissage d'un RNN-LSTM

Pré-conditions : Soit les fonctions d'activations suivantes :
$f(x) = \frac{1}{1+e^{-x}}$
$h(x) = 2 * f(x) - 1$
$g(x) = 4 * f(x) - 2$
et leur dérivées respectives f', h' et g'
Pour tout sequence de taille n **faire**

Pour $i = 0$ à $(n-1)$ **faire**
# Etape 1 : propagation avant
entrée_réseau = sequence[i]
sortie_attendue = sequence[$i+1$]
# Etape 1 : propagation avant
sortie_obtenue = RNN_LSTM.propagation_avant(entrée_réseau)
Erreur_RNN_LSTM = sortie_attendue - sortie_obtenue
# Etape 2 : propagation arrière
RNN_LSTM.propagation_arrière(Erreur_RNN_LSTM)
Fin Pour
Fin Pour

Algorithme 6 Apprentissage d'un RNN-LSTM : propagation avant

Fonction propagation_avant (entrée_réseau)
propagation avant de la couche d'entrée à la couche cachée
Pour tout bloc j **faire**

Pour tout cellule c_j^v **faire**
Calcul des activations reçues (parmi les neurones m, se trouvent les neurones de la couche d'entrée qui reçoivent l'entrée_réseau)

$$net_l(t) = \sum_m (w_{(l,m)} \cdot y^m(t)) + \sum_p (w_{(l,p)} \cdot y^p(t-1)) \tag{E.1}$$

$$avecl \in \{c_j^v, in_j, \phi_j, out_j\} \tag{E.2}$$

Calcul des activations selon les fonctions d'activations

$$y^{in_j}(t) = f_{in_j}(net_{in_j}(t)) \tag{E.3}$$

$$y^{\phi_j}(t) = f_{\phi_j}(net_{\phi_j}(t)) \tag{E.4}$$

$$y^{out_j}(t) = f_{out_j}(net_{out_j}(t)) \tag{E.5}$$

$$s_{c_j^v}(t) = g(net_{c_j^v}(t)) \cdot y^{in_j}(t) + s_{c_j^v}(t-1) \cdot y^{\phi_j}(t) \tag{E.6}$$

$$y^{c_j^v}(t) = h(s_{c_j^v}(t)) \cdot y^{out_j}(t) \tag{E.7}$$

Calcul des dérivées partielles

$$dS_{c,m}^{jv}(t) = dS_{cm}^{jv}(t-1) * y^{\phi_j}(t) + g'(net_{c_j^v})(t) * y^{in_j}(t) * y^m(t-1) \tag{E.8}$$

$$dS_{in,m}^{jv}(t) = dS_{in,m}^{jv}(t-1) * y^{\phi_j}(t) + g(net_{c_j^v}(t)) * f'_{in_j}(net_{in_j}(t)) * y^m(t-1) \tag{E.9}$$

$$dS_{\phi,m}^{jv}(t) = dS_{\phi,m}^{jv}(t-1) * y^{\phi_j}(t) + h(s_{c_j^v}(t)) * f'_{\phi_j}(net_{\phi_j}(t)) * y^m(t-1) \tag{E.10}$$

Fin Pour
Fin Pour
# propagation avant de la couche cachée à la couche de sortie
Erreur_RNN_LSTM = vecteur de taille k
Pour tout unité k de la couche de sortie **faire**
Calcul de la sortie du réseau : activation de la couche de sortie

$$net_k(t) = \sum_m (w_{(k,m)} \cdot y^m(t)) + \sum_p (w_{(k,p)} \cdot y^p(t-1)) \tag{E.11}$$

$$y^k(t) = f_k(net_k(t)) \tag{E.12}$$

$$Erreur\_RNN\_LSTM[k] = y^k(t) \tag{E.13}$$

Fin Pour
return Erreur_RNN_LSTM
Fin Fonction

Algorithme 7 Algorithme d'apprentissage d'un RNN-LSTM : propagation arrière

Fonction propagation_arrière (Erreur_RNN_LSTM) :
Etape 2 : propagation arrière
Pour tout k de la couche de sortie **faire**

$$e^k(t) = Erreur\_RNN\_LSTM[k] \tag{E.14}$$
$$\delta_k(t) = f'_k(net_k(t)) * (e^k(t)) \tag{E.15}$$

Fin Pour
Pour tout bloc j **faire**

Pour tout cellule c_j^v **faire**
entre les unités k de la couche de sortie et les portes de sorties des blocs j

$$\delta_{out_j}(t) = f'_{out_j}(net_{out_j}(t)) * (\sum_{v=1}^{S_j} h(s_{c_j^v}(t)) * (\sum_m w_{kc_j^v} * \delta_k(t))) \tag{E.16}$$

entre les unités k de la couche de sortie et les portes **d'entrées**, les portes **d'oublies**, ainsi que les **cellules** des blocs j

$$e_{c_j^v}(t) = y^{out_j} * h'(s_{c_j^v}(t)) * (\sum_m w_{kc_j^v} * \delta_k(t))) \tag{E.17}$$

