Traitement automatique des langues

Varia

sous la direction de
Jean-Luc Minel
Emmanuel Morin
Pascale Sébillot
Isabelle Tellier

Vol. 58 - n°1 / 2017

ATALA

Varia

Jean-Luc Minel, Emmanuel Morin, Pascale Sébillot, Isabelle Tellier
Préface

Marco Dinarelli, Yoann Dupont
Modélisation de dépendances entre étiquettes dans les réseaux neuronaux récurrents

Rémy Kessler, Guy Lapalme
AGOHRA : génération d'une ontologie dans le domaine des ressources humaines

Cynthia Van Hee, Marjan Van de Kauter, Orphée De Clercq Els Lefever, Bart Desmet, Véronique Hoste
Noise or music? Investigating the usefulness of normalisation for robust sentiment analysis on social media data

Denis Maurel
Notes de lecture

ATALA

Revue de
l'Association
pour le Traitement
Automatique
des Langues

TAL
Vol.
58

n°1
2017

Varia

ATALA

Traitement automatique des langues

Revue publiée depuis 1960 par l'Association pour le Traitement Automatique des LAngues (ATALA), avec le concours du CNRS, de l'Université Paris VII et de l'Université de Provence

ISSN 1965-0906

https ://www.atala.org/revuetal

Traitement automatique des langues

Aurélie Névéol - LIMSI, CNRS

Traitement automatique des langues

Volume 58 – n°1 / 2017

VARIA

Table des matières

Préface
Jean-Luc Minel, Emmanuel Morin, Pascale Sébillot, Isabelle Tellier 7

Modélisation de dépendances entre étiquettes dans les réseaux neuronaux récurrents
Marco Dinarelli, Yoann Dupont 13

AGOHRA : génération d'une ontologie dans le domaine des ressources humaines
Rémy Kessler, Guy Lapalme 39

Noise or music ? Investigating the usefulness of normalisation for robust sentiment analysis on social media data
Cynthia Van Hee, Marjan Van de Kauter, Orphée De Clercq Els Lefever, Bart Desmet, Véronique Hoste 63

Notes de lecture
Denis Maurel 89

Préface

Comme il est de coutume, cette préface de numéro non thématique donne des nouvelles de notre comité de rédaction, mentionne les progrès dans la gestion de notre revue et fournit des statistiques sur les articles soumis et publiés, avant de présenter brièvement les articles du numéro.

Dans la rubrique “vie du comité de rédaction”, le premier auteur de ces lignes – Jean-Luc Minel – a quitté ses fonctions de rédacteur en chef ainsi que le comité de rédaction en septembre 2017. Sophie Rosset, directrice de recherche au CNRS, en poste au LIMSI, est la nouvelle rédactrice en chef.

Certains évènements récents nous ont poussés à revoir ou préciser certaines de nos procédures. En particulier, une longueur maximale des articles soumis avait été adoptée dès 2012 mais avait été quelque peu oubliée depuis. Nos conseils aux auteurs mentionnent donc maintenant explicitement une longueur comprise entre vingt et vingt-cinq pages.

Un très grand pas en avant a été accompli dans la production automatique des numéros sous forme d’un seul fichier PDF incluant couverture et table des matières. La procédure est maintenant maîtrisée. Les prochains numéros seront produits par Sophie Rosset et Maxime Amblard.

Un premier travail actuel vise à augmenter la visibilité de la revue et à permettre le moissonnage automatique de nos métadonnées par les sites de référencement. Il passera sans doute par la mise en place d’un dépôt de ces métadonnées visible par les moteurs d’indexation. Un second travail est en cours afin d’utiliser une plate-forme intégrée de gestion des numéros qui couvrirait tout le cycle de vie d’un numéro de la revue, de l’appel à soumission jusqu’à la production du numéro au format PDF. Depuis plusieurs années, nous utilisons la plate-forme SciencesConf du CCSD. Son usage nous a permis de rationaliser et d’améliorer la procédure de relecture des articles soumis, mais elle présente plusieurs limitations liées à sa finalité première qui est de gérer des appels à communications pour des colloques et non pour une revue. Il existe plusieurs plates-formes de gestion de revues, comme Open Journal Systems, mais celles-ci exigent généralement un important travail d’administration pris en charge par un secrétaire de rédaction. Le comité de rédaction étudie les fonctionnalités de la plate-forme Episciences, développée par le CCSD, qui repose sur un cycle de vie initié par un dépôt dans les archives ouvertes HAL.

Pour ce qui est de la régularité de publication, nous pouvons là aussi nous montrer satisfaits de l'établissement d'un calendrier prévisionnel qui nous permet de caler les différents numéros d'un même volume et de tenir de façon plus régulière les réunions du comité de rédaction. Rappelons que l'une des caractéristiques de notre revue, à laquelle nous sommes foncièrement attachés, est la tenue des réunions du comité de rédaction au cours desquelles nous décidons collégialement, à l'appui des relectures reçues, de l'acceptation ou du rejet des articles soumis.

Passons maintenant à nos statistiques. Elles considèrent toujours les dix derniers numéros sur les trois dernières années, en l'occurrence donc, du début 2014 jusqu'à ce numéro *Varia* de 2017 inclus. Le tableau 1 donne les taux de sélection par numéro et par volume. La ligne du total synthétise ces chiffres sur l'ensemble des dix numéros considérés.

Le taux de sélection sur l'ensemble de ces numéros s'élève à 34,4 % en moyenne, c'est-à-dire que, sur trois articles soumis, un est accepté. Ce taux est stable dans le temps, entre 31 et 35 %, d'après les chiffres donnés depuis le numéro 51-1. Notre comité de rédaction est très attaché à sélectionner les articles selon leur qualité, indépendamment du nombre d'articles soumis. Or on peut observer que ce nombre fluctue. L'un de nos soucis actuels est de nous assurer d'un nombre stable de soumissions, ce qui n'a pas été le cas pour le dernier numéro *Varia*.

Les statistiques que nous donnons sur l'origine des articles considèrent le pays du premier auteur, hors de France ou pas, ainsi que la langue de la soumission, français en principe ou anglais si l'un des coauteurs n'est pas francophone. Les chiffres sont fournis dans le tableau 2 pour la même période de temps que le tableau 1. En comparant aux chiffres des derniers numéros *Varia*, on constate une diminution récente d'articles acceptés en anglais alors que le nombre de soumissions dans cette langue ne baisse pas vraiment (chiffres non communiqués ici). Il en va de même pour le nombre de premiers auteurs hors de France, qui est tombé à 0 durant tout le volume de 2015, pour remonter à un article sur trois dans ce *Varia*.

Ce numéro contient les articles retenus lors de l'appel non thématique lancé début octobre et clos à la mi-novembre 2016. Cet appel portait comme d'habitude sur tous les aspects du traitement automatique des langues. Huit articles ont été soumis dont un en anglais, ce qui représente une baisse notable des soumissions.

À l'issue du processus de sélection habituel à deux tours, trois articles, ont été retenus pour publication, dont un en anglais. Les tâches abordées par ces articles sont diverses : exploitation des réseaux neuronaux récurrents pour modéliser l'étiquetage de séquences et les dépendances entre étiquettes ; génération automatique d'une ontologie à partir d'un corpus dans le domaine de l'e-recrutement ; étude de l'impact de la normalisation de données bruitées issues de réseaux sociaux pour l'analyse de sentiments :

1) « Modélisation de dépendances entre étiquettes dans les réseaux neuronaux récurrents », Marco Dinarelli et Yoann Dupont ;

Intitulé	Vol.	N°	Année	Soumis	Acceptés	% acceptés
Varia	55	1	2014	19	5	26,3 %
Trait. auto. langage parlé	55	2	2014	9	6	66,6 %
TAL et sciences cognitives	55	3	2014	7	4	57,1 %
Sous-total	**55**		**2014**	**35**	**15**	**42,9 %**
Varia	56	1	2015	20	4	20,0 %
Sémantique distributionnelle	56	2	2015	7	4	57,1 %
Recherche d'information	56	3	2015	12	3	25,0 %
Sous-total	**56**		**2015**	**39**	**11**	**28,2 %**
Varia	57	1	2016	19	5	26,3 %
TAL et éthique	57	2	2016	7	3	42,9 %
TALP et didactique	57	3	2016	14	5	35,7 %
Sous-total	**57**		**2016**	**40**	**13**	**32,5 %**
Varia	58	1	2017	8	3	37,5 %
Total	**Dix derniers n^{os}**			**122**	**42**	**34,4 %**

Tableau 1. *Taux de sélection aux appels de la revue TAL sur les dix derniers numéros de la période 2014-2017*

Intitulé	Vol.	N°	Année	% 1er auteur hors France	% en anglais
Varia	55	1	2014	0,0 %	0,0 %
Trait. auto. langage parlé	55	2	2014	16,7 %	0,0 %
TAL et sciences cognitives	55	3	2014	25,0 %	50,0 %
Pourcentages par volume	**55**		**2014**	**13,9 %**	**16,7 %**
Varia	56	1	2015	0,0 %	0,0 %
Sémantique distributionnelle	56	2	2015	0,0 %	0,0 %
Recherche d'information	56	3	2015	0,0 %	0,0 %
Pourcentages par volume	**56**		**2015**	**0,0 %**	**0,0 %**
Varia	57	1	2016	20,0 %	20,0 %
TAL et éthique	57	2	2016	0,0 %	0,0 %
TALP et didactique	57	3	2016	80,0 %	40,0 %
Pourcentages par volume	**57**		**2016**	**33,3 %**	**20,0 %**
Varia	58	1	2017	33,3 %	33,3 %
Pourcentages totaux	**Dix derniers n^{os}**			**20,1 %**	**17,5 %**

Tableau 2. *Proportion des articles publiés d'un premier auteur hors de France et proportion des articles publiés rédigés en anglais sur les dix derniers numéros de la période 2014-2017. Attention, les pourcentages totaux ne sont pas de simples moyennes des chiffres donnés plus haut, car les dénominateurs changent.*

2) « AGOHRA : Génération d'une ontologie dans le domaine des ressources humaines », Guy Lapalme et Rémy Kessler ;

3) « Noise or Music ? Investigating the Usefulness of Normalisation for Robust Sentiment Analysis on Social Media Data », Cynthia Van Hee, Marjan Van de Kauter, Orphée De Clercq, Els Lefever, Bart Desmet et Véronique Hoste.

On trouvera en suite des articles des notes de lecture. Nous encourageons nos lecteurs à se faire mutuellement profiter de leurs lectures et à se mettre en contact avec Denis Maurel (`denis.maurel@univ-tours.fr`) pour les publier ici. Suit une liste de résumés de thèses ou d'habilitations à diriger les recherches en traitement automatique des langues. Merci à Denis Maurel et Sylvain Pogodalla pour leur travail de veille et de collecte.

Enfin, rappelons que la revue TAL reçoit un soutien financier de l'Institut des sciences humaines et sociales du CNRS et de la Délégation générale à la langue française et aux langues de France (DGLFLF). Nous adressons nos remerciements à ces organismes.

Jean-Luc Minel
MoDyCo, université Paris-Nanterre,
Jean-Luc.Minel@parisnanterre.fr

Emmanuel Morin
LS2N, université de Nantes
Emmanuel.Morin@univ-nantes.fr

Pascale Sébillot
IRISA, INSA Rennes
pascale.sebillot@irisa.fr

Isabelle Tellier
Lattice, université Paris 3
isabelle.tellier@sorbonne.nouvelle.fr

Merci aux relecteurs spécifiques de ce numéro :

Aussenac Nathalie, IRIT, CNRS
Despres Sylvie, LIMICS, université Paris13
Dinarelli Marco, Lattice, CNRS
Dymetman Marc, Naver Labs Europe
Gaussier Eric, LIG, université Grenoble Alpes
Philippe Langlais, RALI, université de Montréal
Lecorvé Gwénolé, IRISA, université de Rennes 1
Tannier Xavier, LIMICS, Sorbonne université

ainsi qu'aux membres du comité de rédaction de la revue (voir sa composition sur notre site).

Modélisation de dépendances entre étiquettes dans les réseaux neuronaux récurrents

Marco Dinarelli*, Yoann Dupont***

* *Lattice (UMR 8094), CNRS, ENS Paris, Université Sorbonne Nouvelle - Paris 3*
PSL Research University, USPC (Université Sorbonne Paris Cité)
1 rue Maurice Arnoux, 92120 Montrouge, France
** *Expert System France, 207 rue de Bercy, 75012 Paris*
marco.dinarelli@ens.fr, yoa.dupont@gmail.com

RÉSUMÉ. Les réseaux neuronaux récurrents (RNN) se sont montrés très efficaces dans plusieurs tâches de traitement automatique des langues. Cependant, leur capacité à modéliser *l'étiquetage de séquences* reste limitée. Cette limitation a conduit la recherche vers la combinaison des RNN avec des modèles déjà utilisés avec succès dans ce contexte, comme les CRF. Dans cet article, nous étudions une solution plus simple mais tout aussi efficace : une évolution du RNN de Jordan dans lequel les étiquettes prédites sont réinjectées comme entrées dans le réseau et converties en plongements, de la même façon que les mots. Nous comparons cette variante de RNN avec tous les autres modèles existants : Elman, Jordan, LSTM et GRU, sur deux tâches de compréhension de la parole. Les résultats montrent que la nouvelle variante, plus complexe que les modèles d'Elman et Jordan, mais bien moins que LSTM et GRU, est non seulement plus efficace qu'eux, mais qu'elle fait aussi jeu égal avec des modèles CRF plus sophistiqués.

ABSTRACT. Recurrent Neural Networks have proved effective on several NLP tasks. Despite such great success, their ability to model *sequence labeling* is still limited. This lead research toward solutions where RNNs are combined with models successfully employed in this context, like CRFs. In this work we propose a simpler solution: an evolution of the Jordan RNN, where labels are reinjected as input into the network and converted into embeddings, the same way as words. We compare this variant to all the other RNN models, Elman, Jordan, LSTM and GRU, on two tasks of Spoken Language Understanding. Results show that the new variant, which is more complex than Elman and Jordan RNNs, but far less than LSTM and GRU, is more effective than other RNNs and outperforms sophisticated CRF models.

MOTS-CLÉS : *RNN, modélisation de séquences, compréhension automatique de la parole.*

KEYWORDS: *RNNs, sequence modelling, spoken language understanding.*

1. Introduction

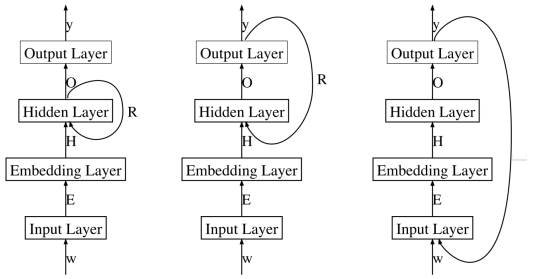
Dans les dernières années les réseaux neuronaux récurrents (RNN) (Jordan, 1989 ; Elman, 1990 ; Hochreiter et Schmidhuber, 1997) se sont montrés très efficaces dans plusieurs tâches de traitement automatique des langues (TAL) comme l'étiquetage des parties du discours, la segmentation, la reconnaissance des entités nommées, la compréhension automatique de la parole, la traduction automatique statistique et d'autres encore (Mikolov *et al.*, 2010 ; Mikolov *et al.*, 2011 ; Collobert et Weston, 2008 ; Collobert *et al.*, 2011 ; Yao *et al.*, 2013 ; Mesnil *et al.*, 2013 ; Vukotic *et al.*, 2015). L'efficacité de ces modèles vient de leur architecture récurrente, qui permet de garder une mémoire de l'information traitée dans le passé et de la réutiliser à l'étape courante.

Dans la littérature, plusieurs types d'architectures ont été proposés, notamment les RNN d'Elman (Elman, 1990) et ceux de Jordan (Jordan, 1989), aussi appelés RNN simples. La différence entre ces deux modèles réside simplement dans la connexion qui donne son caractère récurrent au réseau de neurones : dans les RNN d'Elman, la boucle se situe au niveau de la couche cachée, alors qu'elle se trouve entre la couche de sortie et la couche cachée dans les RNN de Jordan. Dans ce dernier cas, elle permet d'utiliser à l'étape courante les étiquettes prédites pour les positions précédentes d'une séquence.

Ces deux types de modèles sont limités pour l'encodage de contextes relativement longs (Bengio *et al.*, 1994). Pour cette raison les RNN de type *Long Short-Term Memory* (LSTM) ont été proposés (Hochreiter et Schmidhuber, 1997). Récemment, une variante simplifiée et plus efficace des LSTM a été introduite, utilisant des *Gated Recurrent Units*, d'où son nom : GRU (Cho *et al.*, 2014).

Malgré leur relative efficacité sur plusieurs tâches, les RNN présentent toujours des limitations dans leur capacité à modéliser les dépendances entre les unités de sortie, c'est-à-dire les étiquettes. Leurs fonctions de décision restent en effet intrinsèquement locales. Pour pallier cette limitation, des modèles hybrides *RNN+CRF* ont été testés (Huang *et al.*, 2015 ; Lample *et al.*, 2016 ; Ma et Hovy, 2016). Ces modèles atteignent des résultats à l'état de l'art, et bien que leur efficacité soit indéniable, il n'est pas clair qu'elle soit due aux modèles eux-mêmes ou aux conditions expérimentales particulièrement favorables dans lesquelles ils ont été évalués (notamment grâce à des plongements de mots entraînés sur de très grandes quantités de données avec des outils tels que *word2vec* (Mikolov *et al.*, 2013a) ou *GloVe* (Pennington *et al.*, 2014)).

L'intuition à la base de cet article est que les plongements permettent une modélisation plus efficace non seulement des mots, mais également des *étiquettes* et de leurs dépendances, qui sont cruciales dans certaines tâches d'étiquetage de séquences. Dans cet article, nous étudions une variante favorisant cette modélisation. Cette variante redonne en entrée du réseau les étiquettes prédites aux étapes précédentes du traitement d'une séquence, en plus de l'entrée habituelle constituée d'un ou plusieurs mots. Puisque l'entrée du réseau est constituée d'indices qui sont utilisés pour sélectionner des plongements, dans notre variante les étiquettes sont converties en plongements de la même façon que les mots. Nous espérons, grâce à l'utilisation des plongements



(a) Elman (b) Jordan (c) Notre variante

Figure 1. *Schéma général des principaux RNN et de notre variante*

sur plusieurs étiquettes comme contexte, modéliser de façon efficace les dépendances entre étiquettes.

Un modèle similaire a été proposé par Dinarelli et Tellier (2016a) et Dinarelli et Tellier (2016c), et ensuite amélioré dans **?**. Nous décrivons ici une version améliorée de ce modèle suivant différentes dimensions : i) nous étudions l'effet de couches cachées plus sophistiquées telles que les *ReLU*, les *Leaky ReLU* et les *ReLU* paramétrés (He *et al.*, 2015) ; ii) nous utilisons la régularisation *dropout* (Srivastava *et al.*, 2014) dans les couches de plongements et les couches cachées ; iii) nous intégrons une couche de convolution sur les caractères pour modéliser des traits à un niveau plus fin que les mots ; iv) nous évaluons les modèles de façon exhaustive en montrant les effets des différents composants et niveaux d'information.

Idéalement, la variante de RNN proposée dans cette article peut être vue comme une évolution du modèle simple de Jordan, dans laquelle la connexion récurrente relie la couche de sortie à la couche d'entrée. Un schéma général des réseaux d'Elman, de Jordan et de la variante proposée est montré dans la figure 1, où w sont les mots, y les étiquettes, et E, H, O et R sont les paramètres du modèle. Tous les RNN mentionnés dans cet article sont étudiés dans leur version *en avant* (dits *forward* dans la suite de l'article), *en arrière* (*backward*) et bidirectionnelle (Schuster et Paliwal, 1997).

Nous évaluons ces modèles sur deux tâches différentes de compréhension de la parole (De Mori *et al.*, 2008) : ATIS (Dahl *et al.*, 1994) et MEDIA (Bonneau-Maynard *et al.*, 2006). ATIS est relativement simple et ne nécessite pas une modélisation poussée des dépendances entre étiquettes. Cette tâche nous permet d'évaluer les modèles dans des conditions similaires à celles des tâches comme l'étiquetage des parties du discours, ou la reconnaissance des entités nommées telle que définie dans la *CoNLL Shared Task* 2003, toutes les deux largement utilisées comme références dans le TAL. MEDIA est une tâche plus difficile pour laquelle la capacité d'un modèle à prendre en compte les dépendances entre étiquettes est cruciale. Les résultats obtenus montrent que notre variante est aussi efficace que les meilleurs RNN actuels sur la tâche ATIS, tout en étant plus simple. Sur MEDIA notre variante obtient des résultats

bien meilleurs que tous les autres RNN, dépassant aussi les CRF, la référence pour l'étiquetage de séquences.