Fin Pour
Fin Pour
Mise à jour des poids d'un réseau RNN-LSTM

$$\Delta_{lm}(t) = \alpha * (\delta_l(t)) * y^m(t), l \in \{k, i, out\} \tag{E.18}$$
$$\Delta_{c_j^v,m}(t) = \alpha * e_{c_j^v}(t) * dS^j v_{c,m}(t) \tag{E.19}$$
$$\Delta_{in,m}(t) = \alpha * (\sum_{v=1}^{S_j} e_{c_j^v}(t) * dS^j v_{in,m}(t)) \tag{E.20}$$
$$\Delta_{\phi,m}(t) = \alpha * (\sum_{v=1}^{S_j} e_{c_j^v}(t) * dS^j v_{\phi,m}(t)) \tag{E.21}$$
$$W_{zm}(t) = W_{zm}(t) + \Delta_{zm}(t), z \in \{k, i, out, c_j^v, in, \phi\} \tag{E.22}$$

Fin Fonction

Bibliographie

Aimé, Xavier, Furst, Frédéric, Kuntz, Pascale et Trichet, Francky, 2010. Enrichissement sémantique de requêtes au moyen d'ontologies de domaine personnalisées. *Actes de l'atelier Personnalisation du Web, 10ième Journées francophones d'Extraction et de Gestion de Connaissances (EGC'2010), Hammamet*, 189.

Aimé, Xavier et Charlet, Jean, 2013. IC : Ingénierie des connaissances ou ingénierie du conformisme ? Dans *IC-24èmes Journées francophones d'Ingénierie des Connaissances*.
URL https://hal.inria.fr/hal-01103767/

Alonso, Fernando, Martínez, Loïc, Pérez, Aurora et Valente, Juan P, 2012. Cooperation between expert knowledge and data mining discovered knowledge : Lessons learned. *Expert Systems with Applications*, 39(8) :7524–7535.

Alvarez-Macias, JL, Mata-Vazquez, J et Riquelme-Santos, JC, 2004. Data mining for the management of software development process. *International Journal of Software Engineering and Knowledge Engineering*, 14(06) :665–695.

Amini, Massih-Reza et Gaussier, Eric, 2013. *Recherche d'information : applications, modèles et algorithmes*. Editions Eyrolles.

Anderson, John R, 1987. Skill acquisition : Compilation of weak-method problem situations. *Psychological review*, 94(2) :192.

Anderson, John R, 2005. *Cognitive psychology and its implications*. Macmillan.

Anderson, JR, 1983. Cognitive science series. the architecture of cognition.

Angeline, Peter J, Saunders, Gregory M et Pollack, Jordan B, 1994. An evolutionary algorithm that constructs recurrent neural networks. *IEEE transactions on Neural Networks*, 5(1) :54–65.

BARNEY, Jay, 1991. Firm resources and sustained competitive advantage. *Journal of management*, 17(1) :99–120.

BELAÏD, Abdel et CECOTTI, Hubert, 2006. Reconnaissance de caractères : évaluation des performances.

BENGIO, Yoshua, SIMARD, Patrice et FRASCONI, Paolo, 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2) :157–166.

BENGIO, Yoshua *et al.*, 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1) :1–127.

BERRY, Dianne C et DIENES, Zoltan, 1991. The relationship between implicit memory and implicit learning. *British Journal of psychology*, 82(3) :359–373.

BHARGAVA, Sanjay et PUROHIT, GN, 2011. Parsing with neural and finite automata networks : A graph grammar approach. *International Journal of Computer Applications*, 23(4).

BLACK, Paul E., 2004. "finite state machine". in *Dictionary of Algorithms and Data Structures* [online], Vreda Pieterse and Paul E. Black eds. Available from : https ://xlinux.nist.gov/dads/HTML/finiteStateMachine.html (accessed 15 November 2017).

BLANCO, Armando, DELGADO, Miguel et PEGALAJAR, MC, 2000. Extracting rules from a (fuzzy/crisp) recurrent neural network using a self-organizing map. *International Journal of Intelligent Systems*, 15(7) :595–621.

BOOTZ, Jean-Philippe, 2015. Comment concilier auto-organisation et contrôle au sein des communautés de pratique pilotées ? : une scoping review/how to reconcile self-organization and control in driven communities of practice ? : a scoping review/como se pueden conciliar la auto organización y el control en las comunidades de práctica conducidas ? una scoping review. *Management International*, 19(3) :15.

BOOTZ, Jean-Philippe, 2016. Comment définir et gérer l'expert ? Available from : https ://www.agecso.com/wp/wp-content/uploads/2016/01/BourbaKeM-5.pdf (accessed 20 December 2017).

BOOTZ, Jean-Philippe et SCHENK, Eric, 2014. L'expert en entreprise : proposition d'un modèle définitionnel et enjeux de gestion. *Management & Avenir*, (1) :78–100.