Le reste de l'article est organisé comme suit. Dans la section suivante, nous décrivons en détail les réseaux neuronaux récurrents, partant des modèles existants pour arriver à notre variante. La section 3 est consacrée à la présentation des corpus sur lesquels nous évaluons les modèles, aux paramétrages et choix architecturaux et aux résultats obtenus. La section 4 présente nos conclusions.

2. Réseaux neuronaux récurrents (RNN)

Dans cette section nous présentons l'état de l'art sur les RNN classiques, tels que les RNN d'Elman et de Jordan (Jordan, 1989 ; Elman, 1990), mais aussi certains plus sophistiqués comme les LSTM et les GRU (Hochreiter et Schmidhuber, 1997 ; Cho *et al.*, 2014), ainsi que les procédures d'apprentissage et d'inférence. Par souci de continuité dans l'article, nous présentons ici également notre nouvelle variante de RNN.

2.1. *RNN d'Elman et de Jordan*

Les RNN d'Elman et Jordan sont définis respectivement comme suit :

$$\mathbf{h_t}^{\text{Elman}} = \Phi(R\,\mathbf{h_{t-1}} + H\,\mathbf{I_t}) \quad [1]$$

$$\mathbf{h_t}^{\text{Jordan}} = \Phi(R\,\mathbf{y_{t-1}} + H\,\mathbf{I_t}) \quad [2]$$

La différence entre ces deux modèles tient au calcul des activités de la couche cachée, alors que, dans les deux cas, la couche de sortie est calculée avec :

$$\mathbf{y_t} = softmax(O\,\mathbf{h_t}) \quad [3]$$

où $\mathbf{h_t}$ et $\mathbf{y_t}$ sont respectivement la sortie de la couche cachée (que ce soit dans le modèle d'Elman, Jordan ou autres) et de la couche de sortie, Φ est une fonction d'activation, H, O et R sont respectivement les poids à l'entrée de la couche cachée, de sortie et récurrente (pour simplifier la lecture des formules, nous avons omis les biais). La sortie $\mathbf{y_t}$ est un vecteur qui représente une distribution de probabilités sur l'ensemble des étiquettes. À partir de celle-ci nous pouvons calculer la prédiction du modèle en sélectionnant l'étiquette correspondant à la position avec la probabilité maximale dans $\mathbf{y_t}$. $\mathbf{h_{t-1}}$ est la couche cachée calculée à l'étape de traitement précédente prise en compte comme contexte par le réseau d'Elman, alors que $\mathbf{y_{t-1}}$ est la couche de sortie de l'étape précédente et est utilisé comme contexte à l'étape actuelle par le réseau de Jordan. $\mathbf{I_t}$ est l'entrée du réseau, constituée généralement de la concaténation des plongements des mots dans une fenêtre de taille d_w (de *win**d**ow* ***w**ord*) fixée autour du mot courant w_t. Nous indiquons avec $E_w(w_i)$ le plongement d'un mot quelconque w_i, sélectionné dans la matrice E_w. $\mathbf{I_t}$ est alors défini comme suit :

$$\mathbf{I_t} = [E_w(w_{t-d_w})...E_w(w_t)...E_w(w_{t+d_w})] \quad [4]$$

Les crochets [ ] indiquent la concaténation de vecteurs (ou matrices dans la suite de l'article). La fonction $softmax$, calculée sur un ensemble S, de taille m, de valeurs numériques v_i, associées à des éléments discrets $i \in [1, m]$ (dans notre cas, les index des étiquettes), produit la probabilité associée à chaque élément de la façon suivante :

$$\forall i \in [1, m]\ p(i) = \frac{e^{v_i}}{\sum_{j=1}^{m} e^{v_j}}$$

2.2. *Réseaux* Long Short-Term Memory *(LSTM)*

Bien que le nom LSTM soit utilisé pour indiquer un réseau neuronal à part entière, il définit seulement un type de couche cachée. Dans LSTM, la sortie de la couche cachée est contrôlée par des unités dites *gates* et par une unité de mémoire dite *cell state*. Ces unités déterminent la façon dont l'information passée est prise en compte pour calculer la sortie de la couche cachée à l'étape présente. En particulier, les unités *forget*, *input* et *cell state* sont calculées comme suit :

$$\mathbf{f_t} = \Phi(W_f \mathbf{h_{t-1}} + U_f \mathbf{I_t}) \quad [5]$$

$$\mathbf{i_t} = \Phi(W_i \mathbf{h_{t-1}} + U_i \mathbf{I_t}) \quad [6]$$

$$\hat{\mathbf{c}}_\mathbf{t} = \Gamma(W_c \mathbf{h_{t-1}} + U_c \mathbf{I_t}) \quad [7]$$

Γ est une autre fonction d'activation[1], $\hat{\mathbf{c}}_\mathbf{t}$ est une valeur intermédiaire utilisée pour mettre à jour la valeur de l'unité *cell state* de la façon suivante :

$$\mathbf{c_t} = \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \hat{\mathbf{c}}_\mathbf{t} \quad [8]$$

$\odot$ est la multiplication élément par élément. Une fois que toutes ces quantités ont été calculées, la valeur de l'unité *output* est calculée et utilisée pour produire la sortie de la couche cachée à l'étape présente t :

$$\mathbf{o_t} = \Phi(W_o \mathbf{h_{t-1}} + U_o \mathbf{I_t}) \quad [9]$$

$$\mathbf{h_t}^{\text{LSTM}} = \mathbf{o_t} \odot \Phi(\mathbf{c_t}) \quad [10]$$

Comme nous pouvons le voir, chaque *gate* et la *cell state* ont leurs matrices de paramètres W et U. L'évolution de la couche LSTM nommée GRU (de *Gated Recurrent Units*) (Cho *et al.*, 2014), combine les unités *forget* et *input* en une seule unité, et combine également la couche cachée précédente avec l'unité *cell state* :

$$\mathbf{z_t} = \Phi(W_z \mathbf{h_{t-1}} + U_z \mathbf{I_t}) \quad [11]$$

$$\mathbf{r_t} = \Phi(W_r \mathbf{h_{t-1}} + U_r \mathbf{I_t}) \quad [12]$$

$$\hat{\mathbf{h_t}} = \Gamma(W(\mathbf{r_t} \odot \mathbf{h_{t-1}}) + U\mathbf{I_t}) \quad [13]$$

$$\mathbf{h_t}^{\text{GRU}} = (1 - \mathbf{z_t}) \odot \mathbf{h_{t-1}} + \mathbf{z_t} \odot \hat{\mathbf{h_t}} \quad [14]$$

Les schémas des couches cachées LSTM et GRU sont montrés en figure 2.

1. Dans la littérature, dans les réseaux LSTM et GRU Φ et Γ sont respectivement la sigmoïde et la tangente hyperbolique.

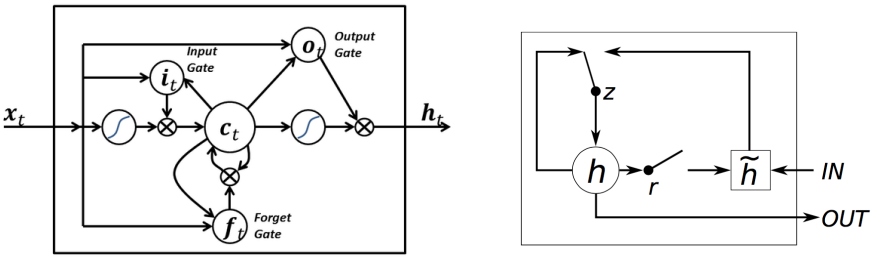


Figure 2. *Schémas des couches cachées LSTM et GRU*

2.3. *LD-RNN : RNN modélisant les dépendances entre étiquettes*

La variante de RNN que nous proposons peut être vue comme un RNN ayant une connexion récurrente entre la couche de sortie et la couche d'entrée. Cette modification permet d'utiliser des plongements pour les étiquettes, de la même façon que pour les mots. En effet, la première couche des réseaux est une table de correspondance qui sert à transformer les représentations creuses *one-hot*[2] en représentations distributionnelles. Ces dernières peuvent encoder des propriétés syntaxiques et sémantiques très attractives, comme cela a été démontré avec *word2vec* (Mikolov *et al.*, 2013a). Ces traits et propriétés peuvent être appris à partir des données étiquetées de la même façon que pour les mots. Dans cet article, on se contentera d'utiliser les séquences d'étiquettes associées aux séquences de mots. Mais cette approche pourrait aussi être étendue à des cas plus sophistiqués, en exploitant des informations structurées quand elles sont disponibles. On pourrait ainsi exploiter de la même façon des arbres syntaxiques ou des entités nommées étendues. L'idée d'utiliser des plongements pour les étiquettes n'est pas nouvelle en soi : Chen et Manning (2014) l'ont introduite dans le contexte de l'analyse syntaxique en dépendances, ce qui a donné un analyseur très performant. Dans ce travail nous allons au-delà de cette approche. En plus de pré-apprendre les plongements des étiquettes, nous utilisons plusieurs étiquettes comme contexte. Cela permet d'apprendre d'une façon très efficace les dépendances entre étiquettes. Pour cette raison, nous appelons cette variante de RNN *LD-RNN*, pour *Label Dependency aware RNN*.

Avec le même formalisme que précédemment, nous notons E_w la matrice des plongements des mots, et E_l celle des plongements des étiquettes. $\mathbf{I_t}$ est la même entrée introduite plus haut, alors que :

$$\mathbf{L_t} = [E_l(y_{t-d_l+1})...E_l(y_{t-d_l+2})...E_l(y_{t-1})] \quad [15]$$

est la concaténation des vecteurs qui représentent les étiquettes prédites aux d_l positions précédentes. Nous notons que y_t indique ici l'étiquette prédite par le modèle,

2. La représentation *one-hot* d'un *token* représenté par son index i dans un dictionnaire est un vecteur v de la même taille que le dictionnaire, et qui vaut zéro partout sauf à la position i où il vaut 1.

et non pas la distribution de probabilités $\mathbf{y_t}$ calculée par le *softmax*. La sortie de la couche cachée est alors calculée comme suit :

$$\mathbf{h_t}^{\text{LD-RNN}} = \Phi(H\ [\mathbf{I_t}\mathbf{L_t}]) \quad [16]$$

Les autres couches sont calculées comme précédemment.

Contrairement à ce qu'indiquaient Dinarelli et Tellier (2016a), la concaténation des deux entrées, mots et étiquettes, à la place de leur somme, ne change pas la capacité du modèle, $H[\mathbf{I_t}\mathbf{L_t}]$ étant équivalent à $H_w\mathbf{I_t} + H_l\mathbf{L_t}$ si $H = [H_w H_l]$.

Les plongements d'étiquettes encodent les caractéristiques du contexte, c'est-à-dire les étiquettes cooccurrentes avec une étiquette donnée. De ce fait, ces plongements encodent l'information permettant de prédire des successions d'étiquettes cohérentes. Les autres RNN n'utilisent pas les étiquettes précédentes dans leur fonction de décision, ou ils utilisent une représentation peu efficace des celles-ci. Ces RNN peuvent donc produire des séquences d'étiquettes incohérentes. Cette limitation a conduit aux modèles hybrides *RNN+CRF* (Huang *et al.*, 2015 ; Lample *et al.*, 2016 ; Ma et Hovy, 2016).

Une conséquence importante de l'utilisation des plongements d'étiquettes est une robustesse accrue du modèle par rapport aux erreurs de prédiction. Puisque nous utilisons plusieurs étiquettes prédites comme contexte (voir $\mathbf{L_t}$ plus haut), en phase de test il est assez improbable que plusieurs erreurs de prédiction subsistent dans le même contexte. Même dans ce cas, grâce aux propriétés des plongements, les étiquettes erronées ont des représentations assez proches de celles des étiquettes attendues. Reprenons un exemple cité dans Mikolov *et al.* (2013b) : si l'on remplace *Paris* par *Rome* dans un texte, cela n'a aucun effet dans plusieurs tâches : ce sont en effet tous les deux des noms propres pour un étiquetage POS, des lieux pour une extraction d'entités nommées, etc. L'utilisation de plongements d'étiquettes permet d'avoir la même robustesse sur les étiquettes.

Il ne se passe en général pas la même chose avec un modèle de Jordan classique. Dans ce modèle, les étiquettes sont représentées soit par la distribution de probabilités sur les étiquettes calculée avec le $softmax$, soit par la représentation *one-hot* calculée à partir de celle-ci. Ces deux représentations sont très creuses[3], elles donnent donc au modèle bien moins de souplesse par rapport à ce que l'on peut obtenir avec des plongements d'étiquettes.

D'un autre point de vue, on peut interpréter le calcul de la couche cachée dans un réseau de Jordan comme sélectionnant des plongements de la matrice R (voir formule 2). Puisque $\mathbf{y_{t-1}}$ est un vecteur creux, nous pouvons interpréter la multiplication de $\mathbf{y_{t-1}}$ par la matrice R comme la sélection d'un plongement pour l'étiquette correspondante. Même avec cette interprétation, il y a une différence importante entre le réseau de Jordan et notre variante. En effet, une fois que la sélection du plongement

3. Nous avons constaté expérimentalement que dans la sortie du *softmax* la masse des probabilités est concentrée dans une ou très peu d'étiquettes seulement (2 ou 3 au maximum).

pour l'étiquette a été faite avec $R\mathbf{y_{t-1}}$, le résultat n'est pas directement utilisé dans la transformation linéaire opérée par la matrice H. Seul le contexte au niveau des mots est multiplié par H ($H\mathbf{I_t}$). Le résultat de cette multiplication est additionné à $R\mathbf{y_{t-1}}$ pour produire l'entrée finale de la couche cachée.

Dans notre variante en revanche, nous sélectionnons d'abord le plongement de chaque étiquette avec $E[y_i]$[4]. Ensuite nous appliquons la transformation linéaire de la couche cachée sur les mots et sur les étiquettes ($H[\mathbf{I_t}\mathbf{L_t}]$). Dans notre variante donc les étiquettes subissent toujours deux transformations : i) la conversion de *one-hot* en plongements, ii) la transformation linéaire de la couche cachée.

2.4. *Apprentissage et inférence*

L'apprentissage de notre variante est réalisée en minimisant l'entropie croisée entre l'étiquette attendue e_t et la sortie du réseau $\mathbf{y_t}$ à la position t dans une séquence, plus un terme de régularisation $L2$:

$$C = -e_t \odot log(\mathbf{y_t}) + \frac{\lambda}{2}|\Theta|^2 \quad [17]$$

où λ est un hyperparamètre du modèle, Θ est un raccourci pour les paramètres d'un modèle. Nous désignons par e_t la représentation creuse *one-hot* de l'étiquette attendue. Puisque $\mathbf{y_t}$ est une distribution de probabilités sur les étiquettes, nous interprétons la sortie du réseau comme la probabilité $P(i|\mathbf{I_t}, \mathbf{L_t})\ \forall i \in [1, m]$, $\mathbf{I_t}$ et $\mathbf{L_t}$ étant les entrées du réseau (mots et étiquettes), i étant l'index d'une des m étiquettes définies dans la tâche. Nous associons au modèle *LD-RNN* la fonction de décision suivante :

$$argmax_{i\in[1,m]}P(i|\mathbf{I_t}, \mathbf{L_t}) \quad [18]$$

Nous pouvons voir que cette fonction de décision est locale, puisque la probabilité des étiquettes est normalisée à chaque position de la séquence traitée. Malgré tout, grâce à l'utilisation des plongements d'étiquettes $\mathbf{L_t}$, la variante *LD-RNN* arrive à modéliser et prendre en compte les dépendances entre étiquettes. Puisque les autres RNN tels que le modèle d'Elman et les LSTM n'utilisent pas les étiquettes comme information contextuelle, leur fonction de décision peut être définie comme :

$$argmax_{i\in[1,m]}P(i|\mathbf{I_t}) \quad [19]$$

Cette fonction de décision peut conduire à des séquences de prédictions incohérentes.

Nous utilisons l'algorithme de rétropropagation classique avec vitesse (*momentum*) et la descente de gradient pour apprendre les poids du modèle (Bengio, 2012), le préférant à l'algorithme de rétropropagation à travers le temps (*Back-propagation through time BPTT*) (Werbos, 1990). En effet, étant donné leur architecture récurrente, pour apprendre proprement un RNN, il faudrait utiliser l'algorithme BPTT. Cet algorithme consiste à *dérouler* (*unfold*) un RNN sur N étapes passées, N étant un paramètre, et utilisant donc les N entrées et contextes précédents pour mettre à jour

4. Dans ce cas, le vecteur $\mathbf{y_i}$ est explicitement converti en une représentation *one-hot*, qui est équivalente à un index scalaire y_i.

tous les paramètres du modèle. L'algorithme de rétropropagation classique est alors appliqué. Cela équivaut donc à apprendre un réseau de profondeur N.

L'algorithme BPTT est censé permettre d'apprendre des contextes arbitrairement longs. Cependant, Mikolov *et al.* (2011) ont montré que les RNN pour les modèles de langues apprennent mieux avec seulement $N = 5$ étapes passées. Ceci pourrait être dû au fait que, dans les tâches de TAL, l'information passée retenue en mémoire par un RNN grâce à sa connexion récurrente se disperse après quelques étapes seulement. Aussi, du moins dans la plupart des tâches de TAL, garder en mémoire un contexte arbitrairement long ne donne pas en général une garantie d'amélioration des performances, un contexte plus long étant aussi plus bruité.

Puisque l'algorithme de rétropropagation à travers le temps est plus coûteux que l'algorithme classique, Mesnil *et al.* (2013) ont préféré utiliser explicitement l'information contextuelle donnée à la connexion récurrente dans les RNN, et utiliser l'algorithme de rétropropagation classique, apparemment sans perte de performances. Dans cet article, nous adoptons la même stratégie d'apprentissage.

Lorsque l'on utilise explicitement les étiquettes prédites aux étapes précédentes dans un réseau de Jordan, la sortie de la couche cachée est calculée par :

$$\mathbf{h_t} = \Phi(R[\mathbf{y_{t-d_l+1}}\ \mathbf{y_{t-d_l+2}} \cdots \mathbf{y_{t-1}}] + H\ \mathbf{I_t}) \quad [20]$$

où d_l est la taille du contexte côté sortie utilisée explicitement dans le réseau. Une modification équivalente permet de prendre en compte les couches cachées précédentes dans un RNN d'Elman, LSTM et GRU.

2.5. *Vers des RNN plus sophistiqués*

2.5.1. *Convolution sur les caractères*

Même si les plongements fournissent une représentation très riche des mots, plusieurs travaux sur les RNN, par exemple (Huang *et al.*, 2015 ; Chiu et Nichols, 2015 ; Lample *et al.*, 2016 ; Ma et Hovy, 2016), ont montré que des modèles plus efficaces peuvent être obtenus en utilisant une couche de convolution sur les caractères de chaque mot donné en entrée au réseau. Cette information est en effet très importante pour permettre au modèle de généraliser les mots dont certaines formes fléchies sont rares ou même hors vocabulaire en phase de test, les plongements sur les mots étant bien moins efficaces dans ces cas. La convolution sur les caractères offre en plus l'avantage d'être assez générique : elle peut être appliquée exactement de la même façon à des langues différentes, permettant la réutilisation d'un même système sur des tâches différentes.

Dans cet article nous utilisons une couche de convolution sur les caractères comme celle utilisée dans (Collobert *et al.*, 2011) pour les mots. Formellement, pour un mot w de longueur $|w|$, en indiquant par $E_{ch}(w, i)$ le plongement du caractère à la position i du mot w, une convolution de taille $2d_c + 1$ est appliquée comme suit :

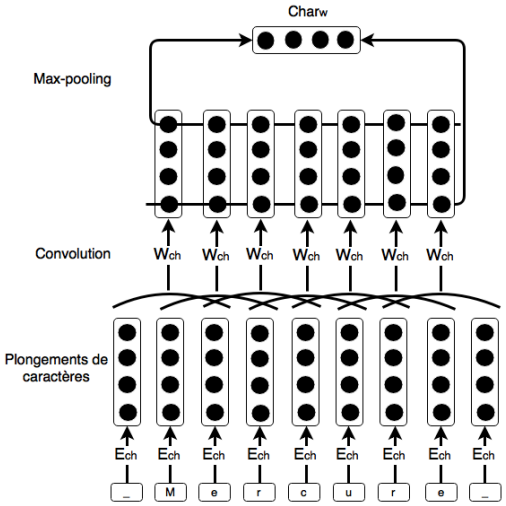


Figure 3. *Schéma de la couche de convolution appliquée sur les caractères du mot* Mercure

$$\forall i \in [1, |w|]\ Conv_i = W_{ch}[E_{ch}(w, i - d_c)...E_{ch}(w, i)...E_{ch}(w, i + d_c)] \quad [21]$$

$$Char_w = Max([Conv_1 \dots Conv_{|w|}]) \quad [22]$$

La sortie de la convolution $[Conv_1 \dots Conv_{|w|}]$ est la concaténation du résultat de l'application du modèle linéaire W_{ch} à chaque fenêtre de taille $2d_c+1$ de plongements de caractères. Pour que la convolution soit bien définie aussi pour les premiers et derniers caractères de chaque mot, celui-ci est augmenté par un préfixe et un suffixe de taille d_c, obtenus par la concaténation d'un caractère conventionnel (nous avons choisi le caractère "_"). Un schéma de la convolution sur les caractères est montré en figure 3.