Bouzidi, Khalil Riad, Faron-Zucker, Catherine, Fies, Bruno, Corby, Olivier et Le Thanh, Nhan, 2010. Modélisation de documents réglementaires dans le domaine du bâtiment. Dans *Proc. 12e Conférence Internationale Francophone sur l'Extraction et la Gestion de Connaissance, EGC 2011*.

Broadbent, Donald E, 1977. Levels, hierarchies, and the locus of control. *The Quarterly Journal of Experimental Psychology*, 29(2) :181–201.

Carr, Thomas H, McCauley, Charley, Sperber, Richard D et Parmelee, CM, 1982. Words, pictures, and priming : on semantic activation, conscious identification, and the automaticity of information processing. *Journal of Experimental Psychology : Human Perception and Performance*, 8(6) :757.

Casey, Mike, 1998. Correction to proof that recurrent neural networks can robustly recognize only regular languages. *Neural Computation*, 10(5) :1067–1069.

Cernansky, Michal et Benuskova, Lubica, 2003. Simple recurrent network trained by rtrl and extended kalman filter algorithms. *Neural Network World*, 13(3) :223–234.

Chambaron-Ginhac, Stéphanie, 2005. *Apprentissage implicite de régularités : mise en évidence d'une différence d'apprentissage entre tâches motrices continues et discrètes*. Thèse de doctorat, Dijon.

Charlet, Jean, Declerck, Gunnar, Dhombres, Ferdinand, Gayet, Pierre, Miroux, Patrick et Vandenbussche, Pierre-Yves, 2012. Construire une ontologie médicale pour la recherche d'information : problématiques terminologiques et de modélisation. Dans *23es journées francophones d'Ingénierie des connaissances*, pages 33–48.

Charness, Neil et Campbell, Jamie ID, 1988. Acquiring skill at mental calculation in adulthood : A task decomposition. *Journal of Experimental Psychology : General*, 117(2) :115.

Chaudet, Bruno, 2009. Donnée, information, connaissance.
URL https://brunochaudet.wordpress.com/2009/03/30/donnee-information-connaissance/

Cherfi, Hacene, Napoli, Amedeo et Toussaint, Yannick, 2003. Vers une méthodologie de fouille de textes s' appuyant sur l'extraction de motifs fréquents et de règles d'association. Dans *Conférence d'Apprentissage (CAp'03), dans le cadre de la plate-forme (AFIA'03)*, pages 61–76. Presses universitaires de Grenoble.

Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger et Bengio, Yoshua, 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv :1406.1078*.

Chomsky, Noam, 1956. Three models for the description of language. *IRE Transactions on information theory*, 2(3) :113–124.

Chung, Junyoung, Ahn, Sungjin et Bengio, Yoshua, 2016. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv :1609.01704*.

Chung, Junyoung, Gülçehre, Çaglar, Cho, KyungHyun et Bengio, Yoshua, 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.
URL http://arxiv.org/abs/1412.3555

Cleeremans, A et Jiménez, L, 1996. Implicit cognition with the symbolic metaphor of mind : Theoretical and methodological issues. *Unpublished manuscript*.

Cleeremans, Axel, 1993. *Mechanisms of implicit learning : Connectionist models of sequence processing*. MIT press.

Cleeremans, Axel et McClelland, James L, 1991. Learning the structure of event sequences. *Journal of Experimental Psychology : General*, 120(3) :235.

Cleeremans, Axel, Servan-Schreiber, David et McClelland, James L, 1989. Finite state automata and simple recurrent networks. *Neural computation*, 1(3) :372–381.

Cordier, Amélie, Lefevre, Marie, Champin, Pierre-Antoine, Mille, Alain, Georgeon, Olivier L et Mathern, Benoît, 2014. Connaissances et raisonnement sur les traces d'interaction. *Revue d'Intelligence Artificielle*, 28(2-3) :375–396.

Coulet, Jean-Claude, 2014. Des caractéristiques de l'expertise au management des compétences individuelles et collectives. *Management & Avenir*, (1) :122–135.

Cowan, Robin, David, Paul Allan et Foray, Dominique, 2000. The explicit economics of knowledge codification and tacitness. *Industrial and corporate change*, 9(2) :211–253.

Crié, Dominique, 2003. De l'extraction des connaissances au knowledge management. *Revue française de gestion*, (5) :59–79.

DAUCE, Emmanuel, 2002. *Traité de Sciences Cognitives : Approche dynamique de la cognition artificielle*. Hermès, Paris, France.
URL http://emmanuel.dauce.free.fr/Hermes/chap3.pdf

DE MULDER, Wim, BETHARD, Steven et MOENS, Marie-Francine, 2015. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, 30(1) :61–98.

DE SAUSSURE, Ferdinand, 1989. *Cours de linguistique générale : Édition critique*, tome 1. Otto Harrassowitz Verlag.

DEACON, S Hélène, CONRAD, Nicole et PACTON, Sébastien, 2008. A statistical learning perspective on children's learning about graphotactic and morphological regularities in spelling. *Canadian Psychology/Psychologie Canadienne*, 49(2) :118.

DELOCHE, François, 2017. Schéma d'un réseau de neurones récurrents à une unité reliant l'entrée et la sortie du réseau. a droite la version dépliée de la structure.
URL https://goo.gl/kc6vQs

DIENES, Zoltan et BERRY, Dianne, 1997. Implicit learning : Below the subjective threshold. *Psychonomic bulletin & review*, 4(1) :3–23.

DIENES, Zoltan, BROADBENT, Donald et BERRY, Dianne C, 1991. Implicit and explicit knowledge bases in artificial grammar learning. *Journal of Experimental Psychology : Learning, Memory, and Cognition*, 17(5) :875.

DIENES, Zoltan et PERNER, Josef, 1999. A theory of implicit and explicit knowledge. *Behavioral and Brain Sciences*, 22(5) :735–808.

DING, Ying et FOO, Schubert, 2002. Ontology research and development. part 1-a review of ontology generation. *Journal of information science*, 28(2) :123–136.

DUBUC, Bruno, 2002. MÉmoire et apprentissage.
URL http://lecerveau.mcgill.ca/flash/i/i_07/i_07_p/i_07_p_tra/i_07_p_tra.html

DUMAIS, Susan T, 1981. The development of automatism. *Cognitive Skills and Their Acquisition, Hillsdale, NJ : Erlbaum*, pages 111–140.

DURAND, Stéphane, 1995. *TOM, une architecture connexionniste de traitement de séquences : application à la reconnaissance de la parole*. Thèse de doctorat, Nancy 1.

El Hihi, Salah et Bengio, Yoshua, 1996. Hierarchical recurrent neural networks for long-term dependencies. Dans *Advances in neural information processing systems*, pages 493–499.

Elman, Jeffrey L, 1990. Finding structure in time. *Cognitive science*, 14(2) :179–211.

Ermine, Jean-Louis, 2001. Les processus de la gestion des connaissances. Dans *EGC*, pages 17–29.

Ettlinger, Marc, Margulis, Elizabeth H et Wong, Patrick CM, 2011. Implicit memory in music and language. *Frontiers in psychology*, 2.

Evans, Jonathan St BT, 2011. Dual-process theories of reasoning : Contemporary issues and developmental applications. *Developmental Review*, 31(2) :86–102.

Fang, Yu-Ching, Huang, Hsuan-Cheng, Chen, Hsin-Hsi et Juan, Hsueh-Fen, 2008. Tcmgenedit : a database for associated traditional chinese medicine, gene and disease information using text mining. *BMC complementary and alternative medicine*, 8(1) :58.

Fataicha, Youssef, 2005. *Recherche d'information dans les images de documents*. Thèse de doctorat, École de technologie supérieure.

Fayyad, Usama, Piatetsky-Shapiro, Gregory et Smyth, Padhraic, 1996. From data mining to knowledge discovery in databases. *AI magazine*, 17(3) :37.

Fischer, Flora, GIBON, Marie-Noel, Raffaelli AFFAELLI, Jean-Luc et Boutonnet, Christophe, 2015. Economie des données personnelles : les enjeux d'un business éthique.
URL http://www.cigref.fr/wp/wp-content/uploads/2015/11/CIGREF-Economie-donnees-perso-Enjeux-business-ethique-2015.pdf

Fitts, Paul M et Posner, Michael I, 1967. Human performance. brooks. *Cole, Belmont, CA*, 5 :7–16.

Frensch, Peter A et Stadler, Michael A, 1998. *Handbook of implicit learning*. Sage Publications.

Fritzke, Bernd, 1995. A growing neural gas network learns topologies. Dans *Advances in neural information processing systems*, pages 625–632.

Frosst, Nicholas et Hinton, Geoffrey, 2017. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv :1711.09784*.

GASPARINI, Silvia, 2004. Implicit versus explicit learning : Some implications for l2 teaching. *European Journal of Psychology of Education*, 19(2) :203–219.

GEGICK, Michael, ROTELLA, Pete et XIE, Tao, 2010. Identifying security bug reports via text mining : An industrial case study. Dans *Mining software repositories (MSR), 2010 7th IEEE working conference on*, pages 11–20. IEEE.

GELLY, Grégory, 2017. *Réseaux de neurones récurrents pour le traitement automatique de la parole*. Thèse de doctorat, Université Paris-Saclay.

GERS, Felix, 2001. Long short-term memory in recurrent neural networks. *Unpublished PhD dissertation, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland*.

GERS, Felix A et SCHMIDHUBER, E, 2001. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6) :1333–1340.

GERS, Felix A et SCHMIDHUBER, Jürgen, 2000. Recurrent nets that time and count. Dans *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, tome 3, pages 189–194. IEEE.

GERS, Felix A, SCHMIDHUBER, Jürgen et CUMMINS, Fred, 1999. Learning to forget : Continual prediction with lstm.

GILES, C. L., MILLER, C. B., CHEN, D., CHEN, H. H., SUN, G. Z. et LEE, Y. C., 1992a. Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4(3) :393–405. doi : 10.1162/neco.1992.4.3.393.
URL https://doi.org/10.1162/neco.1992.4.3.393

GILES, C Lee, CHEN, D, MILLER, CB, CHEN, HH, SUN, GZ et LEE, YC, 1991. Second-order recurrent neural networks for grammatical inference. Dans *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, tome 2, pages 273–281. IEEE.