La taille de la matrice $[Conv_1 \dots Conv_{|w|}]$ est $|C| \times |w|$, $|C|$ étant la taille de la couche de convolution. La fonction Max, utilisée pour calculer le *Max-pooling* (Collobert *et al.*, 2011), calcule les maximums dans la direction de la longueur du mot, produisant une sortie de la taille $|C|$, indépendante donc de la longueur du mot. La sortie finale du *Max-pooling* $Char_w$ peut être interprétée comme une représentation distributionnelle du mot w encodant l'information au niveau des ses caractères. Celle-ci est une information complémentaire par rapport aux plongements des mots, qui encodent des informations intermots. En effet, elle fournit au modèle des informations similaires à celles fournies par des traits discrets tels que les préfixes, les suffixes, l'information de capitalisation, etc.

2.5.2. *Couches cachées* ReLU

Depuis quelques années, les couches cachées des réseaux neuronaux utilisent les unités *ReLU* (Bengio, 2012). Celles-ci ont remplacé les unités utilisant des fonctions d'activation telles que la sigmoïde. Ce type de fonction, appelée *squashing function*[5], a tendance à saturer après un certain nombre d'itérations d'apprentissage, ne permettant pas l'apprentissage de réseaux arbitrairement grands et/ou profonds.

Les unités *ReLU* utilisent une fonction d'activation linéaire rectifiée, définie comme $f(x) = max(0, x)$. Malgré leur efficacité reconnue, les unités *ReLU* ne transmettent pas de signal lorsque l'entrée est négative. Puisque les paramètres des différentes couches des réseaux neuronaux sont repartis autour de zéro, cela implique que les unités *ReLU* n'utilisent pas l'information d'environ la moitié de leur entrée.

Pour pallier cette limitation, des variantes des unités *ReLU* ont été proposées (He *et al.*, 2015). Une première, appelée *Leaky ReLU* (LReLU), est définie comme :

$$LReLU(x) = max(0, x) + \epsilon\, min(0, x)$$

Cette version garde la partie négative de l'entrée, mais celle-ci est multipliée par une valeur négligeable ϵ fixée (He *et al.* (2015) proposent 0,01 ou 0,001 pour ϵ).

Une autre variante proposée par He *et al.* (2015) est une généralisation de *Leaky ReLU* dans laquelle ϵ est remplacé par une matrice de paramètres A. Cette version est appelée *ReLU* paramétré (PReLU) :

$$PReLU(x) = max(0, x) + A \odot min(0, x)$$

Elle a l'avantage d'utiliser un facteur multiplicatif paramétrique pour chaque valeur de l'entrée x, mais ajoute au modèle les paramètres A qui sont appris en même temps que les autres. Grâce à cette version, He *et al.* (2015) obtiennent des résultats meilleurs que l'évaluation humaine sur une tâche de classification d'images.

Nous avons testé toutes ces variantes d'unités *ReLU* dans notre variante *LD-RNN*.

2.6. *Complexité des RNN*

La modélisation des dépendances entre étiquettes, plus efficace dans notre variante *LD-RNN*, est obtenue au prix de l'utilisation de plus de paramètres par rapport aux RNN simples. Cependant le nombre de paramètres est bien moins important que dans les réseaux LSTM et GRU. Dans cette section, nous faisons une analyse de la complexité de chaque type de RNN en termes du nombre de paramètres.

Nous introduisons les symboles suivants : $|H|$ et $|O|$ sont respectivement les tailles des couches cachée et de sortie. La taille de la couche de sortie correspond au nombre d'étiquettes. N est la taille des plongements, la même pour les mots et les étiquettes.

5. En effet cette fonction met en correspondance l'ensemble $\mathbb{R}$ avec l'intervalle réel $[0, 1]$. En théorie cela ne pose aucun problème puisque entre 0 et 1 il y a un nombre infini de réels, mais sur un ordinateur ce n'est pas le cas.

d_w et d_l sont les tailles des contextes déjà introduites précédemment. Nous analysons la couche des plongements et la couche cachée, les autres couches ont le même nombre de paramètres dans tous les réseaux.

Dans la couche cachée des RNN d'Elman et de Jordan nous avons les paramètres suivants :

$$\{|H| * |H|\}_R + \{|H| * (2d_w + 1)N\}_{H^{\text{Elman}}}$$

$$\{|O| * |H|\}_R + \{|H| * (2d_w + 1)N\}_{H^{\text{Jordan}}}$$

Les indices indiquent de quelle matrice proviennent les paramètres. Le facteur $(2d_w + 1)N$ vient de l'utilisation des plongements de $(2d_w + 1)$ mots en entrée, alors que le facteur $|O| * |H|$ dans le réseau de Jordan est dû au fait que R relie la couche de sortie à la couche cachée.

Dans le réseau *LD-RNN* nous avons :

$$\{\{((2d_w + 1)N + d_l N) * |H| + |O| * N\}_{E_l}\}_{H^{\text{LD-RNN}}}$$

Le facteur $|O| * N$ vient de la matrice E_l contenant $|O|$ plongements d'étiquettes de taille N. Dans les tâches utilisées dans cet article pour l'évaluation, nous avons choisi $|H| > |O|$ et $N = |H|$. Par conséquent, l'utilisation des plongements d'étiquettes dans *LD-RNN* n'augmente pas en soi le nombre de paramètres.

En revanche, la matrice H dans *LD-RNN* est dimensionnée pour connecter tous les plongements de mots et d'étiquettes à chaque neurone de la couche cachée. $H^{\text{LD-RNN}}$ contient donc $d_l N$ paramètres de plus que H^{Elman} et H^{Jordan}.

Dans les réseaux LSTM et GRU nous avons 2 matrices additionnelles W et U, pour chaque *gate* et pour la *cell state*, utilisées pour connecter la couche cachée précédente et l'entrée courante. Ces 2 matrices contiennent respectivement $|H| * |H|$ et $(2d_w + 1)N * |H|$ paramètres.

Utilisant la même notation que précédemment, pour LSTM et GRU nous avons :

$$\{4(|H| * |H| + |U| * (2d_w + 1)N)\}_{H^{\text{LSTM}}}$$

$$\{3(|H| * |H| + |U| * (2d_w + 1)N)\}_{H^{\text{GRU}}}$$

Le 3 pour le RNN GRU est dû au fait que dans ce réseau nous avons 2 unités *gate*, plus l'unité *cell state*.

En conclusion, nous pouvons constater que la variante *LD-RNN* utilise plus de paramètres que les RNN simples, mais bien moins que les réseaux plus sophistiqués LSTM et GRU.

2.7. *Réseaux* forward*,* backward *et bidirectionnels*

Nous avons étudié le comportement de notre réseau *LD-RNN* dans trois versions : *forward*, *backward* et bidirectionnel (Schuster et Paliwal, 1997). La version *forward* est celle que nous venons de détailler. La version *backward* est équivalente, à la seule différence près qu'elle traite les séquences de la fin vers le début.

Les RNN bidirectionnels utilisent à la fois l'information contextuelle passée et future pour prédire la prochaine étiquette. Les informations futures ne sont pas disponibles directement, c'est pourquoi un modèle bidirectionnel se sert d'un modèle *backward* comme modèle auxiliaire. Ce modèle est utilisé dans une première passe en arrière (*backward*) pour prédire les informations futures. Ensuite le modèle bidirectionnel fait une passe en avant, utilisant l'information future prédite par le modèle *backward* comme contexte.

Comme cela est décrit en détail par Schuster et Paliwal (1997), il existe deux versions de modèle bidirectionnel. La première utilise un modèle *forward* et un modèle *backward* séparés. La seconde est constituée d'un modèle unique, dans lequel on distingue des couches *forward* et des couches *backward*.

Dans les premières phases de test de nos réseaux, nous avons observé que les deux versions sont équivalentes sur les tâches présentées dans cet article. Nous avons donc choisi d'utiliser la première version. Celle-ci présente l'avantage de pouvoir être initialisée totalement avec les poids des modèles *forward* et *backward* déjà calculés. Ceci a pour conséquence que le modèle bidirectionnel est très proche de l'optimum dès la première itération, et très peu d'itérations suffisent donc pour obtenir un modèle très performant.

Dans cette version de modèle bidirectionnel la sortie est calculée comme la moyenne géométrique des sorties des modèles *forward* et *backward* :

$$\mathbf{y_t} = \sqrt{\mathbf{y_t^f} \odot \mathbf{y_t^b}} \qquad [23]$$

$\mathbf{y_t^f}$ et $\mathbf{y_t^b}$ étant respectivement les sorties du modèle *forward* et *backward*.

3. Évaluation

3.1. *Corpus pour la compréhension automatique de la parole*

Nous avons évalué nos modèles sur deux tâches assez populaires de compréhension automatique de la parole.

Le corpus ATIS (*Air Travel Information System*) (Dahl *et al.*, 1994) a été collecté pour construire des systèmes de dialogues automatiques capables de donner des informations sur les vols aux États-Unis. La représentation sémantique est fondée sur la notion de *slot* d'un *frame* ; le but dans la tâche de compréhension de la parole (SLU) est d'associer chaque mot au *slot* correspondant.

ATIS est une tâche assez simple qui date de 1993. Les données d'apprentissage sont constituées de 4 978 phrases prises parmi celles sans dépendances de contexte dans les corpus `ATIS-2` et `ATIS-3`. Les données de test sont constituées de 893 phrases prises des corpus `ATIS-3 NOV93` et `DEC94`. Il n'y a pas de données de développement, nous avons donc isolé une partie des données d'apprentissage et les avons utilisées comme données de développement. Les dictionnaires de mots et

d'étiquettes sont constitués de 1 117 et 85 unités, respectivement. Nous utilisons la version du corpus publiée dans Raymond et Riccardi (2007), dans laquelle sont aussi disponibles des classes de mots, comme par exemple les noms des villes, des aéroports, expressions de temps etc., permettant aux modèles de généraliser certains mots rares ou absents en phase de test. Nous renvoyons à Dahl *et al.* (1994) pour plus de détails. Dans cet article nous nous comparons aux travaux qui ont utilisé la version officielle d'ATIS (Raymond et Riccardi, 2007).

Un exemple de phrase de ce corpus est *« I want all the flights from Boston to Philadelphia today »*[6]. Les mots *Boston*, *Philadelphia* et *today* sont associés aux *slots* `DEPARTURE.CITY`, `ARRIVAL.CITY` et `DEPARTURE.DATE`, respectivement. Tous les autres mots de la phrase n'appartiennent à aucun *slot*, ils sont donc annotés avec l'étiquette "vide" *O*. Ces caractéristiques de l'annotation montrent la simplicité de cette tâche : l'annotation est assez creuse, seulement 3 mots de la phrase sont annotés avec des concepts non vides. Chaque étiquette correspond à un seul mot, il n'y a aucune segmentation des étiquettes sur plusieurs mots (comme c'est le cas avec un codage *BIO*). À cause de ces caractéristiques, la tâche ATIS est similaire à une tâche d'étiquetage en partie du discours (*POS tagging*), où il n'y a pas non plus de segmentation des étiquettes sur les mots. Cette tâche peut aussi être comparée à de la reconnaissance d'entités nommées linéaires, où l'annotation est également creuse.

Le corpus français MEDIA (Bonneau-Maynard *et al.*, 2006) a été créé pour l'évaluation de systèmes de dialogues automatiques destinés à fournir des informations touristiques sur les hôtels en France. Il est constitué de 1 250 dialogues acquis avec le protocole *Wizard-of-OZ* où 250 orateurs ont suivi 5 scénarios de réservation différents. Les dialogues ont été transcrits et annotés manuellement suivant une ontologie de concepts riche. Des composants sémantiques peuvent être combinés pour former des étiquettes sémantiques complexes.[7] Outre la richesse de l'annotation utilisée, une autre source de difficultés pour l'étiquetage provient des phénomènes de coréférence. Certains mots ne peuvent pas être annotés correctement sans connaître un contexte relativement long, comprenant souvent le tour de parole précédent. Dans la phrase *« Oui, celui à moins de 50 euros par nuit »*, *celui* réfère à un hôtel introduit dans un tour de parole précédent. Les propriétés statistiques des données d'apprentissage, de développement et de test du corpus MEDIA sont données dans le tableau 2.

La tâche MEDIA peut être modélisée comme un étiquetage de séquences en utilisant la convention classique *BIO* (Ramshaw et Marcus, 1995).

Grâce à ces différentes caractéristiques, combinées avec une taille relativement réduite qui permet l'apprentissage des modèles en un temps raisonnable, ces deux tâches constituent un terrain idéal pour l'évaluation de modèles pour l'étiquetage de séquences. Un exemple comparatif de ces deux tâches est montré dans le tableau 1.

6. Je voudrais tous les vols d'aujourd'hui de Boston à Philadelphie.

7. Par exemple l'étiquette `localisation` peut être combinée avec les composants `ville`, `distance-relative`, `localisation-relative-générale`, `rue`, etc.

MEDIA			ATIS		
Mots	**Classes**	**Étiquettes**	**Mots**	**Classes**	**Étiquettes**
Oui	-	Answer-B	i'd	-	O
l'	-	BDObject-B	like	-	O
hotel	-	BDObject-I	to	-	O
le	-	Object-B	fly	-	O
prix	-	Object-I	Delta	airline	airline-name
à	-	Comp.-payment-B	between	-	O
moins	relative	Comp.-payment-I	Boston	city	fromloc.city-name
cinquante	tens	Paym.-amount-B	and	-	O
cinq	units	Paym.-amount-I	Chicago	city	toloc.city-name
euros	currency	Paym.-currency-B			

Tableau 1. *Un exemple de phrase annotée pris des deux corpus MEDIA et ATIS*

	Apprentissage		Développement		Test	
# phrases	12,908		1,259		3,005	
	Mots	Concepts	Mots	Concepts	Mots	Concepts
# mots	94,466	43,078	10,849	4,705	25,606	11,383
# vocab.	2,210	99	838	66	1,276	78
# OOV%	–	–	1.33	0.02	1.39	0.04

Tableau 2. *Statistiques sur le corpus MEDIA*

3.2. *Choix architecturaux et réglages*

Notre variante de RNN *LD-RNN* a été implémentée par nous-mêmes en *Octave*[8] avec le support de la bibliothèque *OpenBLAS*.[9]

Les modèles *LD-RNN* sont entraînés suivant la procédure suivante :

– deux modèles de langage neuronaux comme ceux de Bengio *et al.* (2003) sont entraînés pour générer les plongements des mots et des étiquettes ;

– les modèles *forward* et *backward* sont appris en utilisant les plongements entraînés à l'étape précédente ;

– le modèle bidirectionnel est appris en utilisant comme poids initiaux les modèles *forward* et *backward* entraînés à l'étape précédente.

Les plongements des mots et des étiquettes sont appris sur les données d'apprentissage de la tâche concernée. Nos systèmes sont donc entièrement endogènes. Nous avons également fait des expériences avec des plongements appris avec *word2vec* (Mikolov *et al.*, 2013a). Les résultats obtenus ne sont pas significativement différents des autres, ils ne seront donc pas discutés dans la suite de l'article.

8. https ://www.gnu.org/software/octave/ ; Notre logiciel est décrit à la page http ://www.marcodinarelli.it/software.php et il est disponible sous requête.
9. http ://www.openblas.net ; Cette bibliothèque permet une accélération d'un facteur d'environ 330 par rapport à une bibliothèque BLAS basique, en utilisant 16 cœurs.

Nous avons grossièrement optimisé le nombre d'époques d'entraînement pour chaque modèle sur les données de développement : 30 époques pour entraîner les plongements des mots, 20 pour les plongements d'étiquettes ; 30 époques aussi pour les modèles *forward* et *backward*, 8 pour le modèle bidirectionnel (l'optimum de ce dernier est souvent atteint à la première époque sur ATIS, entre la 3^e et la 5^e sur MEDIA). À la fin de l'entraînement, nous gardons le modèle qui donne la meilleure précision sur les données de développement, nous arrêtons l'apprentissage si elle n'est pas améliorée pendant 5 époques consécutives (*Early stopping* (Bengio, 2012)).

Nous initialisons les poids des modèles avec la procédure dite *Xavier initialization* (Bengio, 2012). Celle-ci est motivée théoriquement dans (He *et al.*, 2015) comme permettant de préserver pendant l'apprentissage la variance de la distribution dans laquelle les valeurs initiales des poids sont prises.

Nous avons optimisé certains hyperparamètres de nos modèles sur les données de développement : sur la base de cette optimisation, le taux d'apprentissage est initialisé à 0,5 et diminué d'une valeur fixe à chaque époque (*Learing Rate decay*). Cette valeur est le rapport entre le taux d'apprentissage initial et le nombre d'itérations. Pour la régularisation, nous combinons le *dropout* et la régularisation L_2 (Bengio, 2012), la meilleure probabilité de *dropout* sur la couche cachée est de 0,5, alors qu'elle est de 0,2 sur les plongements pour ATIS, 0,15 pour MEDIA. Le meilleur coefficient pour la régularisation L_2 est de 0,01 dans les modèles *forward* et *backward* et il est de $3e^{-4}$ pour le modèle bidirectionnel.

Nous avons aussi mené des expériences pour choisir une taille optimale pour les couches des plongements, les couches cachées et la couche de convolution. Pour rendre la phase d'optimisation la plus rapide possible, nous avons optimisé la taille de ces couches sur les données de développement des deux corpus, avec le seul modèle *forward*, intégrant seulement les mots et les étiquettes comme informations d'entrée (donc sans classes ni convolution sur les caractères). La meilleure taille trouvée pour les plongements et la couche cachée a été de 200 pour les deux tâches. La meilleure taille pour la couche de convolution est de 50 pour la tâche ATIS, et de 80 pour MEDIA. Dans les deux cas, la meilleure taille pour la fenêtre de convolution est de 1. Une fenêtre de taille 3 (un caractère à gauche et un à droite, plus le caractère central) donne à peu près les mêmes résultats, nous préférons donc le modèle le plus simple. Avec une fenêtre de taille 5 les résultats commencent à se dégrader. Nous avons également optimisé la taille du contexte de mots et d'étiquettes dans notre variante de RNN *LD-RNN*. Sur ATIS, les meilleures tailles pour ces contextes sont de 11 pour les mots (5 à gauche, 5 à droite et le mot courant) et de 5 pour les étiquettes. Sur MEDIA elles sont de 7 et 5 respectivement. Dans le cas des étiquettes, le contexte est pris d'un seul côté par rapport au mot courant. Seul le modèle bidirectionnel utilise un contexte de deux côtés. Par conséquent notre modèle bidirectionnel utilise un contexte de 10 étiquettes : 5 avant et 5 après le mot courant.

Tous les paramètres fixés dans cette phase d'optimisation ont été utilisés pour obtenir des modèles *baseline*. Le but est de comprendre le comportement des différents réseaux en ajoutant les autres informations d'entrée : les classes disponibles dans les

Modèle	Mesure F1		
	Forward	*Backward*	Bidirectionnel
LD-RNN Mots	94,23	94,30	94,45
LD-RNN Mots + CC	94,56	94,69	94,79
LD-RNN Mots + Classes	95,31	95,42	95,53
LD-RNN Mots + Classes + CC	**95,55**	**95,45**	**95,65**
$LD\text{-}RNN_{LReLU}$ Mots + Classes + CC	95,29	**95,45**	95,52
$LD\text{-}RNN_{PReLU}$ Mots + Classes + CC	94,67	95,08	95,13

Tableau 3. *Résultats (F1) sur le corpus ATIS selon différents niveaux d'information*

deux tâches et la convolution sur les caractères. Comme cela sera discuté plus loin, certains paramètres ont été ensuite raffinés.

En ce qui concerne le temps d'apprentissage de nos modèles, le temps total pour apprendre et tester les trois modèles *forward*, *backward* et bidirectionnel, utilisant seulement les mots et les étiquettes comme entrées, est d'environ 1 h 10 pour la tâche MEDIA, 40 minutes pour ATIS. Ce temps s'élève à 2 h 00 pour MEDIA, à 2 h 10 pour ATIS, en utilisant aussi les classes et la convolution sur les caractères. Tous ces temps sont mesurés sur un processeur *Intel Xeon E5-2620* à 2.1 *GHz* en utilisant 16 cœurs.