GILES, C Lee, MILLER, Clifford B, CHEN, Dong, CHEN, Hsing-Hen, SUN, Guo-Zheng et LEE, Yee-Chun, 2008. Learning and extracting finite state automata with second-order recurrent neural networks. *Learning*, 4(3).

GILES, C Lee, MILLER, Clifford B, CHEN, Dong, SUN, Guo-Zheng, CHEN, Hsing-Hen et LEE, Yee-Chun, 1992b. Extracting and learning an unknown grammar with recurrent neural networks. Dans *Advances in neural information processing systems*, pages 317–324.

GOLAB, Lukasz, KARLOFF, Howard, KORN, Flip, SAHA, Avishek et SRIVASTAVA, Divesh, 2009. Sequential dependencies. *Proceedings of the VLDB Endowment*, 2(1) :574–585.

GOMBERT, Jean Emile, 2006. Epi/méta vs implicite/explicite : niveau de contrôle cognitif sur les traitements et apprentissage de la lecture. *Langage & pratiques*, 38 :68–76.

GOYAL, Shubham, 2017. Review paper on sentiment analysis of twitter data using text mining and hybrid classification approach.

GRAVES, Alex et SCHMIDHUBER, Jürgen, 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5) :602–610.

GREFF, Klaus, SRIVASTAVA, Rupesh K, KOUTNÍK, Jan, STEUNEBRINK, Bas R et SCHMIDHUBER, Jürgen, 2017. Lstm : A search space odyssey. *IEEE transactions on neural networks and learning systems*.

GRUNDSTEIN, Michel, 1995. La capitalisation des connaissances de l'entreprise, système de production des connaissances. Dans *Actes du Colloque L'Entreprise Apprenante et les Sciences de la Complexité, Aix en Provence, France*, pages 22–24.

HADDAD, Hatem et MULHEM, Philippe, 2001. Utilisation de la fouille de données images pour l'indexation automatique des images. Dans *INFORSID*, pages 405–418.

HAMMING, Richard W, 1950. Error detecting and error correcting codes. *Bell Labs Technical Journal*, 29(2) :147–160.

HE, Wu, ZHA, Shenghua et LI, Ling, 2013. Social media competitive analysis and text mining : A case study in the pizza industry. *International Journal of Information Management*, 33(3) :464–472.

HEES, Jörn, 2015. SciPy hierarchical clustering and dendrogram tutorial. URL https://joernhees.de/blog/2015/08/26/scipy-hierarchical-clustering-and-dendrogram-tutorial/

HINAUT, Xavier, 2013. *Réseau de neurones récurrent pour le traitement de séquences abstraites et de structures grammaticales, avec une application aux interactions homme-robot*. Thèse de doctorat, Université Claude Bernard Lyon 1.

HOCHREITER, Sepp, BENGIO, Yoshua, FRASCONI, Paolo, SCHMIDHUBER, Jürgen *et al.*, 2001. Gradient flow in recurrent nets : the difficulty of learning long-term dependencies.

HOCHREITER, Sepp et SCHMIDHUBER, Jürgen, 1997. Long short-term memory. *Neural computation*, 9(8) :1735–1780.

HOFFMAN, Robert R, 2014. *The psychology of expertise : Cognitive research and empirical AI*. Psychology Press.

HOPCROFT, JE *et al.*, 1979. Ullman. jd (1979) introduction to automata theory, languages, and computation.

HOPCROFT, John, 1971. An/n log n algorithm for minimizing states in kf in ite automaton.

HOPFIELD, J J, 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8) :2554–2558.
URL http://www.pnas.org/content/79/8/2554.abstract

JACCARD, Paul, 1901. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37 :547–579.

JACOBSSON, Henrik, 2005. Rule extraction from recurrent neural networks : Ataxonomy and review. *Neural Computation*, 17(6) :1223–1263.

JAEGER, Herbert, 2002. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach*, tome 5. GMD-Forschungszentrum Informationstechnik.

JIN, Xiaolong, WAH, Benjamin W, CHENG, Xueqi et WANG, Yuanzhuo, 2015. Significance and challenges of big data research. *Big Data Research*, 2(2) :59–64.

JORDAN, MI, 1986. Serial order : a parallel distributed processing approach. technical report, june 1985-march 1986. Rapport technique, California Univ., San Diego, La Jolla (USA). Inst. for Cognitive Science.

KARPATHY, Andrej, JOHNSON, Justin et FEI-FEI, Li, 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv :1506.02078*.

KEMBHAVI, Aniruddha, SALVATO, Mike, KOLVE, Eric, SEO, Minjoon, HAJISHIRZI, Hannaneh et FARHADI, Ali, 2016. A diagram is worth a dozen images.
URL http://arxiv.org/abs/1603.07396

KHOUZAM, Bassem, 2014. *Neural networks as cellular computing models for temporal sequence processing.* Thèse de doctorat, Supélec.

KIM, Jong W, RITTER, Frank E et KOUBEK, Richard J, 2013. An integrated theory for improved skill acquisition and retention in the three stages of learning. *Theoretical Issues in Ergonomics Science*, 14(1) :22–37.

KOHONEN, Teuvo, 1982. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1) :59–69.

KORCZAK, J, SCHEIBER, C, HOMMET, J et LACHICHE, N, 2005. Fouille interactive en temps réel de séquences d'images irmf.

KRUGER, Norbert, JANSSEN, Peter, KALKAN, Sinan, LAPPE, Markus, LEONARDIS, Ales, PIATER, Justus, RODRIGUEZ-SANCHEZ, Antonio J et WISKOTT, Laurenz, 2013. Deep hierarchies in the primate visual cortex : What can we learn for computer vision ? *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1847–1871.

LAPALME, Jasmin, 2006. Composition automatique de musique à l'aide de réseaux de neurones récurrents et de la structure métrique.

LAWRENCE, Steve, GILES, C Lee et FONG, Sandiway, 2000. Natural language grammatical inference with recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 12(1) :126–140.

LEE, Honglak, EKANADHAM, Chaitanya et NG, Andrew Y, 2008. Sparse deep belief net model for visual area v2. Dans *Advances in neural information processing systems*, pages 873–880.

LELEBINA, Olga, 2014. *Management of experts in industrial companies : dynamics of internal professional communities and career trajectories*. Theses, Ecole Nationale Supérieure des Mines de Paris.
URL https://pastel.archives-ouvertes.fr/tel-01123778

LÉTÉ, Bernard, 2006. L'apprentissage implicite des régularités statistiques de la langue et l'acquisition des unités morphosyntaxiques. *Langue française*, (3) :41–58.

LIPTON, Zachary C, BERKOWITZ, John et ELKAN, Charles, 2015. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv :1506.00019*.

LOGAN, Gordon D, 1988. Automaticity, resources, and memory : Theoretical controversies and practical implications. *Human factors*, 30(5) :583–598.

LOWE, Richard K., 1993. Constructing a mental representation from an abstract technical diagram. *Learning and Instruction*, 3(3) :157–179.
URL http://www.sciencedirect.com/science/article/pii/095947529390002H

MALHOTRA, Yogesh, 2000. Knowledge management and new organization forms : A framework for business model innovation. *Knowledge management and virtual organizations*, 2(1) :13–27.

MARSLAND, Stephen, SHAPIRO, Jonathan et NEHMZOW, Ulrich, 2002. A self-organising network that grows when required. *Neural networks*, 15(8) :1041–1058.

MARTINEZ, Régis, 2011. *Dynamique des systèmes cognitifs et des systèmes complexes*. Thèse de doctorat, Université Lumière Lyon 2.

MATHERN, Benoît, 2012. *Découverte interactive de connaissances à partir de traces d'activité : Synthèse d'automates pour l'analyse et la modélisation de l'activité de conduite automobile*. Thèse de doctorat, Université Claude Bernard-Lyon I.

MEULEMANS, Thierry, 1998. Apprentissage implicite, mémoire implicite et développement. *Psychologie Française*, 43(1) :27–37.

MOONEY, Raymond J et BUNESCU, Razvan, 2005. Mining knowledge from text using information extraction. *ACM SIGKDD explorations newsletter*, 7(1) :3–10.

MOSTAFA, Mohamed M, 2013. More than words : Social networks' text mining for consumer brand sentiments. *Expert Systems with Applications*, 40(10) :4241–4251.

MOZER, Michael C, KINOSHITA, Sachiko, SHETTEL, Michael *et al.*, 2007. Sequential dependencies in human behavior offer insights into cognitive control. *Integrated models of cognitive systems*, pages 180–193.

MURDOCH, W James et SZLAM, Arthur, 2017. Automatic rule extraction from long short term memory networks. *arXiv preprint arXiv :1702.02540*.

NADEAU, Marie et FISHER, Carole, 2011. Les connaissances implicites et explicites en grammaire : quelle importance pour l'enseignement ? quelles conséquences ? *Bellaterra journal of teaching and learning language and literature*, 4(4) :0001–31.

NASSIRTOUSSI, Arman Khadjeh, AGHABOZORGI, Saeed, WAH, Teh Ying et NGO, David Chek Ling, 2014. Text mining for market prediction : A systematic review. *Expert Systems with Applications*, 41(16) :7653–7670.

NASSIRTOUSSI, Arman Khadjeh, AGHABOZORGI, Saeed, WAH, Teh Ying et NGO, David Chek Ling, 2015. Text mining of news-headlines for forex market prediction : A multi-layer dimension reduction algorithm with semantics and sentiment. *Expert Systems with Applications*, 42(1) :306–324.

Neto, Joel Larocca, Santos, Alexandre D., Kaestner, Celso A.A., Alexandre, Neto, Santos, D., A, Celso A., Alex, Kaestner, Freitas, Alex A. et Parana, Catolica, 2000. Document clustering and text summarization.