3.3. *Résultats*

Tous les résultats discutés dans cette section sont obtenus en moyennant 6 expériences différentes, dans lesquelles les plongements sont entraînés une fois pour toutes, alors que les poids des autres couches sont réinitialisés à chaque fois avec une graine différente.

3.3.1. *Résultats principaux avec différents niveaux d'information*

Nous commentons ici les résultats obtenus en ajoutant progressivement les différentes informations disponibles pour l'entrée de nos modèles : les mots seuls, les mots plus la convolution sur les caractères, les mots plus les classes, les mots plus les classes et la convolution. Ces différents niveaux d'information sont indiqués par *Mots*, *Classes* et *CC* (pour convolution de caractères). Nos modèles utilisent tous un contexte d'étiquettes de taille 5, la taille optimale déjà commentée dans la section 3.2.

Les résultats obtenus pour la tâche ATIS sont donnés dans le tableau 3, et ceux sur MEDIA dans le tableau 4. Nous y montrons également les résultats obtenus avec les différentes couches cachées *ReLU* décrites dans la section 2.5.2. Par défaut nous utilisons la couche *ReLU* classique, l'utilisation des couches *Leaky ReLU* et *ReLU* paramétré est indiquée dans le nom du modèle avec *LReLU* et *PReLU*, respectivement.

Comme nous pouvons le voir dans les tableaux, le comportement des modèles sur les deux tâches est assez similaire. Sur la tâche ATIS notamment, en ajoutant les différents niveaux d'information les résultats s'améliorent progressivement. Les meilleurs résultats sont obtenus avec tous les niveaux d'information (mots, classes et convo-

Modèle	Mesure F1		
	Forward	*Backward*	Bidirectionnel
LD-RNN Mots	85,39	86,54	87,05
LD-RNN Mots + CC	85,41	86,48	86,98
LD-RNN Mots + Classes	**85,46**	86,59	87,16
LD-RNN Mots + Classes + CC	85,38	**86,79**	**87,22**
$LD\text{-}RNN_{LReLU}$ Mots + Classes + CC	**85,48**	86,72	87,18
$LD\text{-}RNN_{PReLU}$ Mots + Classes + CC	84,91	86,36	86,79

Tableau 4. *Résultats (F1) sur le corpus MEDIA selon différents niveaux d'information*

lution des caractères) et ce, malgré que certains gains ne semblent pas significatifs compte tenu de la taille assez réduite de cette tâche.

Cela est plus évident sur la tâche MEDIA, où l'ajout de la convolution des caractères aux mots provoque des légères pertes de performances. Pour les deux tâches, nous avons analysé le comportement du modèle lors de l'apprentissage et nous avons constaté que le modèle était saturé. C'est-à-dire que la taille de la couche cachée, optimisée avec le modèle *forward* et en n'utilisant que les mots, semblait ne pouvoir plus modéliser de façon efficace l'information supplémentaire donnée en entrée au modèle. Nous avons alors lancé d'autres expériences avec une couche cachée de taille 256, qui ont donné les résultats montrés dans les tableaux avec le modèle *LD-RNN Mots + Classes + CC*. Par manque de temps nous n'avons pas optimisé davantage les paramètres de ces réseaux.

Nous notons que l'utilisation des couches cachées *LReLU* et *PReLU* n'améliore pas les résultats obtenus avec la couche *ReLU* classique, voire les dégrade dans le cas de la couche *PReLU*. Ces résultats relativement décevants ont une explication : en analysant la phase d'apprentissage nous avons remarqué que les valeurs de la fonction de coût (l'entropie croisée) sur les données d'apprentissage sont meilleures qu'avec la couche *ReLU* classique. Il s'agit donc de surapprentissage. Avec la couche *PReLU*, notamment, en utilisant les mêmes paramètres, le réseau est instable (le fameux problème du *exploding gradient* (Hochreiter *et al.*, 2001)). Nous avons résolu le problème en baissant le taux d'apprentissage de 0,5 à 0,1.

Malgré un taux d'apprentissage bien moins important, ces couches cachées conduisent à des valeurs de la fonction de coût sur les données d'apprentissage bien meilleures qu'avec une couche cachée *ReLU* classique. Cela montre un grand potentiel pour apprendre les RNN, mais aussi le besoin d'une optimisation plus fine de la régularisation. Par manque de temps nous laissons cela pour des travaux futurs.

Au-delà de tout cela, les résultats discutés ici sont très compétitifs par rapport aux meilleurs résultats de la littérature, comme nous le montrons dans la section suivante.

3.3.2. *Comparaison avec d'autres travaux*

Nous comparons ici les résultats obtenus par nos modèles avec ceux publiés dans la littérature. Pour que la comparaison soit équitable, nous comparons nos modèles *LD-RNN* utilisant la même information en entrée : les mots et les classes de mots.

Modèle	Mesure F1		
	Forward	*Backward*	Bidirectionnel
(Vukotic *et al.*, 2016) LSTM	95,12	–	95,23
(Vukotic *et al.*, 2016) GRU	**95,43**	–	**95,53**
(Dinarelli et Tellier, 2016a) E-RNN	94,73	93,61	94,71
(Dinarelli et Tellier, 2016a) J-RNN	94,94	94,80	94,89
(Dinarelli et Tellier, 2016a) I-RNN	95,21	94,64	94,75
LD-RNN Mots + Classes	95,31	95,42	**95,53**

Tableau 5. *Comparaison des résultats sur le corpus ATIS en termes de mesure* F1

Modèle	Mesure F1		
	Forward	*Backward*	Bidirectionnel
(Vukotic *et al.*, 2015) CRF		86,00	
(Vukotic *et al.*, 2015) E-RNN	81,94	–	–
(Vukotic *et al.*, 2015) J-RNN	83,25	–	–
(Vukotic *et al.*, 2016) LSTM	81,54	–	83,07
(Vukotic *et al.*, 2016) GRU	83,18	–	83,63
(Dinarelli et Tellier, 2016a) E-RNN	82,64	82,61	83,13
(Dinarelli et Tellier, 2016a) J-RNN	83,06	83,74	84,29
(Dinarelli et Tellier, 2016a) I-RNN	84,91	86,28	86,71
LD-RNN Mots + Classes	**85,46**	**86,59**	**87,16**

Tableau 6. *Comparaison des résultats sur le corpus MEDIA en termes de mesure* F1

Les résultats pour la tâche ATIS sont montrés dans le tableau 5. Ils sont comparés aux résultats de Vukotic *et al.* (2016) et Dinarelli et Tellier (2016a), ces derniers ayant été obtenus avec un modèle similaire à notre modèle *LD-RNN*.

Comme on peut le voir dans le tableau 5, tous les modèles obtiennent des très bons résultats sur cette tâche, au-delà de 94,5 en $F1$, confirmant ce que nous avions anticipé dans la section 3.1 concernant la simplicité de cette tâche. Les modèles GRU et notre variante *LD-RNN* obtiennent des résultats équivalents (95,53) et légèrement meilleurs par rapport aux autres, notamment avec leur version bidirectionnelle. Ceci est en soi un bon résultat, notre variante *LD-RNN* étant bien plus simple qu'un modèle GRU en termes de nombre de paramètres du modèle (voir la section 2.6). Les modèles comme LSTM et GRU sont réputés comme très efficaces pour modéliser le contexte côté source (les mots), ce qui constitue l'information la plus importante dans cette tâche : la meilleure taille du contexte de mots est de 11 (5 mots à gauche plus 5 à droite et le mot courant à étiqueter) dans nos modèles *LD-RNN*.

Si l'on compare ces résultats à ceux de Dinarelli et Tellier (2016a) avec un modèle de Jordan, qui utilise un contexte d'étiquettes de la même taille que nos modèles, nous pouvons conclure que l'avantage de notre variante *LD-RNN* vient de l'utilisation des plongements d'étiquettes et leur combinaison au niveau de la couche cachée.

Cette conclusion devient évidente quand on compare les résultats d'un RNN employant les plongements d'étiquettes aux autres RNN sur la tâche MEDIA. La comparaison des résultats avec la littérature pour cette tâche est donnée dans le tableau 6. Comme nous l'avons expliqué dans la section 3.1, cette tâche est bien plus difficile que

Modèle	CER
(Dinarelli *et al.*, 2011)	11,7
(Dinarelli et Rosset, 2011)	11,5
(Hahn *et al.*, 2010)	10,6
LD-RNN Mots	10,73 (10,63)
LD-RNN Mots + Classes	10.52 (10,15)
LD-RNN Mots + Classes + CC	**10,41 (10,09)**
$LD\text{-}RNN_{LReLU}$ Mots + Classes + CC	**10,38** (10,16)

Tableau 7. *Résultats sur le corpus MEDIA comparés à d'autres travaux en termes de* Concept Error Rate *(CER) sur l'extraction des concepts seuls (sans les valeurs). Les valeurs entre parenthèses sont les meilleurs résultats des* 6 *moyennés pour chaque modèle.*

la tâche ATIS, et ce pour plusieurs raisons, mais dans le contexte de cet article nous nous intéressons plus particulièrement aux dépendances entre étiquettes que nous prétendons modéliser de façon plus efficace avec les plongements d'étiquettes.

Dans ce contexte, nous notons qu'un modèle de Jordan classique, le modèle *J-RNN* de Dinarelli et Tellier (2016a), seul modèle traditionnel employant un contexte d'étiquettes, est plus efficace que les autres modèles classiques, y compris LSTM et GRU (F1 de 84,29 avec J-RNN, contre 83,63 pour le modèle GRU, deuxième meilleur modèle dans les RNN plus classiques). Nous notons également que, sur cette tâche, un CRF, modèle conçu pour l'étiquetage de séquences, est bien plus efficace que tous les RNN classiques (F1 86,00 avec le CRF de Vukotic *et al.* (2015)). Les seuls modèles capables d'obtenir des résultats meilleurs qu'un CRF sur cette tâche sont l'*I-RNN* de Dinarelli et Tellier (2016a) et notre variante *LD-RNN*, tous les deux employant les plongements d'étiquettes. Grâce à l'utilisation d'une couche cachée de type *ReLU* et d'une régularisation *dropout* sur la couche cachée et sur la couche des plongements, notre *LD-RNN* est le modèle le plus efficace aussi sur la tâche MEDIA.

Même si les résultats sur la tâche MEDIA commentés jusqu'ici sont très compétitifs, celle-ci est à la base conçue pour la reconnaissance automatique de la parole (De Mori *et al.*, 2008). Dans cette tâche, le but est d'extraire correctement les concepts que le système de dialogue va utiliser pour interpréter la requête de l'utilisateur. Bien que la mesure F1 y soit fortement corrélée, la mesure d'évaluation classique pour cette tâche est le *Concept Error Rate* (CER), défini de la même façon que le *Word Error Rate* pour la reconnaissance vocale, où les mots sont remplacés par les concepts.

Pour placer nos modèles sur une échelle absolue dans la tâche MEDIA, nous proposons une comparaison de nos meilleurs modèles avec ceux de la littérature en termes de CER, à savoir (Hahn *et al.*, 2010), (Dinarelli et Rosset, 2011) et (Dinarelli *et al.*, 2011). Cette comparaison est montrée dans le tableau 7. Les meilleurs modèles individuels publiés dans (Hahn *et al.*, 2010 ; Dinarelli et Rosset, 2011 ; Dinarelli *et al.*, 2011) sont des CRF, dont le CER est de 10,6, 10,5 et 11,7, respectivement. Nous notons que ces modèles utilisent les mots et les classes comme entrées, mais aussi un certains nombre de traits classiques pour augmenter le pouvoir de généralisation du modèle sur les mots rares et/ou hors vocabulaire, tels que les préfixes et suffixes des mots, des traits binaires sur la capitalisation des mots, etc. Tous ces traits fournissent

au modèle le même type d'information qui peut être obtenu avec la convolution sur les caractères utilisée par nos modèles. Nous notons également que l'écart si important entre ces modèles CRF est dû au fait que le CRF de Hahn *et al.* (2010) est entraîné avec un critère d'apprentissage amélioré par rapport à un CRF classique, et fondé sur une notion de marge similaire à celle des *SVM* (Herbrich *et al.*, 2000 ; Hahn *et al.*, 2009). Par ailleurs, nous notons aussi que d'après les tests de significativité publiés dans (Dinarelli *et al.*, 2011), une différence de 0,1 dans le CER est déjà significative sur MEDIA.

Notre meilleur modèle *LD-RNN* obtient un CER de 10,41. À notre connaissance, il s'agit du meilleur résultat obtenu sur cette tâche avec un modèle individuel. En effet les auteurs de (Hahn *et al.*, 2010) obtiennent un CER de 10,2 avec un modèle *ROVER* (Fiscus, 1997) combinant 6 modèles individuels. Comme cela a déjà été fait dans d'autres travaux, par exemple (Yao *et al.*, 2013), plutôt que moyenner 6 résultats obtenus avec autant de modèles, il est possible de lancer toujours 6 expériences différentes et de ne garder que le meilleur modèle des 6 sur les données de développement d'une tâche donnée.[10] Les résultats obtenus avec nos modèles *LD-RNN* en suivant cette procédure sont donnés dans le tableau 7 entre parenthèses. Notre meilleur résultat est un CER de 10,09, constituant le meilleur résultat absolu jusqu'ici sur la tâche MEDIA.

3.4. *Analyses*

Pour montrer que les meilleurs résultats de notre variante de RNN sur MEDIA sont dus à une meilleure modélisation des séquences, nous avons fait des analyses des résultats en les comparant avec ceux obtenus par des réseaux d'Elman et de Jordan entraînés avec le code mis à disposition par Mesnil *et al.* (2013).[11]

La plus grosse différence dans les résultats est que le modèle d'Elman commence l'annotation d'un concept avec l'étiquette *I* (schéma d'annotation *BIO*). Sans être une erreur grave en soi, c'est une conséquence claire d'une décision locale. Le modèle de Jordan souffre moins de ce problème puisqu'il intègre les étiquettes dans son contexte. Les plongements d'étiquettes utilisés par la variante *LD-RNN* rendent ce modèle bien plus robuste, ce dernier ne produisant aucune erreur de segmentation (*BIO*). Ceci est d'autant plus remarquable que nous n'avons jamais observé une telle précision même sur des modèles CRF optimisés pour la tâche MEDIA.

Le tableau 8 montre les 5 erreurs d'insertion les plus fréquentes d'un RNN de Jordan et de *LD-RNN*. Le modèle de Jordan fait plus d'erreurs sur des concepts difficiles à annoter pour un modèle qui n'arrive pas à prendre en compte les dépendances entre étiquettes. Un exemple frappant est le concept *command-tache*, utilisé pour désigner de façon générale la requête de l'utilisateur. Ce concept est instancié par des séquences de mots relativement longues. Le modèle de Jordan a tendance à le segmenter en plu-

10. Éventuellement en sauvegardant la graine utilisé pour initialiser les poids du modèle de façon à ce que l'entraînement puisse être reproduit.
11. http ://deeplearning.net/tutorial/rnnslu.html

Jordan RNN	LD-RNN
1 : 46 -> connectprop	1 : 39 -> lienref-coref
2 : 44 -> lienref-coref	2 : 36 -> connectprop
3 : 28 -> command-tache	3 : 25 -> nombre
4 : 26 -> nombre	4 : 11 -> reponse
5 : 13 -> reponse	5 : 11 -> loc.-distancerelative

Tableau 8. *Les* 5 *erreurs les plus fréquentes sur MEDIA avec un RNN de Jordan et LD-RNN*

	Jordan RNN			LD-RNN		
label	P	R	F1	P	R	F1
connectprop	74,03	76,05	75,03	82,06	70,98	76,12
lienRef-coRef	82,56	84,87	83,70	88,89	87,91	88,40
command-tache	73,28	77,31	75,24	82,89	80,47	81,67

Tableau 9. *Détails sur les concepts du corpus MEDIA les plus difficiles à annoter correctement*

sieurs concepts, en introduisant une étiquette vide (*O*). Ce même comportement est observé avec un modèle d'Elman. Puisqu'il ne prend pas en compte les étiquettes dans son contexte, nous pensons qu'il sera observé aussi dans un LSTM. En revanche, ce concept ne fait pas partie des 5 erreurs les plus fréquentes du *LD-RNN*.

Les concepts *connectprop* et *lienref-coref* sont utilisés pour les coréférences. Ceux-ci sont de loin les concepts les plus difficiles à annoter dans la tâche MEDIA, leur annotation dépendant fortement du contexte. Cela est visible dans le tableau 8, où ils sont les concepts les plus sujets à erreur. Sur ces concepts aussi, le modèle *LD-RNN* obtient des résultats bien meilleurs, comme nous le montrons dans le tableau 9.

4. Conclusion

Dans cet article nous avons proposé une variante de RNN employant des plongements d'étiquettes et utilisant un large contexte d'étiquettes comme information additionnelle pour prédire la prochaine étiquette dans une séquence. Nous avons motivé ce type d'architecture comme étant plus efficace que les autres RNN pour modéliser des dépendances entre étiquettes. Les résultats sur deux tâches de compréhension automatique de la parole montrent que i) sur une tâche relativement simple comme ATIS, notre variante obtient les mêmes résultats que des réseaux plus complexes et réputés plus efficaces comme LSTM et GRU ; ii) sur la tâche MEDIA, dans laquelle la modélisation des dépendances entre étiquettes est cruciale, notre variante obtient des résultats largement meilleurs que tous les autres RNN. Comparée aux meilleurs modèles publiés dans la littérature en termes de *Concept Error Rate*, notre variante s'avère plus performante, obtenant un CER à l'état de l'art de 10,09.

Remerciements

Ce travail a été financé en partie par le projet ANR Democrat ANR-15-CE38-0008.

5. Bibliographie

Bengio Y., « Practical recommendations for gradient-based training of deep architectures », *CoRR*, 2012.

Bengio Y., Ducharme R., Vincent P., Jauvin C., « A Neural Probabilistic Language Model », *JOURNAL OF MACHINE LEARNING RESEARCH*, vol. 3, p. 1137-1155, 2003.

Bengio Y., Simard P., Frasconi P., « Learning Long-term Dependencies with Gradient Descent is Difficult », *Trans. Neur. Netw.*, vol. 5, n° 2, p. 157-166, March, 1994.

Bonneau-Maynard H., Ayache C., Bechet F., Denis A., Kuhn A., Lefèvre F., Mostefa D., Qugnard M., Rosset S., Servan S. Vilaneau J., « Results of the French Evalda-Media evaluation campaign for literal understanding », *LREC*, Genoa, Italy, p. 2054-2059, May, 2006.

Chen D., Manning C., « A Fast and Accurate Dependency Parser using Neural Networks », *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, p. 740-750, October, 2014.

Chiu J. P. C., Nichols E., « Named Entity Recognition with Bidirectional LSTM-CNNs », *CoRR*, 2015.

Cho K., van Merrienboer B., Gülçehre Ç., Bougares F., Schwenk H., Bengio Y., « Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation », *CoRR*, 2014.

Collobert R., Weston J., « A Unified Architecture for Natural Language Processing : Deep Neural Networks with Multitask Learning », *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, ACM, New York, NY, USA, p. 160-167, 2008.

Collobert R., Weston J., Bottou L., Karlen M., Kavukcuoglu K., Kuksa P., « Natural Language Processing (Almost) from Scratch », *J. Mach. Learn. Res.*, vol. 12, p. 2493-2537, November, 2011.

Dahl D. A., Bates M., Brown M., Fisher W., Hunicke-Smith K., Pallett D., Pao C., Rudnicky A., Shriberg E., « Expanding the Scope of the ATIS Task : The ATIS-3 Corpus », *Proceedings of the Workshop on Human Language Technology*, HLT '94, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 43-48, 1994.

De Mori R., Bechet F., Hakkani-Tur D., McTear M., Riccardi G., Tur G., « Spoken Language Understanding : A Survey », *IEEE Signal Processing Magazine*, vol. 25, p. 50-58, 2008.

Dinarelli M., Moschitti A., Riccardi G., « Discriminative Reranking for Spoken Language Understanding », *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 20, p. 526-539, 2011.

Dinarelli M., Rosset S., « Hypotheses Selection Criteria in a Reranking Framework for Spoken Language Understanding », *Conference of Empirical Methods for Natural Language Processing*, Edinburgh, U.K., p. 1104-1115, Jul, 2011.

Dinarelli M., Tellier I., « Etude des reseaux de neurones recurrents pour etiquetage de sequences », *Actes de la 23eme conférence sur le Traitement Automatique des Langues Na-*

turelles, Association pour le Traitement Automatique des Langues, Paris, France, Juillet, 2016a.

Dinarelli M., Tellier I., « Improving Recurrent Neural Networks For Sequence Labelling », *CoRR*, 2016b.