Neves, David M et Anderson, John R, 1981. Knowledge compilation : Mechanisms for the automatization of cognitive skills. *Cognitive skills and their acquisition*, pages 57–84.

Nissen, Mary Jo et Bullemer, Peter, 1987. Attentional requirements of learning : Evidence from performance measures. *Cognitive psychology*, 19(1) :1–32.

Nonaka, Ikujiro et Takeuchi, Hirotaka, 1995. *The knowledge-creating company : How Japanese companies create the dynamics of innovation*. Oxford university press.

Omlin, Christian W et Giles, C Lee, 1996a. Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM (JACM)*, 43(6) :937–972.

Omlin, Christian W et Giles, C Lee, 1996b. Extraction of rules from discrete-time recurrent neural networks. *Neural networks*, 9(1) :41–52.

Omlin, Christian W et Giles, C Lee, 2000. Symbolic knowledge representation in recurrent neural networks : Insights from theoretical models of computation. *Knowledge based neurocomputing*, pages 63–115.

O'Neill, Catherine, 2016. Weapons of math destruction. *How Big Data Increases Inequality and Threatens Democracy*.

O'Reilly, Randall C et Frank, Michael J, 2006. Making working memory work : a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*, 18(2) :283–328.

Oudeyer, Pierre-Yves, 2009. L'auto-organisation dans l'évolution de la parole.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. et Duchesnay, E., 2011. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830.

Perruchet, Pierre et Nicolas, Serge, 1998. L'apprentissage implicite : un débat théorique. *Psychologie française*, 43 :13–26.

PERRUCHET, Pierre et PACTON, Sebastien, 2004. Qu'apportent à la pédagogie les travaux de laboratoire sur l'apprentissage implicite ? *L'année Psychologique*, 104(1) :121–146.

PERRUCHET, Pierre et VINTER, Annie, 1998. Learning and development : The implicit knowledge assumption reconsidered.

PHAM, Nguyen-Khang, MORIN, Annie, POULET, François et GROS, Patrick, 2011. Analyse factorielle des correspondances hiérarchique pour la fouille d'images. Dans *EGC*, pages 161–172.

POLANYI, Michael, 2009. *The tacit dimension*. University of Chicago press.

PRAHALAD, Coimbatore K et HAMEL, Gary, 1999. The core competence of the corporation. Dans *Knowledge and strategy*, pages 41–59. Elsevier.

RAMOS, Sérgio, FIGUEIREDO, Vera, RODRIGUES, Fátima, PINHEIRO, Raul et VALE, Zita, 2007. Knowledge extraction from medium voltage load diagrams to support the definition of electrical tariffs. *Engineering Intelligent Systems for Electrical Engineering and Communications*, 15(3) :143–149.

RASMUSSEN, Jens, 1986. Information processing and human-machine interaction. an approach to cognitive engineering.

REBER, Arthur S, 1967. Implicit learning of artificial grammars. *Journal of verbal learning and verbal behavior*, 6(6) :855–863.

REBER, Arthur S, 1993. Oxford psychology series, no. 19. implicit learning and tacit knowledge : An essay on the cognitive unconscious.

REBER, Paul J, 2013. The neural basis of implicit learning and memory : a review of neuropsychological and neuroimaging research. *Neuropsychologia*, 51(10) :2026–2042.

REMM, Jean-François, 1996. *Extraction de connaissances par réseaux neuronaux : application au domaine du radar*. Thèse de doctorat, Nancy 1.

ROCHE, Mathieu, 2011. *Fouille de Textes : de l'extraction des descripteurs linguistiques à leur induction*. Habilitation à diriger des recherches, Université Montpellier II - Sciences et Techniques du Languedoc.
URL https://tel.archives-ouvertes.fr/tel-00816263

ROSENBLATT, Frank, 1958. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386.

ROSENBLUM, Nathan, ZHU, Xiaojin et MILLER, Barton, 2011. Who wrote this code ? identifying the authors of program binaries. *Computer Security–ESORICS 2011*, pages 172–189.

ROUGIER, Nicolas P, 2000. *Modeles de mémoires pour la navigation autonome*. Thèse de doctorat, Université Henri Poincaré-Nancy I.

RUMELHART, David E, HINTON, Geoffrey E et WILLIAMS, Ronald J, 1985. Learning internal representations by error propagation. Rapport technique, California Univ San Diego La Jolla Inst for Cognitive Science.

SAMUELIDES, M et CESSAC, Bruno, 2007. Random recurrent neural networks dynamics. *The European Physical Journal-Special Topics*, 142(1) :89–122.

SCHACTER, Daniel L, 1987. Implicit memory : History and current status. *Journal of experimental psychology : learning, memory, and cognition*, 13(3) :501.

SERROUR, Belkacem, DJOUADI, Slimane et MILLE, Alain, 2006. Gestion des connaissances.

SERVAN-SCHREIBER, D, CLEERMANS, A et MCCLELLAND, JL, 1988. Encoding sequential structure in simple recurrent networks. *School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, CMU-CS-88-183.(November, 1988)*.