Dinarelli M., Tellier I., « New Recurrent Neural Network Variants for Sequence Labeling », *Proceedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics*, Lecture Notes in Computer Science (Springer), Konya, Turkey, Avril, 2016c.

Dupont Y., Dinarelli M., Tellier I., « Label-Dependencies Aware Recurrent Neural Networks », *Proceedings of the 18th International Conference on Computational Linguistics and Intelligent Text Processing*, Lecture Notes in Computer Science (Springer), Budapest, Hungary, April, 2017.

Elman J. L., « Finding structure in time », *COGNITIVE SCIENCE*, vol. 14, n° 2, p. 179-211, 1990.

Fiscus J. G., « A Post-Processing System to Yield Reduced Word Error Rates : Recogniser Output Voting Error Reduction (ROVER) », *Proceedings 1997 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Santa Barbara, CA, p. 347-352, December, 1997.

Hahn S., Dinarelli M., Raymond C., Lefèvre F., Lehen P., De Mori R., Moschitti A., Ney H., Riccardi G., « Comparing Stochastic Approaches to Spoken Language Understanding in Multiple Languages », *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, 2010.

Hahn S., Lehnen P., Heigold G., Ney H., « Optimizing CRFs For SLU Tasks In Various Languages Using Modified Training Criteria », *Proceedings of the International Conference of the Speech Communication Assosiation (Interspeech)*, Brighton, U.K., 2009.

He K., Zhang X., Ren S., Sun J., « Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification », *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, p. 1026-1034, 2015.

Herbrich R., Graepel T., Obermayer K., *Large Margin Rank Boundaries for Ordinal Regression*, MIT Press, March, 2000.

Hochreiter S., Bengio Y., Frasconi P., Schmidhuber J., « Gradient flow in recurrent nets : the difficulty of learning long-term », *in* Kremer, Kolen (eds), *A Field Guide to Dynamical Recurrent Neural Networks*, IEEE Press, 2001.

Hochreiter S., Schmidhuber J., « Long Short-Term Memory », *Neural Comput.*, vol. 9, n° 8, p. 1735-1780, November, 1997.

Huang Z., Xu W., Yu K., « Bidirectional LSTM-CRF models for sequence tagging », *arXiv preprint arXiv :1508.01991*, 2015.

Jordan M. I., « Serial Order : A Parallel, Distributed Processing Approach », *in* J. L. Elman, D. E. Rumelhart (eds), *Advances in Connectionist Theory : Speech*, Erlbaum, Hillsdale, NJ, 1989.

Lample G., Ballesteros M., Subramanian S., Kawakami K., Dyer C., « Neural architectures for named entity recognition », *arXiv preprint arXiv :1603.01360*, 2016.

Ma X., Hovy E., « End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF », *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, 2016.

Mesnil G., He X., Deng L., Bengio Y., « Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding », *Interspeech 2013*, August, 2013.

Mikolov T., Chen K., Corrado G., Dean J., « Efficient Estimation of Word Representations in Vector Space », *CoRR*, 2013a.

Mikolov T., Karafiát M., Burget L., Cernocký J., Khudanpur S., « Recurrent neural network based language model », *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, p. 1045-1048, 2010.

Mikolov T., Kombrink S., Burget L., Cernocky J., Khudanpur S., « Extensions of recurrent neural network language model. », *ICASSP*, IEEE, p. 5528-5531, 2011.

Mikolov T., Yih W., Zweig G., « Linguistic Regularities in Continuous Space Word Representations », *Human Language Technologies : Conference of the North American Chapter of the Association of Computational Linguistics*, p. 746-751, 2013b.

Pennington J., Socher R., Manning C. D., « GloVe : Global Vectors for Word Representation », *Empirical Methods in Natural Language Processing (EMNLP)*, p. 1532-1543, 2014.

Ramshaw L., Marcus M., « Text chunking using transformation-based learning », *Proceedings of the 3rd Workshop on Very Large Corpora*, Cambridge, MA, USA, p. 84-94, June, 1995.

Raymond C., Riccardi G., « Generative and Discriminative Algorithms for Spoken Language Understanding », *Proceedings of the International Conference of the Speech Communication Assosiation (Interspeech)*, Antwerp, Belgium, p. 1605-1608, August, 2007.

Schuster M., Paliwal K., « Bidirectional Recurrent Neural Networks », *Trans. Sig. Proc.*, vol. 45, nº 11, p. 2673-2681, nov, 1997.

Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., « Dropout : A Simple Way to Prevent Neural Networks from Overfitting », *Journal of Machine Learning Research*, vol. 15, p. 1929-1958, 2014.

Vukotic V., Raymond C., Gravier G., « Is it time to switch to Word Embedding and Recurrent Neural Networks for Spoken Language Understanding ? », *InterSpeech*, Dresde, Germany, September, 2015.

Vukotic V., Raymond C., Gravier G., « A step beyond local observations with a dialog aware bidirectional GRU network for Spoken Language Understanding », *Interspeech*, San Francisco, United States, September, 2016.

Werbos P., « Backpropagation through time : what does it do and how to do it », *Proceedings of IEEE*, vol. 78, p. 1550-1560, 1990.

Yao K., Zweig G., Hwang M.-Y., Shi Y., Yu D., « Recurrent Neural Networks for Language Understanding », Interspeech, August, 2013.

AGOHRA : génération d'une ontologie dans le domaine des ressources humaines

Rémy Kessler[*] [1] — **Guy Lapalme**[**]

[*] *IRISA - UMR 6074, Université de Bretagne-Sud, 56017 Vannes, France*
kessler@univ-ubs.fr
[**] *RALI - Département d'informatique et de recherche opérationnelle*
Université de Montréal
C.P. 6128, Succ Centre-Ville, Montréal, Québec, Canada H3C 3J7
lapalme@iro.umontreal.ca

RÉSUMÉ. *Nous présentons une méthode d'analyse de corpus afin de générer une ontologie dans le domaine du e-recrutement. Notre approche de construction semi-automatique s'appuie sur des millions de profils issus de plusieurs réseaux sociaux et sur des dizaines de milliers d'offres d'emploi collectées sur Internet pour faire émerger des compétences et des connaissances communes afin de construire un ensemble structuré de termes et concepts représentant chaque métier. Notre approche combinant statistiques et extraction de n-grammes de mots permet de créer une ontologie en anglais et en français contenant 440 métiers issus de 27 domaines d'activité. Chaque métier est ainsi relié aux compétences nécessaires à sa pratique pour un total d'environ 6 000 compétences différentes. Une évaluation manuelle sur une portion de l'ontologie a été réalisée par un expert du recrutement et a montré des résultats de très bonne qualité.*

ABSTRACT. *We describe a corpus analysis method for generating an ontology in the field of e-recruitment from millions of user profiles gathered on social networks and tens of thousands of job offers collected over the internet. Using statistics and n-gram analyses, we create a structured set of terms and concepts in English and French for 440 occupations in 27 fields of activity. Each occupation is linked with the necessary practice skills for around 6000 different skills. A manual evaluation of the results was performed by a domain expert and has shown excellent results.*

MOTS-CLÉS : *ontologie, réseaux sociaux, ressources humaines, application industrielle.*

KEYWORDS: *ontology, social networks, human resources, industrial application.*

1. Travail effectué lors d'un stage postdoctoral à l'Université de Montréal

1. Introduction

Les développements rapides du Web et des réseaux sociaux durant la dernière décennie ont considérablement modifié la dynamique de recherche d'emploi comme le décrivent Sivabalan *et al.* (2014) . Les informations professionnelles publiées par les utilisateurs dans leurs profils (formations, antécédents de travail, résumé de carrière, liens sociaux, etc.) peuvent être exploitées par les recruteurs pour identifier de nouveaux candidats ou pour obtenir des informations complémentaires à leur propos.

D'après une étude de RegionsJob[1] (2011), « 43 % des recruteurs avouent recourir à des recherches de type nom/prénom sur les candidats qui postulent chez eux et 8 % des recruteurs interrogés déclarent avoir écarté un candidat à cause de traces jugées négatives trouvées en ligne ». La plupart de ces recherches étant effectuées de façon rapide et manuelle, les informations recueillies sur un individu à un premier niveau de recherche sont peu structurées, disparates, incomplètes, redondantes, parfois obsolètes et peuvent être biaisées, voire trompeuses (p. ex. à cause des homonymes).

La plupart des systèmes d'appariement entre une offre d'emploi et un profil s'appuient sur une ou plusieurs ressources linguistiques, coûteuses en entretien et en mise à jour. En effet, les métiers d'aujourd'hui ne sont pas forcément les mêmes que ceux d'hier, et ne seront peut-être pas identiques aux métiers de demain. L'évolution de notre société entraîne l'apparition de nouveaux métiers et de nouvelles compétences ainsi que la disparition d'autres. Les résultats d'une étude menée par la Harvard Business Review, citée par le journal *Libération*[2], indiquent que plus de 31 % de nouveaux métiers apparaissent chaque année et que 60 % des emplois actuels disparaîtront au cours des deux prochaines décennies. Afin de pallier ce problème, nous souhaitons développer un système pour générer une ontologie de façon semi-automatique en s'appuyant sur des données collectées sur Internet. Nous supposons que l'exploitation de ces informations est une voie prometteuse pour constituer un référentiel commun, une représentation de chaque domaine avec des liens logiques reliant chaque métier aux compétences nécessaires à sa pratique. Même si chaque profil est différent dans le détail, nous faisons l'hypothèse qu'un regroupement d'informations issues des mêmes métiers fera émerger des relations communes pour construire un ensemble structuré de termes et de concepts représentant chaque domaine. Même si la structure de l'ontologie reste simple, la génération de ces ressources, tout en restant automatisée, permettra de créer une représentation de la connaissance de chaque domaine qui pourra être exploitée par la suite dans le cadre de l'appariement (p. ex. pour évaluer si un candidat a toutes les compétences pour un métier) ou au travers d'expansion de requêtes (p. ex. afin de trouver les candidats les plus compétents pour un poste) ou de la génération de texte (p. ex. pour suggérer à un candidat comment mettre en avant ses compétences les plus en adéquation pour un poste).

1. https://entreprise.regionsjob.com/enquetes/reseaux_sociaux/resultats_enquete_2.pdf
2. http://www.liberation.fr/evenements-libe/2016/05/10/l-intelligence-artificielle-au-service-de-l-emploi_1451670

Dans le cadre du projet de recherche Butterfly Predictive Project[3] (BPP), nous développons une plateforme pour améliorer l'appariement entre des candidats et des offres d'emploi. Nous faisons l'hypothèse que l'acquisition et l'exploitation des traces laissées par les individus sur les réseaux sociaux (LinkedIn, Viadeo, etc.), diffuses, plus ou moins accessibles et peu structurées sont une voie prometteuse pour les recruteurs afin de déterminer le positionnement professionnel des candidats et faciliter leur mise en correspondance avec des emplois à pourvoir.

Même si de grands efforts ont été déployés ces dernières années afin de développer des ressources linguistiques et aider les systèmes d'appariement de candidatures et d'offres d'emploi, ces derniers se heurtent à la difficulté de créer des ontologies ou des taxonomies spécifiques à chaque domaine et particulièrement à leur entretien et leur mise à jour. Nous présentons ici une approche de génération dynamique d'ontologies avec une possibilité de mise à jour continue au fur et à mesure que le Web évolue.

Une version préliminaire de ce travail a été présentée à CORIA 2016 (Kessler *et al.*, 2016). Cet article présente plusieurs extensions telles que la recherche de compétences transversales, le dictionnaire dynamique ou encore le regroupement des synonymes qui ont grandement amélioré tant la qualité que la quantité des informations récoltées. Dans la section suivante, nous présentons des travaux liés à notre étude. La section 3 présente les ressources utilisées tandis que la méthodologie est détaillée en section 4. La section 5 décrit l'ensemble du système, avant de présenter les résultats et leur évaluation dans la dernière section.

2. Travaux connexes

De nos jours, des centaines de milliers de candidats mettent en ligne leur profil, et les entreprises ou les établissements publient une quantité importante de postes recherchés. Analyser automatiquement cette quantité d'informations pour mettre en correspondance emplois et candidats est une tâche difficile. Comme le décrivent Yahiaoui *et al.* (2006), cet appariement repose, d'une part, sur la connaissance des individus et de leurs compétences et, d'autre part, sur la connaissance des métiers. De grands efforts ont ainsi été déployés ces dernières années afin de constituer des ressources linguistiques pour améliorer les systèmes d'appariement.

Lau et Sure (2002) développent une ontologie, en se fondant sur une étude de cas de la société Swiss Life, centrée sur le domaine des technologies de l'information. Ils précisent que celle-ci a finalement été construite manuellement même si des approches semi-automatiques avaient été tentées, mais les résultats ne permettaient pas d'obtenir une représentation claire et structurée des compétences.

Les premiers travaux dans l'appariement de candidatures et d'offres d'emploi à l'aide d'une ontologie ont été proposés par Colucci *et al.* (2003 ; 2007). Leur système,

3. http://rali.iro.umontreal.ca/rali/?q=fr/butterfly-predictive-project

IMPAKT[4](Colucci *et al.*, 2013) permet d'extraire des compétences de CV et repose sur des méthodes de formalisation des raisonnements. Le système propose un appariement en effectuant des correspondances partielles ou complètes des compétences entre les offres d'emploi et les candidatures. Zimmermann *et al.* (2016) extraient les informations des CV en combinant apprentissage machine et traitement de la langue pour déterminer les candidatures les plus pertinentes en fonction d'un emploi déterminé.

Desmontils *et al.* (2002) et Trichet *et al.* (2004) décrivent une méthode d'indexation sémantique de CV. Celle-ci exploite les caractéristiques dispositionnelles du document afin d'identifier chacune des parties et de les indexer en conséquence. Ils proposent une instanciation de l'ontologie en partant des données récoltées et en s'appuyant sur des ressources externes (base ROME[5] et CIGREF[6]). Même si l'approche semble intéressante, l'absence de résultat ne permet pas d'évaluer l'apport de cette indexation particulière. Kmail *et al.* (2015) proposent une approche similaire en construisant des réseaux sémantiques à partir des candidatures et des offres d'emploi et enrichissent ceux-ci à l'aide de ressources externes telles que WordNet et YAGO2 (Hoffart *et al.*, 2011). Mochol et Simperl (2006) décrivent l'importance d'une ontologie commune (HR ontology) ainsi qu'un guide pour mettre en place ce type d'application tandis que Trichet *et al.* (2004) et Yahiaoui *et al.* (2006) décrivent différentes approches pour l'annotation sémantique de document et la gestion des compétences à l'aide d'ontologie dans le cadre du e-recrutement.

Dans le cadre du projet Prolix, Trog *et al.* (2008) décrivent une ontologie de ressources humaines en s'appuyant sur le cas de British Telecom. Ils proposent une architecture en plusieurs niveaux en fonction des compétences, des interactions et du contexte. Gómez-Pérez *et al.* (2007) proposent, quant à eux, une annotation sémantique des documents (offres d'emploi et CV) afin de construire différentes ontologies. Développées en anglais et disponibles en ligne[7], ces ontologies décrivent des compétences et des formations spécifiques au domaine des technologies de l'information tandis que d'autres sont plus généralistes comme les classes de métiers, les permis de conduire ou encore les secteurs d'activité.

Roche et Kodratoff (2006) présentent une extraction de terminologie sur un corpus de CV. Leur approche extrait un certain nombre de collocations contenues dans les CV sur la base de patrons (tels que nom-nom, adjectif-nom, nom-préposition-nom, etc.) et les classe en fonction de leur pertinence en vue de la construction d'une ontologie spécialisée.

Partant du constat que les médias sociaux deviennent une source incontournable pour les recruteurs dans leur recherche de candidats, Tétreault *et al.* (2011) décrivent

4. Information Management and Processing with the Aid of Knowledge-based Technologies.
5. Répertoire organisationnel des métiers et emplois : `http://www.pole-emploi.fr/candidat/le-code-rome-et-les-fiches-metiers-@/article.jspz?id=60702`
6. Club informatique des grandes entreprises françaises.
7. `http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/ontologies/99-hrmontology`

la maquette d'une plateforme de recrutement intégrant les technologies du Web sémantique ainsi que l'élaboration d'une ontologie à l'aide du *Web Ontology Language* (OWL) dédiée au domaine TI. L'approche présente les avantages d'une application de ce type et différents scénarios de recrutement.

Plus récemment, le Vrang *et al.* (2014) présentent l'ontologie ESCO[8], un projet européen multilingue de classification de compétences, de métiers et de certifications afin de créer une harmonisation européenne en matière de recrutement. Cependant, même si le modèle est d'excellente qualité, il n'en est actuellement qu'à la version 0.1, ce qui restreint les domaines et les métiers couverts ainsi que les types de relation. Une autre limite de cette approche est son caractère rigide et figé. En effet, l'évolution d'une telle ressource est complexe puisqu'elle nécessite des connaissances dans plusieurs domaines ainsi que dans de nombreuses langues, ce qui passe par une longue phase manuelle de saisie et de vérifications.

La version actuelle regroupe 4 761 métiers et 5 096 compétences en 24 langues différentes pour environ 250 000 termes différents (21 000 termes en anglais et 18 000 en français). Chaque métier ou compétence est défini dans chaque langue par un *label préféré* ainsi que par un ou plusieurs *labels alternatifs* pouvant être un synonyme, une féminisation, une forme abrégée ou une variation orthographique. Chaque métier est en outre relié aux compétences nécessaires à sa pratique et de la même façon chaque compétence est reliée aux différents métiers. La version actuelle n'a qu'un seul type de relation `is-related-to`, mais d'autres relations sont prévues dans la prochaine version de cette ontologie[9].

Ces travaux montrent que différentes approches de construction d'ontologies dans le domaine du recrutement ont été envisagées, que ce soit avec des méthodes statistiques, sémantiques ou encore à l'aide de ressources linguistiques complémentaires. Même s'il existe d'autres approches et domaines pour lesquels des méthodes de construction semi-automatique d'ontologies ont été développées, par exemple l'astronomie (Bendaoud *et al.*, 2007), les dépêches journalistiques (Sami *et al.*, 2014) ou encore le domaine médical (Osborne *et al.*, 2009), nous observons que les travaux dans le domaine de l'e-recrutement reposent sur des créations manuelles d'ontologies et non sur une analyse de corpus.

L'originalité de notre approche est d'exploiter les réseaux sociaux et plus généralement l'information récoltée sur Internet pour mettre à jour la nomenclature des métiers et des compétences afin de construire un ensemble structuré de termes et de concepts et c'est ce que nous explorons dans cet article.

8. European Skills Competences and Occupations `https://ec.europa.eu/esco/`
9. `https://ec.europa.eu/esco/portal/escopedia/ESCO_data_model`

3. Données et connaissances

Une ontologie est un modèle de connaissances constitué de concepts relatifs à un domaine ainsi que des relations entre ces concepts. Afin de construire ce modèle, nous nous appuyons sur diverses données et connaissances, issues d'Internet ou fournies par notre partenaire industriel Little Big Job (LBJ). Le projet BPP s'appuie sur une modélisation en 44 secteurs d'activité (univers) utilisée par LBJ, chacun regroupant des familles de métiers assez proches et un ou plusieurs univers connexes. Les univers connexes à secteur donné sont ceux pour lesquels une transition est possible pour un candidat envisageant un changement de secteur. Par exemple, un candidat issu de l'univers *banque, finance, capital risque, fonds privés* pourra plus facilement envisager une transition vers les domaines connexes *assurances, mutuelles, prévoyance* ou *immobilier* que vers *cosmétique*.

Nous avons combiné ces connaissances avec l'ontologie issue du modèle ESCO décrit plus loin en 4, ainsi qu'un dictionnaire des métiers issus de la *Classification nationale des professions*[10] (CNP) et du *Répertoire opérationnel des métiers et des emplois (ROME)*[11]. La CNP est la référence reconnue des professions au Canada tandis que le ROME est la référence en France. La CNP répartit plus de 40 000 appellations d'emploi (anglais/français) en 500 profils de groupes professionnels tandis que le ROME comprend 10 000 appellations différentes réparties en 531 groupes ; pour notre projet, notre partenaire industriel nous a suggéré de nous limiter à un sous-ensemble de domaines qui l'intéressent soit 3 730 métiers. Nous avons par ailleurs observé des appellations différentes entre ROME et CNP suivant le pays (instituteur/enseignant, commis de bureau/réceptionniste, garde forestier/forestier, etc.), ce qui nous a conduits à conserver les deux classifications.