SERVAN-SCHREIBER, David, CLEEREMANS, Axel et MCCLELLAND, James L, 1989. Learning sequential structure in simple recurrent networks. Dans *Advances in neural information processing systems*, pages 643–652.

SERVAN-SCHREIBER, David, CLEEREMANS, Axel et MCCLELLAND, James L, 1991. Graded state machines : The representation of temporal contingencies in simple recurrent networks. *Machine Learning*, 7(2) :161–193.

SINGHAL, Amit, 2001. Modern information retrieval : A brief overview. *IEEE Data Eng. Bull.*, 24(4) :35–43.

SMITH, Andrew, 2003. Grammar inference using recurrent neural networks. *Department of Computer Science, University of San Diego, California, www. cse. ucsd. edu/˜ atsmith*.

SMITH, Elizabeth A, 2001. The role of tacit and explicit knowledge in the workplace. *Journal of knowledge Management*, 5(4) :311–321.

SOKAL, Robert R et ROHLF, F James, 1962. The comparison of dendrograms by objective methods. *Taxon*, 11(2) :33–40.

Spiekermann, Sarah, Acquisti, Alessandro, Böhme, Rainer et Hui, Kai-Lung, 2015. The challenges of personal data markets and privacy. *Electronic Markets*, 25(2) :161–167.

Squire, Larry R, 1992. Declarative and nondeclarative memory : Multiple brain systems supporting learning and memory. *Journal of cognitive neuroscience*, 4(3) :232–243.

Squire, Larry R et Zola, Stuart M, 1996. Structure and function of declarative and nondeclarative memory systems. *Proceedings of the National Academy of Sciences*, 93(24) :13515–13522.

Strobelt, Hendrik, Gehrmann, Sebastian, Huber, Bernd, Pfister, Hanspeter et Rush, Alexander M, 2016. Visual analysis of hidden state dynamics in recurrent neural networks. *arXiv preprint arXiv :1606.07461*.

Sutton, Richard S, 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1) :9–44.

Tan, Ah-Hwee *et al.*, 1999. Text mining : The state of the art and the challenges. Dans *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*, tome 8, pages 65–70. sn.

Teulier, Régine, Charlet, Jean et Tchounikine, Pierre, 2005. L'ingénierie des connaissances : acquis et nouvelles perspectives. Dans *Ingénierie des connaissances*, pages 11–26. L'Harmattan.

Touzet, Claude, 1992. *les réseaux de neurones artificiels, introduction au connexionnisme*. EC2.

VanLehn, Kurt, 1996. Cognitive skill acquisition. *Annual review of psychology*, 47(1) :513–539.

Viola, Jean-Michel, 2005. La gestion des transferts de connaissances entre générations.

Wang, Qinglong, Zhang, Kaixuan, Ororbia, II, Alexander, G, Xing, Xinyu, Liu, Xue et Giles, C Lee, 2017. An empirical evaluation of recurrent neural network rule extraction. *arXiv preprint arXiv :1709.10380*.

Weiss, Gail, Goldberg, Yoav et Yahav, Eran, 2017. Extracting automata from recurrent neural networks using queries and counterexamples. *arXiv preprint arXiv :1711.09576*.

Williams, Ronald J. et Zipser, David, 1989. Experimental analysis of the real-time recurrent learning algorithm. *Connection Science*, 1(1) :87–111. doi :10.1080/09540098908915631.
URL https://doi.org/10.1080/09540098908915631

WIRTZ, Angelika, KRON, Wolfgang, LÖW, Petra et STEUER, Markus, 2014. The need for data : natural disasters and the challenges of database management. *Natural Hazards*, 70(1) :135–157.

WITT, Arnaud, 2010a. *Artificial grammar implicit learning in children with and without mental retardation : role of the material's properties and influence of the instructions*. Theses, Université de Bourgogne.
URL https://tel.archives-ouvertes.fr/tel-00651503

WITT, Arnaud, 2010b. *L'apprentissage implicite d'une grammaire artificielle chez l'enfant avec et sans retard mental : rôle des propriétés du matériel et influence des instructions*. Thèse de doctorat, Université de Bourgogne.

WITTEN, Ian H, FRANK, Eibe, HALL, Mark A et PAL, Christopher J, 2016. *Data Mining : Practical machine learning tools and techniques*. Morgan Kaufmann.

YIN, Wenpeng, KANN, Katharina, YU, Mo et SCHÜTZE, Hinrich, 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv :1702.01923*.

ZAOUI, Brahim et MEBARKI, Bouhafs, 2017. Les nouvelles technologies de communication et de l'information, quel impact sur le travail et son organisation ?-approche ergonomique.

ZENG, Zheng, GOODMAN, Rodney M et SMYTH, Padhraic, 1993. Learning finite state machines with self-clustering recurrent networks. *Learning*, 5(6).

ZHANG, Ji, HSU, Wynne et LEE, Mong Li, 2001. Image mining : Issues, frameworks and techniques. Dans *Proceedings of the Second International Conference on Multimedia Data Mining*, pages 13–20. Springer-Verlag.