Par ailleurs, plus de dix millions de profils issus de plusieurs réseaux sociaux (*LinkedIn*, *Viadeo*, *Indeed* et d'autres) ont été récoltés à l'aide d'un processus de collecte automatique de sites Internet. Ces données issues de profils publics professionnels ont été préalablement anonymisées puis agrégées dans un format uniforme. L'origine géographique étant le Canada ou la France, les profils sont soit en français soit en anglais ou encore bilingues. Chaque profil résume différentes informations sur le parcours du candidat telles que ses diplômes et ses formations ainsi que leurs dates. De la même façon, une section du profil rend compte de ses expériences. Chacune contient plusieurs éléments tels que les dates de début et de fin, le nom de la société employeur (avec éventuellement une URL vers sa page Internet), la fonction occupée par le candidat au cours de cette expérience ainsi que le lieu et un éventuel descriptif de sa mission au sein de cette société. Un résumé des expériences sous forme d'un texte est par ailleurs présent ainsi qu'une courte description sous la forme d'une phrase d'accroche permettant au candidat de se décrire en quelques mots. D'autres informations sont récoltées ou éventuellement calculées telles que l'expérience totale du candidat,

10. http://www5.hrsdc.gc.ca/noc/

11. http://www.pole-emploi.fr/candidat/le-code-rome-et-les-fiches-metiers-@/article.jspz?id=60702

```
{ ...
"countryCode": "FR",
"createdDatetime": "2012-12-19 19:56:47",
"city": "Vannes",
"personalBrandingClaim": "Chef de projets au CHU de Vannes",
"personalBrandingPitch": "Professionnelle dynamique et proactive ayant eu l'opportunité de
            développerdes compétences variées tant dans le milieu de la santé que dans celui de
            l'aéronautique . Principalement axée sur la gestion, l'amélioration continue
            des processus et des pratiques et l'atteinte des objectifs"},
"educations":[{"name": "Marketing pharmaceutique",
               "schoolName": "Universite Claude Bernard (Lyon I)"},
              {"name": "MBA ", "schoolName": "Universite de Montreal - HEC Montreal"}],
"experiences":[{"function": "Chef de projets",  "companyName": "CHU de Vannes"},
               {"function": "Directrice du service 'a la clientele pharmaceutique",
                "companyName": "zootopia"},
               {"function": "Coordonnatrice de projets",  "companyName": "chu-saint-Trudeau"}]
"skills": ["E-commerce", "SDL Fredhopper", "Gestion de projet", "Gestion d'équipe",
          "Accessibilité"],
"languages": [{language:"Français", "level": "Native or bilingual"},
              {language:"Anglais", "level": "Native or bilingual"}],
...
}
```

Figure 1. *Extrait de la structure JSON d'un profil créé à partir d'informations collectées sur des réseaux sociaux*

	Canada	France
Nombre de profils	2 658 467	7 484 311
Moyenne du nombre d'expériences	3,2	2,3
Moyenne du nombre de formations	1,39	1,06
Moyenne du nombre de compétences	0,17	0,12
Statistiques textuelles des profils		
Profil vide	9,61 %	16,34 %
Moins de 100 caractères	26,95 %	40,53 %
Moins de 300 caractères	16,18 %	12,70 %
Moins de 500 caractères	6,43 %	4,66 %
Plus de 500 caractères	40,83 %	25,77 %

Tableau 1. *Statistiques de la collection de profils de réseaux sociaux*

les langues qu'il maîtrise, ses loisirs, le nombre de relations avec d'autres candidats ou encore les compétences acquises au cours de son parcours professionnel. La figure 1 présente un exemple de profil avec des informations extraites de réseaux sociaux.

Chaque profil regroupe une cinquantaine de champs, mais il existe cependant un nombre important de profils ne contenant pas ou peu d'informations comme le montre le tableau 1 qui présente quelques statistiques descriptives de cette collection. Dans le cadre de cette application, nous nous concentrons sur les 2,3 millions de compétences issues de ces profils.

En complément de ces données, 300 000 offres d'emploi ont été collectées sur Internet, 200 000 en anglais et 100 000 en français. Ces offres d'emploi couvrent un grand nombre de métiers et sont issues, elles aussi, du Canada ou de la France. Chaque offre d'emploi contient un titre, une description contenant le détail de l'offre d'emploi, la date de mise en ligne, le lieu de la mission proposée ainsi que le nom de la compagnie qui recrute. Les premières observations ont montré que, bien qu'extrêmement bruitées, ces données peuvent constituer une source intéressante d'informations afin de constituer une ontologie de façon semi-automatique.

4. Méthodologie

Nous présentons tout d'abord le modèle ESCO (le Vrang *et al.*, 2014) et sur lequel s'appuie notre démarche. Celui-ci est organisé selon le schéma de modélisation SKOS (Miles et Bechhofer, 2009). Il classe les connaissances disponibles en trois piliers : professions, compétences et qualifications. Dans le cadre de ces travaux, nous nous concentrons sur les professions et les compétences. Le modèle se structure par la suite en concepts et en termes. Chaque profession, compétence et qualification est associée à un concept et est identifiée de façon unique par un *uniform resource identifier (URI)*. Chaque concept comporte au moins une étiquette préférée alors que des possibles synonymes, des variantes d'orthographe et des abréviations sont enregistrés comme des étiquettes alternatives. Des relations sont créées manuellement entre chaque profession et les compétences nécessaires à leur pratique.

Dans un premier temps, nous avons effectué une comparaison entre les compétences issues des réseaux sociaux et celles issues de l'ontologie ESCO. Les résultats ont montré de nombreuses correspondances exactes ou partielles (plus d'un million) et variées. Les correspondances partielles ont été calculées, dans un premier temps, en comparant uniquement les quatre premiers caractères de chaque compétence puis dans un second temps avec la mesure de Levenshtein (Levenshtein, 1966). Nous observons que les compétences les plus fréquentes dans les données issues des réseaux sociaux se retrouvent bien dans l'ontologie ESCO.

Cette approche a cependant montré certaines limites. Même si le modèle ESCO est d'excellente qualité, les domaines et métiers couverts par l'ontologie restent limités (par exemple, on ne retrouve pas *business analyst* ou *account manager*, pour les métiers, ni *marketing strategy*, *financial modelling*, *food cost management* pour les compétences). Un enrichissement d'ESCO a été envisagé, cependant le modèle de 44 univers du partenaire industriel était incompatible avec les regroupements par domaines effectués dans ESCO. Quant aux classifications ROME et CNP (décrites dans la section précédente), elles ont été développées dans un cadre administratif afin de codifier les professions pour les besoins des bureaux d'immigration et de statistiques gouvernementales. Si ces organismes cherchent bien à couvrir tous les métiers (y compris celui de Premier ministre ou député), ils ont recours à une nomenclature très différente : nous n'observons que 73 appellations identiques entre les classifications ROME et CNP, 29 entre ESCO et ROME et seulement 13 entre ROME, CNP

et ESCO ; de plus, ces classifications gouvernementales ne font aucun lien entre métiers et compétences qui est pourtant le but que nous cherchons à atteindre avec notre ontologie.

Par ailleurs, il est extrêmement difficile de discerner dans les profils des candidats les compétences issues d'une expérience plutôt que d'une autre : par exemple, le profil d'un chef de projet senior en technologies de l'information cumulera des compétences en tant que développeur issues de ses premières expériences, avec celles, par la suite, de chef de projet. Il est donc difficile de se restreindre à cette unique source pour déterminer les compétences requises pour un emploi.

Afin de pallier ces problèmes, nous avons décidé d'utiliser comme source de documents pour l'ensemble des offres d'emploi. En effet, ces offres définissent les expériences et qualifications désirées pour un métier donné. Comme nous disposons d'un nombre important d'offres d'emploi (300 000), nous croyons être en mesure de faire émerger les compétences et les connaissances communes pour construire un ensemble structuré des termes et des concepts représentant chaque métier. L'ensemble des offres d'emploi étant collecté sur Internet, le modèle pourra s'enrichir, à terme, de façon semi-automatique avec l'apparition et la disparition de métiers dans les offres d'emploi.

5. Vue d'ensemble du système AGOHRA

La figure 2 présente une vue d'ensemble du système AGOHRA[12] dont les étapes seront détaillées dans le reste de la section. Au cours d'une première étape ①, nous effectuons une normalisation des offres d'emploi. Le module suivant ② utilise les titres des offres d'emploi afin de détecter les métiers avant d'y associer un univers ③ (voir section 3 pour la notion d'univers). Une première recherche au cours de l'étape ④ est effectuée par la suite, afin de constituer un dictionnaire composé uniquement de *compétences transversales*[13]. L'étape suivante ⑤ consiste à utiliser l'ensemble du vocabulaire récolté afin de faire émerger les compétences. À l'aide de la base de profils ainsi que de dictionnaires constitués de façon dynamique, le module suivant ⑥ classe par la suite le vocabulaire obtenu afin de déterminer s'il s'agit de compétences. Le dernier module ⑦ transforme ensuite les informations ordonnées et structurées en une ontologie au format RDF.

5.1. *Normalisation des offres d'emploi*

Nous effectuons au préalable une extraction du contenu textuel des offres d'emploi au format HTML. Au cours de l'étape ①, nous effectuons une séparation des

12. *Automatic Generation of an Ontology for Human Resource Applications*
13. Les compétences transversales, aussi appelées *soft skills*, sont des compétences personnelles et sociales, orientées vers les interactions humaines.

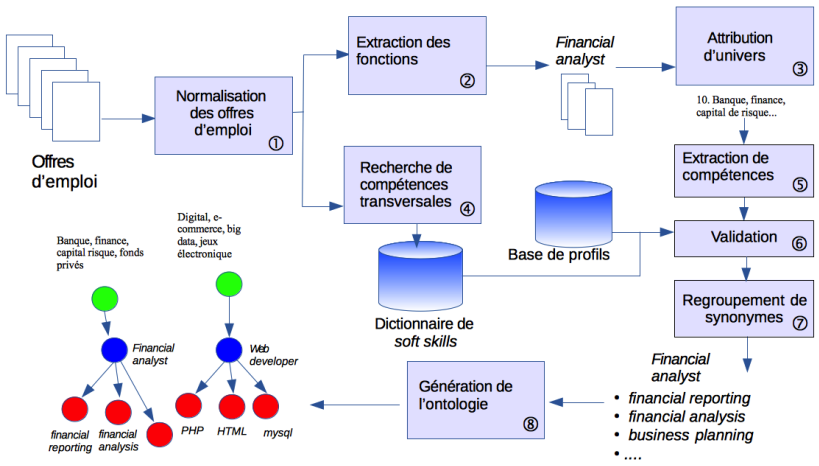


Figure 2. *Vue d'ensemble du système AGOHRA*

offres en français et en anglais ainsi qu'un filtrage des quelques offres dans les autres langues (environ 150 offres sont en espagnol, japonais, italien, etc.). L'observation de l'ensemble des offres montrant un grand nombre de doublons (offres similaires avec des dates différentes, offres similaires pour des lieux différents, etc.), nous effectuons un dédoublonnage des offres d'emploi afin d'éviter d'augmenter artificiellement les fréquences des termes présents dans plusieurs offres pratiquement identiques. Environ 6 000 offres d'emploi sont ainsi écartées. Toujours au cours de cette étape, une normalisation des titres des offres d'emploi est effectuée afin de détecter les métiers en utilisant différents patrons afin d'améliorer la qualité du processus. On supprime ainsi les accents, les caractères particuliers tels que « - », « / », « ( ) » ou encore les expressions couramment utilisées dans les titres d'offres d'emploi telles que *full-time*, *temporary*, ou encore *entry level*. Différents processus linguistiques sont utilisés afin de réduire le bruit dans le modèle : les expressions courantes (par exemple, *c'est-à-dire*, *chacun de*, etc.), les chiffres et les nombres (numériques et/ou textuels), les symboles spéciaux ainsi que les termes contenus dans un antidictionnaire adapté à notre problème.

Nous avons également constitué une liste de noms de métiers normalisés regroupant sous un terme unique les différentes écritures pour chaque métier (féminin, pluriel, erreurs typographiques, etc.) classées selon leur fréquence d'apparition dans le

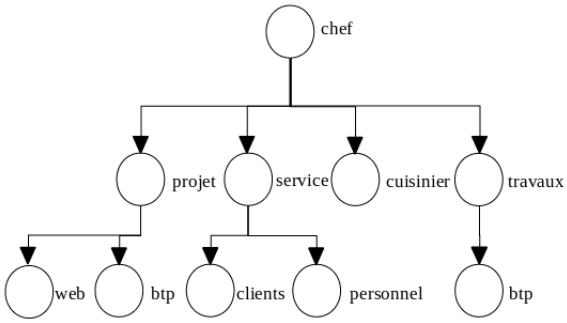


Figure 3. *Extrait de l'arbre de métiers avec* chef *comme premier nœud ce qui permet de retrouver* chef projet web*,* ... chef cuisinier*,* ... chef travaux btp*.*

champ, en fonction des expériences issues des profils de réseaux sociaux. On observe par exemple 292 façons différentes (bien comptées !) d'écrire la fonction *développeur* dans l'ensemble (p. ex. *développeur, developper, développeuse, dévelloppeur, developer, developpper,* etc.). Un processus de lemmatisation avait été intégré lors des premières expériences, mais il s'est avéré peu performant, car la plupart des noms d'emploi ne sont pas déclinés et ne posaient aucun problème compte tenu de cette normalisation qui est mieux adaptée à notre application.

5.2. *Extraction des fonctions et attribution d'univers*

Nous utilisons au cours de cette étape ② les titres des offres d'emploi normalisés (voir 5.1) afin de construire une liste de métiers. À l'aide de règles, le système compare les titres avec un arbre préfixe contenant la liste de métiers normalisés décrite dans la section 5.1. Chaque métier est ainsi transformé en un ensemble de nœuds où il existe un nœud pour chaque mot, comme le montre l'exemple de la figure 3.

Cette structure permet d'effectuer des comparaisons rapides entre la liste des métiers et les titres tout en conservant les variations d'écriture. Nous conservons ainsi la chaîne la plus longue comme métier reconnu. Grâce à un seuil de fréquences minimal déterminé empiriquement, nous ne conservons que les métiers avec une fréquence importante. Une première version du système traitait de l'ensemble des métiers présents dans les offres d'emploi, mais nous avons décidé de privilégier les métiers ciblés par LBJ, notre partenaire industriel, dont l'activité est centrée sur le recrutement dans les domaines de la gestion, de la finance ou de la haute technologie.

En retirant les métiers qui n'intéressaient pas notre partenaire industriel, comme *gardienne de chat*, *nounou*, *camionneur*... nous avons filtré une grande quantité d'offres d'emploi. Nous en conservons tout de même 45 000 (25 000 en anglais et 20 000 en français). L'étape ③ consiste à attribuer un univers à chacun des noms de métiers obtenu.

Afin d'associer un univers à chaque nom de métier, nous avons expérimenté plusieurs approches à l'aide d'algorithmes de type machine à vecteurs de support ou de *boosting*. Nous avons transformé le texte en vecteurs de mots afin de détecter de manière automatique l'univers associé à chaque métier. Après avoir découpé l'ensemble des offres d'emploi en cinq sous-ensembles approximativement de la même taille, nous avons appliqué la procédure suivante : quatre des cinq sous-ensembles ont été concaténés pour produire un corpus d'entraînement et le cinquième a été utilisé pour le test. La procédure a été effectuée cinq fois afin que chacun des sous-ensembles du corpus d'apprentissage soit utilisé une fois pour le test. Cependant, les chevauchements de vocabulaire entre univers pour certaines offres d'emploi ont rendu la tâche délicate (par exemple des offres d'emploi d'analystes financiers travaillant pour des entreprises qui ne sont pas dans le domaine de la finance ou des postes en informatique pour un groupe de cosmétique, etc.). Nous avons finalement décidé d'associer manuellement à cette liste un univers pour chaque métier.

5.3. *Extraction de compétences*

L'objectif de l'étape ⑤ est d'utiliser le vocabulaire issu des offres d'emploi afin de faire émerger les compétences associées à chacun des métiers. Nous effectuons pour cela une agrégation du vocabulaire en le regroupant par métier. L'extraction d'information dans une offre d'emploi n'est pas une tâche triviale comme le soulignent Loth *et al.* (2010). Kessler *et al.* (2008) montrent qu'en raison d'une grande variété dans les paramètres (texte libre, tailles différentes, découpage incertain, délimiteurs variés), le découpage d'offres d'emploi en blocs d'information est une tâche délicate. Ces offres apparaissent cependant dans un ordre conventionnel. Afin de diminuer la taille du vocabulaire considéré, nous recherchons différents motifs séparateurs et fréquents dans une offre d'emploi, tels qu'*exigences*, *qualifications*, *responsibilities*, etc. Ces motifs, bien que pas toujours présents, permettent ainsi de réduire considérablement le vocabulaire en ne prenant en compte que la partie de l'annonce suivant le motif.

L'observation des compétences de l'ensemble des profils de réseaux sociaux décrit en section 3 (champs `skills` de la figure 1) montre que plus de 80 % des compétences se présentent sous la forme de n-grammes de mots (par exemple *financial modelling*, *php development*) répartis comme suit : 24 % d'unigrammes, 42 % de bigrammes et 20 % de trigrammes, 8 % de 4-grammes, 4 % de 5-grammes et 1 % vide (c'est-à-dire des profils sans aucune compétence). Compte tenu de ces observations, nous avons décidé de transformer l'ensemble du vocabulaire issu des offres d'emploi sous forme d'unigrammes, de bigrammes et trigrammes et de les ordonner selon un score S_{job}, inspiré du *TF-IDF* :

$$S_{job}(u, m) = tf(u).log\frac{D_m}{df(u)} \quad [1]$$

avec u l'unité lexicale considérée (unigrammes, bigrammes ou trigrammes), $tf(u)$ la fréquence de u dans la collection, $df(u)$ le nombre de documents où l'unité lexicale u apparaît et D_m le nombre de documents associés au métier m.

Afin de filtrer certaines compétences courantes dans les offres d'emploi telles que les compétences « Microsoft » (par exemple *microsoft office*, *suite office*, *word*, etc.) et qui viennent parfois bruiter les résultats, nous avons mis en place un certain nombre de règles complémentaires. Nous avons fait de même pour les compétences de langues (*anglais écrit*, *français*, *bilingue*, etc.) qui, bien qu'essentielles, n'étaient pas des compétences représentatives de chaque métier. Ces règles ont été développées pour répondre à des suggestions de LBJ, notre partenaire industriel, suite à l'examen des résultats d'une version préliminaire du système.

5.4. *Recherche de compétences transversales*

Les premiers résultats ont montré qu'un grand nombre de compétences extraites étaient des *compétences transversales* (telles que *verbal/written communication skills*, *capacité à travailler en équipe*, etc.) qui peuvent être considérées comme pertinentes quel que soit le métier considéré. Nous avons donc ajouté l'étape ④ afin de les distinguer des compétences plus techniques, communément appelées *hard skills*[14]. Les compétences transversales étant des compétences demandées dans l'ensemble des offres d'emploi et quelle que soit la fonction considérée, nous effectuons un premier traitement en réutilisant l'ensemble des offres d'emploi sans tenir compte des fonctions. Nous recherchons ainsi les compétences les plus fréquentes quel que soit le métier considéré. Après l'étape de normalisation (section 5.1), nous effectuons une agrégation du vocabulaire décrite en section 5.3, que nous transformons par la suite sous forme d'unigrammes, de bigrammes et de trigrammes et que nous ordonnons en fonction de S_{job}, tel que défini par l'équation [1]. L'ensemble des compétences obtenues permet de constituer un dictionnaire qui sera utilisé par la suite, au cours de l'étape de validation (section 5.5), afin de séparer les compétences transversales des compétences plus techniques. Nous obtenons ainsi un dictionnaire de 250 termes en français et en anglais, regroupant unigrammes, bigrammes et trigrammes.

5.5. *Validation des compétences*

Une première version du module ⑥ effectuait une comparaison de la liste ordonnée de n-grammes obtenus avec un dictionnaire contenant environ 25 000 compétences issues des profils de réseaux sociaux afin de valider ou invalider les n-grammes. Cette méthode était relativement efficace pour écarter les n-grammes qui n'étaient pas des compétences tels certains termes ou expressions courants dans les offres d'emploi (par

14. Les *hard skills* sont les compétences formellement démontrables, nées d'un apprentissage technique, souvent d'ordre académique, et dont la preuve est apportée par l'obtention de notes, de diplômes, de certificats.

exemple *employment equity*, *strong experience required*, etc.). Mais elle ne permettait pas de prendre en considération la spécificité de certaines compétences vis-à-vis de certains métiers. Afin de résoudre ce problème, nous avons constitué un dictionnaire *dynamique*. Pour le construire, les 10 millions de profils ont été indexés avec le moteur de recherche Lucene[15] en fonction de la langue du profil et du pays d'origine. Chaque champ textuel a été indexé individuellement afin de pouvoir être interrogé séparément. Cette approche affine les résultats en fonction du secteur d'activité, de la fonction occupée par le profil ou des compétences qu'il possède. Pour chaque métier sélectionné au cours de l'étape 5.2, nous effectuons une requête avec le nom de ce métier et en spécifiant que les résultats doivent contenir uniquement des profils dont la section *compétences* n'est pas vide. Les profils retournés sont donc des profils qui occupent ou qui ont occupé le métier recherché et dont la section *compétences* a été renseignée. Pour une requête donnée, nous agrégeons les compétences des 10 000 premiers profils[16] obtenus. Les compétences sont ainsi transformées en une liste ordonnée par fréquence avec comme seuil minimal 10 % de la fréquence maximale obtenue. Chaque liste constitue ainsi un dictionnaire *dynamique* des compétences les plus fréquentes pour chaque métier selon la base de profils de réseaux sociaux. Nous comparons la liste ordonnée de n-grammes obtenus avec ce dictionnaire dynamique et avec le dictionnaire de *soft skills* afin de séparer les compétences en fonction de leur type comme expliqué en section 5.4.

5.6. *Regroupement de synonymes*

L'analyse de la première version de l'ontologie a montré qu'un grand nombre de compétences, seul l'adjectif étant différent, pouvaient être regroupées en une seule (telles que *good work ethic / strong work ethic*, *bonne capacité adaptation / excellente capacité adaptation*, etc.). Un module spécifique ⑦ recense l'ensemble des adjectifs issus de la collection de profils de réseaux sociaux afin de constituer une liste des plus fréquents. Même si certains d'entre eux sont classiques (par exemple *good*, *excellent*), d'autres n'ont de sens que dans le domaine particulier du recrutement et de la qualification de compétences (tels que *strong*, *outstanding*). Nous utilisons par la suite cette liste afin de constituer plusieurs motifs pour sélectionner, pour chaque type de n-grammes, un sous-ensemble de compétences susceptibles d'être des synonymes une fois l'adjectif retiré. Toutes les occurrences des synonymes sont ensuite regroupées sous une compétence unique, les adjectifs classés par ordre alphabétique et séparés par un « / » (par exemple, *excellent/good/strong communication skills*).

15. http ://lucene.apache.org
16. Ce nombre a été déterminé de façon empirique.

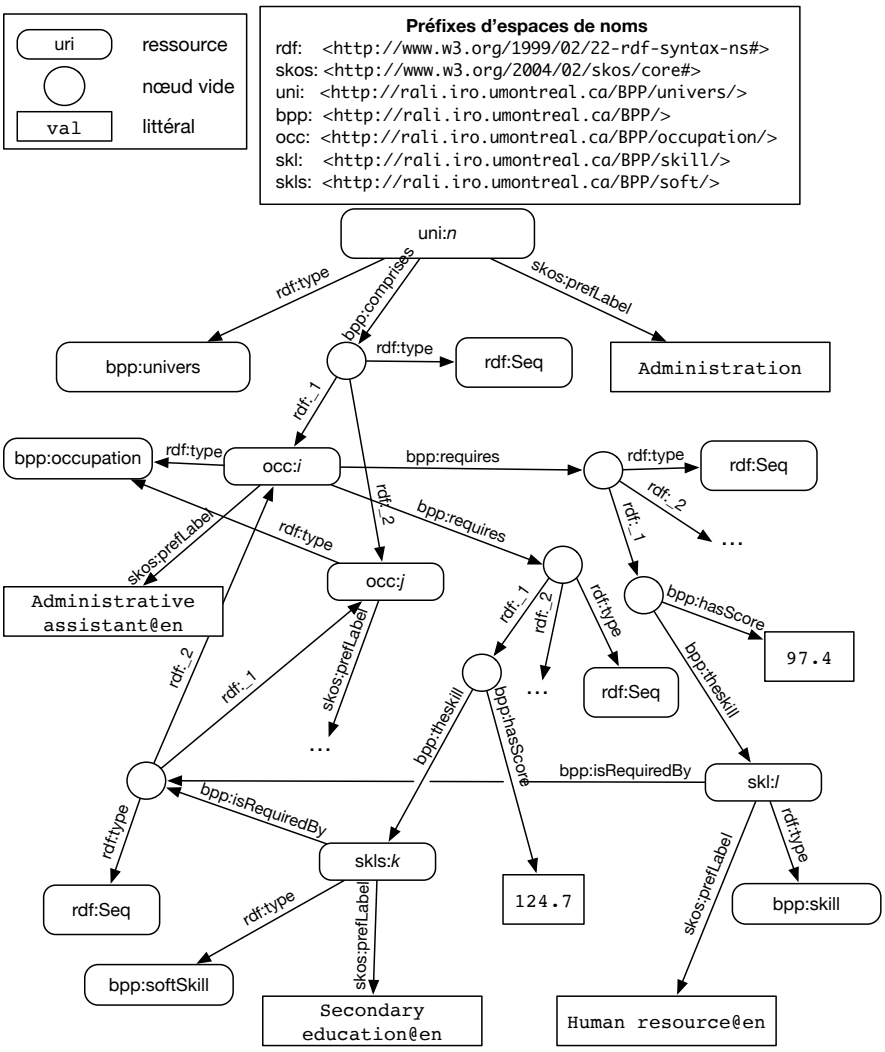


Figure 4. *Organisation du graphe RDF de l'ontologie générée*

5.7. *Création de l'ontologie*

À la suite de l'application des traitements linguistiques précédents sur l'ensemble de nos sources de données (bases de données spécialisées ou textes glanés du Web), nous avons décidé d'organiser le résultat sous forme d'une ontologie dont la structure reprend des éléments de l'ontologie ESCO (liens entre métiers et compétences), mais en y intégrant la notion d'univers définie par LBJ (section 3).

Notre ontologie est une classification hiérarchique, exprimée en RDF, dans laquelle chaque univers (`uni:n` dans la figure 4) est lié par la relation `bpp:comprises` à un ensemble de métiers `occ:i` nommés *occupations* par souci de compatibilité avec la nomenclature ESCO. Chaque *occupation* est liée par la relation `bpp:requires` à deux séquences (`rdf:Seq`) de compétences : une pour les compétences transversales (`bpp:softskill`) et une autre pour les compétences spécialisées (`bpp:skill`). Chaque compétence est liée à une étiquette par la relation `skos:prefLabel` sous forme de n-grammes et associée à un score `bpp:hasScore` qui est la valeur de S_{job}. Chaque métier est ainsi rattaché à son domaine ainsi qu'à une liste des compétences associées. De plus, un lien direct `bpp:isRequiredBy` est ajouté entre chaque compétence et le métier qui l'exige. Même si chaque compétence est unique, celle-ci peut être reliée à plusieurs métiers. Notons que conformément aux principes du Web sémantique, une classe RDF définit l'ensemble des individus qui partagent des propriétés communes, il n'y a donc pas d'instanciation d'objets à partir des classes comme dans les modèles à objets plus *classiques*.

Le module ⑧ crée une ontologie[17] composée des titres de 440 métiers (128 en anglais et 312 en français) répartis dans 27 univers différents (sur les 44 univers possibles) et reliés à 6 226 compétences différentes (4 059 pour l'anglais et 2 167 pour le français) et 485 compétences transversales (259 pour l'anglais et 226 pour le français). Des exemples de deux métiers avec leurs compétences identifiées sont présentés dans les tableaux 3 et 4. Plusieurs métiers peuvent partager des compétences qui ne sont pas répétées dans l'ontologie. La structure permet également de trouver tous les métiers qui demandent une compétence particulière.

Le réseau RDF pour l'anglais contient 109 156 triplets pour 128 métiers reliés à 4 059 compétences spécifiques et 259 compétences transversales et celui pour le français comprend 160 584 triplets pour 312 métiers reliés à 2167 compétences spécifiques et 226 compétences transversales. Ils sont interrogeables avec SPARQL dans nos applications. Pour en faciliter l'exploration visuelle, nous avons aussi développé un navigateur spécialisé qui affiche la structure RDF en faisant ressortir les différents niveaux : univers, métiers et compétences. La figure 5 présente une capture d'écran du navigateur contenant un extrait de l'ontologie finale avec un des métiers évalués en section 6. Le champ de texte en haut du navigateur permet de rechercher des métiers ou des occupations qui contiennent une chaîne. Il est ensuite possible d'explorer les compétences ou les métiers associés. Ce navigateur regroupe les résultats sous forme de triplets pour en faciliter l'évaluation, mais la structure de l'ontologie ne repose pas sur cette distinction qu'il est simple d'ignorer lors de requêtes SPARQL.

Cette première ontologie a une structure relativement simple, mais suffisante pour les besoins de notre application : elle est composée de listes d'occupations regroupées en univers, les occupations étant reliées à des listes de termes identifiant les compétences requises par ces occupations. Il aurait été intéressant de hiérarchiser les métiers

17. Disponible à `http://www-labs.iro.umontreal.ca/~lapalme/LBJ/BPPontologie/`

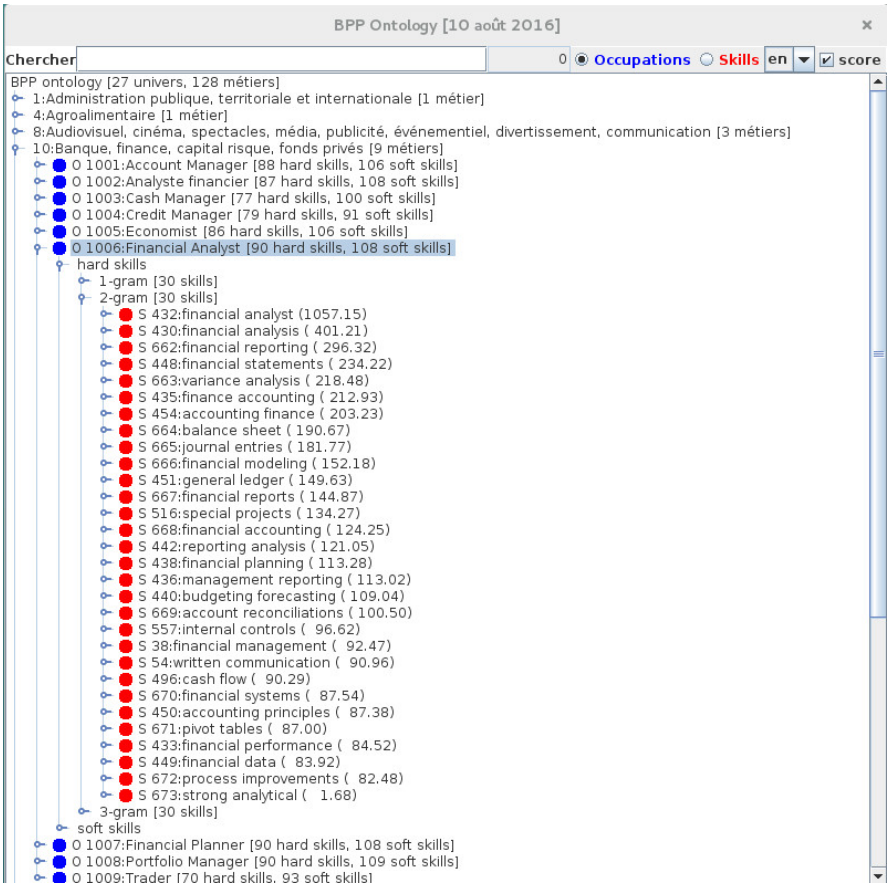


Figure 5. *Navigateur pour rechercher dans l'ontologie. Chaque concept présent dans l'ontologie est déterminé à l'aide d'un identifiant numérique, les compétences sont précédées de la lettre S (*Skills*) les métiers par O (*Occupations*), suivi du numéro associé à l'univers (par exemple O1006* financial analyst *appartient à l'univers 10* banque, finance, capital risque, fonds privés*). Le nombre entre parenthèses est la valeur du score calculée selon l'équation* [1] *(section 5.3).*

(p. ex. regrouper les *chefs de projets* de tous les domaines) ou de créer une structure d'héritage entre les compétences, mais ceci fera l'objet d'un travail futur.

6. Évaluation

Nous présentons maintenant nos résultats ainsi que son évaluation. Les tableaux 3 et 4 présentent les compétences obtenues, sous forme de n-grammes classés arbitrairement en fonction de la taille. Nous ne présentons ici que les dix premières compétences en ordre décroissant de S_{job} (voir section 5.3) pour les métiers *analyste financier* (Tableau 3) et *analyste programmeur* (Tableau 4) en français et en anglais. L'évaluation de cette portion de l'ontologie a cependant été effectuée sur les trente premières compétences par un expert du domaine. Nous avons choisi ces métiers afin de privilégier des métiers ciblés par notre partenaire industriel et pour ne pas nous limiter aux métiers des technologies de l'information, habituellement utilisés pour les évaluations dans la littérature. Nous présentons dans le tableau 2 un aperçu des métiers contenus dans l'ontologie dans chacune des langues, classés en fonction du nombre d'offres d'emploi utilisées par le système.

Anglais		Français	
Métier	# offres	Métier	# offres
sales representative	2 211	commercial	3 159
designer	1 764	développeur	2 692
administrative assistant	1 644	technicien commercial	1 451
customer service representative	1 309	comptable	1 363
account manager	1 183	contrôleur de gestion	1 000
developer	1 178	ingénieur commercial	983
business analyst	1 053	responsable commercial	732
store manager	918	acheteur	706
truck drive	729	ingénieur études et développement	570
sales manager	665	responsable comptable	479
assistant manager	621	responsable ressource humaines	422
restaurateur	565	chef produit	407
financial analyst	497	architecte	380
business developer	496	responsable production	373
account executive	454	infirmier	340

Tableau 2. *Liste des métiers dans chacune des langues, classés par ordre décroissant du nombre d'offres d'emploi. Seuls les 15 premiers sont présentés ici, mais les résultats génèrent 128 métiers en anglais et 312 en français.*

Financial analyst	
Soft skills	Hard skills
financial, business, support, management, process, reports, data, project, ~~including~~, projects ...	accounting, analysis, finance, reporting, cpa, cma, budget, cga, ~~end~~, forecast ...
analytical skills, communication skills, problem solving, ability work, internal external, ~~real estate~~, decision making, interpersonal skills, financial services, verbal written ...	~~financial analyst~~, financial analysis, financial reporting, financial statements, variance analysis, finance accounting, accounting finance, balance sheet, journal entries, financial modelling ...
analytical problem solving, problem solving skills, verbal written communication, ability work independently, key performance indicators, fast paced environment, oral written communication, time management skills, communication interpersonal skills, interpersonal communication skills ...	financial planning analysis, ad hoc reporting, financial reporting analysis, financial analysis reporting, financial statement preparation, year end close, consolidated financial statements, planning budgeting forecasting, business case analysis, possess strong analytical ...

Analyste financier	
Soft skills	Hard skills
comptabilite, analyse, connaissance, information, gestion, direction, ~~recherche~~, environnement, organisation, experience	cpa, finance, finances, consolidation, qualifications, cma, cga,~~principal~~, ~~ecrit~~, bilinguisme ...
etats financiers, experience client, esprit analyse, capacite travailler, analyse synthese, travail equipe, relations interpersonnelles, administration affaires, resolution problemes, gestion priorites ...	analyse financiere, analyses financieres, amelioration processus, processus budgetaire, modelisation financiere, processus affaires, financial reporting, financial analyst, cycle comptable, prix revient ...
cycle comptable complet, capacite travailler pression, esprit analyse synthese, capacite analyse synthese, facilite travailler equipe, capacite travailler equipe, ~~word power point~~, problem solving skills, communication orale ecrite, strong analytical skills ...	consolidated financial statements, analysis problem solving, ~~strong business acumen~~, excellent organizational skills

Tableau 3. *Liste de compétences classées par unigrammes, bigrammes, trigrammes pour les métiers* financial analyst *et* analyste financier *obtenue à partir de respectivement 497 et 127 offres. Les n-grammes considérés comme non pertinents ont été barrés. Les compétences sont triées en ordre décroissant de score.*

Programmer analyst	
Soft skills	Hard skills
~~development~~, technical, systems, software, application, business, design, support,~~data~~, solutions	~~programmer~~, ~~analyst~~, programming, applications, sql, java, test, web, developing, integration
computer science, software development, information technology, business requirements, working knowledge, problem solving, internal external, best practices, ~~experience working~~, communication skills	application development, sql server, design development, web services, unit testing, asp net, java developer, web applications, production support, vb net
problem solving skills, analytical problem solving, ability work independently, verbal written communication, written verbal communication, subject matter expert, written oral communication, build strong relationships, verbal communication skills, communication interpersonal skills	object oriented programming, service oriented architecture, team foundation server, user acceptance testing, ms sql server, asp net mvc, sql server reporting, sql stored procedures, visual studio 2010, software development methodologies,

Analyste programmeur	
Soft skills	Hard skills
developpement, applications, informatique, connaissance, solutions, analyse, web, environnement, ~~recherche~~, experience ...	net, sql, programmation, javascript, server, java, cgi, agile, oracle, langage ...
~~sql server~~, ~~bases donnees~~, esprit equipe, ~~mise place~~, resolution problemes, ~~base donnees~~, projets developpement, capacite analyse, ~~capacite travailler~~, developpement logiciel ...	asp net, vb net, html css, visual studio, ~~intelligence affaires~~, services web, applications web, apache tomcat, oracle sql, ms sql ...
~~ms visual studio~~, esprit analyse synthese,~~ms sql server~~, capacite travailler equipe, team foundation server, ~~sql server 2008~~, ~~asp net mve~~, capacite travailler pression, ~~visual studio 2010~~ ...	master data management, object oriented programming, high pressure environment, visual basic net, visual studio team

Tableau 4. *Liste de compétences sous forme de n-grammes classées par unigrammes, bigrammes, trigrammes pour les métiers* programmer analyst *et* analyste programmeur *obtenue à partir de respectivement 185 et 169 offres. Les n-grammes considérés comme non pertinents ont été barrés. Les compétences sont triées en ordre décroissant de score.*

n-grammes	Anglais						Français					
	Unigramme		Bigramme		Trigramme		Unigramme		Bigramme		Trigramme	
	soft	hard	soft	hard	soft	hard	soft	hard	soft	hard	soft	hard
Total	60	60	60	60	57	60	60	60	60	60	42	9
Pertinents	48	50	53	58	57	59	57	52	52	56	33	8
Précision	0,80	0,83	0,88	0,97	1,0	0,98	0,95	0,87	0,87	0,93	0,79	0,88

Tableau 5. *Synthèse de l'évaluation des résultats.* soft *et* hard *représentent respectivement les compétences transversales et les compétences métier.*

Nous avons barré les n-grammes qui ont été considérés comme non pertinents par l'expert. Le tableau 5 présente une synthèse de l'évaluation de ces compétences en termes de *précision*. Ne disposant pas d'une liste complète des compétences pour chaque métier, nous n'avons pas été en mesure de calculer le *rappel*.

Même si l'échantillon évalué est de taille relativement petite (656 compétences sur environ 60 000 compétences générées), les résultats obtenus sont de très bonne qualité (0,89 sur l'ensemble de l'évaluation). La qualité des listes en anglais (0,91) est légèrement meilleure que celles en français (0,87). Nous attribuons cette différence au plus petit nombre d'offres d'emploi pour les métiers considérés en langue française (497 et 185 offres en anglais contre 127 et 169 offres en français). Ce nombre restreint d'offres d'emploi en français explique aussi le faible nombre de trigrammes obtenus pour le métier d'*analyste programmeur*.

L'analyse détaillée montre par ailleurs une présence importante de termes anglais présents dans les résultats français. Nous attribuons ce mélange à une utilisation fréquente de termes anglais dans les offres d'emploi en français. Celles-ci contiennent des ambiguïtés pour certaines compétences (telles que *leadership*, *management* ou encore *marketing*) ainsi que pour certaines fonctions (*manager*, *trader*, *designer*, etc.), ce qui complique la tâche de détection de langue. L'utilisation des n-grammes de mots occasionne par ailleurs des erreurs entre certaines compétences suivant la taille du n-gramme considéré (*access* et *access to a vehicule*, *office* et *office management*, etc.). Nous constatons aussi une redondance de certaines compétences en fonction de l'usage du singulier ou du pluriel (par exemple : *analyse financière* et *analyses financières*, *bases données* et *base données*) ou de versions différentes d'un logiciel (par exemple : *visual studio 2005/2010* ou *AutoCad 2015/2017*).

Le tableau 5 montre des résultats de meilleure qualité pour les bigrammes et les trigrammes que pour les unigrammes (des précisions moyennes de 0,86 pour les unigrammes, 0,91 pour les bigrammes et 0,93 pour les trigrammes). Nous travaillons à leur amélioration, car l'observation des offres d'emploi et des résultats montre que les technologies requises sont généralement présentées sous forme d'unigrammes (par exemple *cga*, *sap*, *Java*, *MySQL*, etc.). La mise en place du dictionnaire dynamique (section 5.5) a permis d'améliorer globalement la qualité des n-grammes obtenus, cependant des termes ou expressions courants dans les offres d'emploi et dans le dictionnaire viennent parfois bruiter les résultats (par exemple *assurance*). À partir des données du tableau 5, on peut calculer que la précision moyenne des compétences tech-

niques (*hard skills*) 0,92 est légèrement meilleure que celle des compétences transversales (*soft skills*) qui est de 0,88. Afin d'analyser cette différence, nous avons effectué une évaluation de la qualité du dictionnaire généré pour les compétences transversales et les résultats montrent qu'une partie du vocabulaire contenu dans le dictionnaire est non pertinente (précision de 0,87 sur 250 compétences), particulièrement les unigrammes (0,60), ce qui se répercute sur la qualité des résultats finaux.

7. Conclusion et travaux futurs

Nous avons présenté dans cet article les travaux réalisés sur la génération automatique de ressources linguistiques pour les besoins de l'e-recrutement. À partir de dix millions de profils issus de plusieurs réseaux sociaux, ainsi que 45 000 offres d'emploi sur les 300 000 récoltées sur Internet, nous avons fait émerger des compétences et des connaissances communes pour construire un ensemble structuré des termes et concepts représentant chaque métier avec ses compétences associées. Ces offres contenant le profil minimal recherché pour un métier ont été utilisées afin de détecter les compétences. Ces offres d'emploi étant collectées sur Internet, le modèle pourra ainsi s'enrichir de façon semi-automatique avec l'apparition et la disparition de métiers. Nous avons présenté par la suite les premiers résultats obtenus ainsi qu'une évaluation manuelle réalisée par un expert du domaine. L'analyse détaillée a montré des résultats de très bonne qualité (précision moyenne de 0,89) particulièrement en anglais. Nous attribuons cette différence principalement à la quantité plus faible d'offres d'emploi pour les métiers considérés en langue française.

Nous envisageons des traitements plus fins afin de regrouper les compétences fortement similaires ou qui ne diffèrent que par de faibles variations d'écriture. Nous souhaitons utiliser cette ontologie afin d'évaluer si un candidat dispose de toutes les compétences pour un métier ou encore au travers d'un outil de génération de texte afin de suggérer à un candidat comment mettre en avant ses compétences ou encore suggérer des mots-clés pour les recruteurs qui effectuent des recherches dans des bases de profils de candidats. Nous prévoyons par ailleurs de continuer à augmenter la taille de l'ensemble des offres d'emploi afin de couvrir un nombre de métiers plus important pour améliorer la qualité des résultats. Le processus complet étant automatisé, il est possible, et nous l'avons d'ailleurs expérimenté à quelques reprises, de créer une nouvelle ontologie aussitôt que de nouvelles offres d'emploi deviennent disponibles. Il serait toutefois intéressant d'effectuer une mise à jour de l'ontologie existante.

Par ailleurs, nous envisageons l'utilisation de notre système en complément d'un processus classique de sélection des candidats *actifs*, c'est-à-dire ayant posé leur candidature, nous pourrions ainsi identifier les candidats *passifs*, qui ne sont pas en recherche d'emploi, mais qui pourraient être intéressés par de nouvelles opportunités, en parcourant les différents médias sociaux afin de récolter les profils les plus en adéquation avec une offre d'emploi. Une fois cette collecte terminée, une proposition d'opportunité serait transmise à ces candidats *passifs* qui pourraient alors décider si la proposition les intéresse suffisamment pour postuler.

Remerciements

Les auteurs tiennent à remercier le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), qui a financé ces travaux dans le cadre d'une subvention de recherche et développement coopérative ainsi que les membres de l'équipe du projet BPP, en particulier Fabrizio Gotti pour ses contributions sans oublier Éric Tondo et Ludovic Bourg pour leur connaissance du domaine et les discussions fructueuses. L'article a été finalisé dans le cadre d'un séjour de Guy Lapalme à l'IMERA (université Aix-Marseille).

8. Bibliographie

Bendaoud R., Rouane Hacene A. M., Toussaint Y., Delecroix B., Napoli A., « Construction d'une ontologie à partir d'un corpus de textes avec l'ACF », *IC 2007*, Grenoble, France, 2007.

Colucci S., Di Noia T., Di Sciascio E., Donini F. M., Mongiello M., Mottola M., « A formal approach to ontology-based semantic match of skills descriptions », *J. UCS*, vol. 9, nº 12, p. 1437-1454, 2003.

Colucci S., Di Noia T., Di Sciascio E., Donini F. M., Ragone A., Trizio M., « A Semantic-based Search Engine for Professional Knowledge », *Proc. 7th Int. Conf. on Knowledge Management and Knowledge Technologies (I-KNOW 2007),(Sep 2007)*, p. 472-475, 2007.

Colucci S., Tinelli E., Giannini S., Di Sciascio E., Donini F. M., « Knowledge Compilation for Core Competence Extraction in Organizations », *Business Information Systems*, Springer, p. 163-174, 2013.

Desmontils E., Jacquin C., Morin E., « Indexation sémantique de documents sur le Web : application aux ressources humaines », *Journées de l'AS-CNRS Web Sémantique*, 2002.

Gómez-Pérez A., Ramírez J., Villazón-Terrazas B., « An ontology for modelling human resources management based on standards », *Knowledge-Based Intelligent Information and Engineering Systems*, Springer, p. 534-541, 2007.

Hoffart J., Suchanek F. M., Berberich K., Lewis-Kelham E., de Melo G., Weikum G., « YAGO2 : Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages », *Proceedings of the 20th International Conference Companion on World Wide Web*, p. 229-232, 2011.

Kessler R., Béchet N., Roche M., El-Bèze M., Torres-Moreno J. M., « Automatic profiling system for ranking candidates answers in human resources », *On the Move to Meaningful Internet Systems : OTM 2008 Workshops*, Springer, p. 625-634, 2008.

Kessler R., Lapalme G., Tondo É., « Génération d'une ontologie dans le domaine des ressources humaines », *CORIA 2016*, Toulouse, 03/2016, 2016.

Kmail A. B., Maree M., Belkhatir M., Alhashmi S. M., « An Automatic Online Recruitment System Based on Exploiting Multiple Semantic Resources and Concept-Relatedness Measures », *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*, p. 620-627, 2015.

Lau T., Sure Y., « Introducing ontology-based skills management at a large insurance company », *Proceedings of the Modellierung 2002*p. 123-134, 2002.

le Vrang M., Papantoniou A., Pauwels E., Fannes P., Vandensteen D., De Smedt J., « ESCO : Boosting Job Matching in Europe with Semantic Interoperability », *Computer*, vol. 47, nº 10, p. 57-64, Oct, 2014.

Levenshtein V. I., « Binary codes capable of correcting deletions, insertions and reversals. », *Soviet Physics Doklady*, vol. 10, nº 8, p. 707-710, 1966.

Loth R., Battistelli D., Chaumartin F.-R., De Mazancourt H., Minel J.-L., Vinckx A., « Linguistic information extraction for job ads (SIRE project) », *Adaptivity, Personalization and Fusion of Heterogeneous Information*, p. 222-224, 2010.

Miles A., Bechhofer S., « SKOS Simple Knowledge Organization System Reference », 2009.

Mochol M., Simperl E. P. B., « Practical guidelines for building semantic erecruitment applications », *International Conference on Knowledge Management, Special Track : Advanced Semantic Technologies (AST'06)*, Citeseer, 2006.

Osborne J. D., Flatow J., Holko M., Lin S. M., Kibbe W. A., Zhu L. J., Danila M. I., Feng G., Chisholm R. L., « Annotating the human genome with Disease Ontology », *BMC Genomics*, 2009.

Roche M., Kodratoff Y., « Pruning terminology extracted from a specialized corpus for CV ontology acquisition », *On the Move to Meaningful Internet Systems 2006 : OTM 2006 Workshops*, Springer, p. 1107-1116, 2006.

Sami G., Béchet N., Berio G., « Ontologies from Textual Resources : A Pattern Based Improvement Using Deep Linguistic Information », *Workshop on on Ontology and Semantic Web Patterns (WOP)*, vol. 1302 of *Proceedings of Workshop on on Ontology and Semantic Web Patterns (WOP)*, p. 14-25, 2014.

Sivabalan L., Yazdanifard R., Ismail N. H., « How to Transform the Traditional Way of Recruitment into Online System », *International Business Research*, vol. 7, p. 178, 2014.

Trichet F., Bourse M., Leclerc M., Morin E., *Human Resource Management and Semantic Web Technologies*, springer edn, ICTTA, Berlin, 2004.

Trog D., Christiaens S., Zhao G., de Laaf J., « Toward a Community Vision Driven Topical Ontology in Human Resource Management », *On the Move to Meaningful Internet Systems : OTM 2008 Workshops*, Springer, p. 615-624, 2008.

Tétreault M., Dufresne A., Gagnon M., « Development of an Ontology-Based E-Recruitment Application that Integrates Social Web », *Electronic business interoperability : Concepts, opportunities and challenges*p. 363-395, 2011.

Yahiaoui L., Boufaïda Z., Prié Y., « Automatisation du e-recrutement dans le cadre du web sémantique », *Journée francophones d'Ingénierie des Connaissances, IC'2006*, 2006.

Zimmermann T., Kotschenreuther L., Schmidt K., « Data-driven HR - Résumé Analysis Based on Natural Language Processing and Machine Learning », *CoRR*, 2016.

Noise or music? Investigating the usefulness of normalisation for robust sentiment analysis on social media data

Cynthia Van Hee — Marjan Van de Kauter — Orphée De Clercq — Els Lefever — Bart Desmet — Véronique Hoste

Language and Translation Technology Team, Dep. of Translation, Interpreting and Communication, Ghent University. Groot-Brittanniëlaan 45, 9000 Ghent, Belgium

ABSTRACT. *In the past decade, sentiment analysis research has thrived, especially on social media. While this data genre is suitable to extract opinions and sentiment, it is known to be noisy. Complex normalisation methods have been developed to transform noisy text into its standard form, but their effect on tasks like sentiment analysis remains underinvestigated. Sentiment analysis approaches mostly include spell checking or rule-based normalisation as preprocessing and rarely investigate its impact on the task performance. We present an optimised sentiment classifier and investigate to what extent its performance can be enhanced by integrating SMT-based normalisation as preprocessing. Experiments on a test set comprising a variety of user-generated content genres revealed that normalisation improves sentiment classification performance on tweets and blog posts, showing the model's ability to generalise to other data genres.*

RÉSUMÉ. *Ces dernières années ont été marquées par un intérêt croissant pour l'analyse des sentiments sur les réseaux sociaux. Bien que ces derniers soient une source de données utile, ils sont connus pour être bruité. Diverses méthodes de normalisation complexe existent pour transformer du texte bruité en sa forme standard. Cependant, la plupart des études à l'analyse des sentiments incluent la normalisation basée sur des règles et ne recherchent guère sa valeur ajoutée à la classification. Nous présentons un système pour l'analyse des sentiments optimisé et examinons si sa performance s'améliore si la normalisation basée sur la traduction automatique statistique est intégrée. Des expériences avec un corpus de test varié démontrent une meilleure performance si la normalisation est incluse, non seulement pour le genre Twitter, mais aussi pour le genre blog, ce qui démontre la capacité de généralisation du modèle.*

KEYWORDS: *sentiment analysis, social media, optimisation, normalisation.*

MOTS-CLÉS : *analyse de sentiments, réseaux sociaux, optimisation, normalisation.*

TAL. Volume 58 – n°1/2017, pages 63 à 87

1. Introduction

Emotions and sentiment play a key role in successful communication and relationships, and the opinions and beliefs of others play a large part in how we evaluate the world or make decisions. The study of sentiment and its related concepts such as opinions and attitudes is known as automatic *sentiment analysis* or *opinion mining*. With the growing amount of opinionated data on the web, sentiment analysis applications have found their way into various research fields and companies showing interest in understanding how consumers evaluate their goods and services. Although the first studies on automatic sentiment analysis appeared in the early 2000's when researchers focused on newswire text (Wiebe, 2000; Pang *et al.*, 2002), the thriving interest in the field has come along with the birth of social networking sites like Facebook and Twitter. Social media generate a substantial amount of opinionated data, also referred to as **user-generated content** (Moens *et al.*, 2014), that offers valuable insights into the public opinion online.

The accessibility to an ever-growing stream of opinionated data represents one side of the coin, the introduction of what we could call "noise" the other. Given the speed at which social media data are produced, their informal nature and the restrictions with respect to message length, these data are characterised by a rather informal language containing misspellings, grammatical errors, emoticons, abbreviations, slang, etc. (Barbosa and Feng, 2010). Since many natural language processing (NLP) tools have been developed for and trained on standard language, a severe drop in performance is observed when applying these tools to user-generated content (Eisenstein, 2013). Moreover, due to an inconsistent writing style, user-generated content is prone to increase feature sparseness, which complicates the classification task.

In this paper, we present a sentiment analysis pipeline for social media data exploiting a wide variety of information sources. We proceed to **joint optimisation** of the classifier by simultaneously performing feature selection and hyperparameter configuration. In a next step, we investigate to what extent the performance of our optimal sentiment classifier can be further enhanced by applying complex statistical machine translation (SMT)-based **normalisation** to the data prior to training. Although data noisiness has been widely acknowledged as one of the main challenges in the field (Liu, 2015), this is, to our knowledge, the first study to provide a qualitative and comparative analysis of a complex normalisation system and to experimentally test its benefits for an optimised sentiment classifier.

The remainder of this paper is structured as follows. Section 2 presents an overview of sentiment analysis research and text normalisation, while section 3 zooms in on the normalisation system used for this research. Section 4 describes our sentiment analysis architecture and details the experimental setup, after which the results are thoroughly discussed in section 5. Finally, section 6 highlights the conclusions and presents some directions for future research.

2. Related research

Traditional communication tools have increasingly given way to social media, which have become a rich source of opinionated data. To be able to analyse such an amount of data, automatic text processing techniques like sentiment analysis have become highly relevant.

2.1. *Sentiment analysis*

Natural language processing has many applications related to text analysis and initial text classification studies focused on extracting factual information from documents, which resulted in systems for automatic information retrieval (Salton, 1989), text summarisation (Nenkova, 2005), document classification (Finn *et al.*, 2002), topic modelling (Deerwester *et al.*, 1990) and so on. More recently, the research focus of the NLP community is increasingly geared towards the extraction of subjective information from text, thus introducing the field of *sentiment analysis* or the automatic classification of a document as positive, negative or neutral (i.e. when a positive and negative sentiment is expressed, or neither of them).

In its early days, sentiment analysis was mostly applied to newswire documents and movie reviews, seminal work on which has been done by Wiebe *et al.* (2004) and Pang and Lee (2008). The interest in the domain has thrived with the expansion of the Internet and the birth of social media, providing a large amount of opinionated data that is increasingly searched for by researchers, companies, politicians, and trend-watchers. As a result, sentiment analysis has become a major research area in NLP, which is reflected in survey articles by Liu (2015) and Mohammad (2016).

Two dominant approaches to sentiment analysis exist: (i) a **machine learning-based** approach and (ii) a **lexicon-based** approach. Machine-learning approaches are either supervised or unsupervised. Supervised methods include classifiers that are trained on labelled or annotated documents, whereas unsupervised methods infer the semantic orientation of a document without labelled data (e.g. using clustering or seed words). Semi-supervised approaches are a combination of both, for instance when sentiment lexicons are used, but no manually labelled data. Lexicon-based approaches make use of dictionaries of sentiment words to calculate a global sentiment score for a document. Both lexicon-based and machine-learning approaches present difficulties: while the former suffer from the varying and changing nature of language, machine-learning approaches generally require a large corpus of labelled documents to train a model. The two methods have been investigated extensively over the past years and this has resulted in a fair amount of benchmark data in the field. In machine learning, researchers have investigated and compared the performance of different classification and regression algorithms (Davidov *et al.*, 2010; Agarwal *et al.*, 2011), and more recently successful sentiment analysis approaches involve deep learning using neural networks (Severyn and Moschitti, 2015).

Sentiment analysis has also increasingly attracted researchers' interest in the framework of specialised shared tasks like SemEval, a series of evaluations of computational systems for tasks related to semantic text analysis. Datasets for training and testing are provided by the task organisers so that the participants' systems can be compared against each other. SemEval-2013 to -2017 workshops included a Twitter sentiment analysis task, which has resulted in state-of-the-art sentiment classifiers and a number of new resources, such as the NRC Hashtag Sentiment Lexicon (Mohammad *et al.*, 2013). Over the years, the shared task has attracted approximately 200 research teams, submitting together more than 300 sentiment analysis systems. Popular classification methods are Support Vector Machines (SVMs), Naïve Bayes, Conditional Random Fields (CRFs), and deep learning architectures. Often exploited features include bags-of-words (i.e. word or character n-grams), syntactic features based on part-of-speech (PoS) information or dependency relations, semantic features capturing modality and negation, sentiment lexicon features and user-based features. Twitter-specific features are often included to represent creative writing (e.g. punctuation, capitalisation, character flooding, emoticons and at-replies) (Nakov *et al.*, 2016). In short, recent research has successfully explored sentiment classification on Twitter, presenting classifiers with performance scores of up to F_1= 0.69 (Rosenthal *et al.*, 2017). Nevertheless, it has also unveiled a number of bottlenecks, one of them being the use of highly informal language or **data noisiness**.

2.2. *Handling "data noisiness"*

One of the larger and less-researched issues NLP is currently facing is *data noise* (Liu, 2015), which refers to non-standardised language variation. In fact, an inherent characteristic of social media data is its tendency to deviate from the linguistic norm, as it often contains misspellings, creative punctuation and inconsistent capitalisation, or has a poor grammatical structure. This hinders automatic text processing with traditional NLP tools that are trained on standard text and show a significant drop in performance when applied to social media data (Eisenstein, 2013). Among others, Ritter *et al.* (2011) demonstrated that the performance of existing tools for PoS tagging and chunking decreases when applied to tweets, and similar findings were reported by Liu *et al.* (2011) for named entity recognition.

As described by Han *et al.* (2013), one way to minimise the performance drop of such tools is to adapt or **retrain** them using social media data. Recent studies have introduced Twitter-specific tools for, among other tasks, PoS tagging (Gimpel *et al.*, 2011), and named entity recognition (Ritter *et al.*, 2011). A drawback of this strategy is that it implies a process of continuous retraining of each individual tool to make it robust to changes in language use. According to the researchers, another strategy to handle non-standard language is **text normalisation**, which transforms noisy or non-standard text into its standard form. Three dominant approaches to text normalisation exist, referred to as the spell-checking, the machine translation and the speech recognition metaphors (Kobus *et al.*, 2008). The former metaphor refers to normal-

isation as a word-per-word correction primarily targeting out-of-vocabulary (OOV) words. The translation metaphor refers to normalisation as a machine translation task where the noisy text has to be "translated" into its standard form. Finally, the speech recognition metaphor alludes to the analogy between noisy and spoken language, assuming that noisy content like SMS messages tends to be closer to oral expressions than to written text. For a comprehensive overview of the different normalisation approaches, we refer to Kobus *et al.* (2008).

Although various normalisation systems have been developed and have proven successful (De Clercq *et al.*, 2013; Pettersson *et al.*, 2013; Schneider *et al.*, 2017), most approaches to sentiment analysis only incorporate shallow normalisation such as spell-checking mechanisms or rule-based normalisation without evaluating the impact of normalisation on the classification performance (Roy *et al.*, 2011; Haddi *et al.*, 2013; Sharma *et al.*, 2015). In the present study, we therefore investigate the impact of an SMT-based normalisation system as preprocessing for our optimised sentiment classifier.

3. Normalisation of user-generated content

As discussed in the previous sections, data noisiness may severely undermine the accuracy of NLP tasks including tokenisation, lemmatisation and PoS tagging. Moreover, it may hinder the effectivity of bag-of-words or lexicon-based features due to increased sparseness. The negative effects of data noise are well-known, but only few studies have applied normalisation as a preprocessing step for text mining tasks. To our knowledge, the actual impact of complex normalisation on sentiment classification performance has not been investigated sufficiently. Mosquera and Moreda (2013) did tackle the problem and showed a 6% polarity classification improvement after applying normalisation to tweets, but, in contrast to the current approach, they implemented a normalisation system based on dictionaries, rather than a statistical normaliser trained on user-generated data.

We hypothesise that normalisation helps to reduce feature sparseness, as was shown by Desmet (2014), and consequently improves the coverage of sentiment lexicons that are often used to construct features for this task. To this purpose, we applied SMT-based normalisation (De Clercq *et al.*, 2013), the underlying idea of which is to perceive normalisation as a translation process from noisy text into normalised standard text in the same language. An SMT system basically consists of two subsequent models: a **translation model** that is trained to find the right target words given the source words and a **language model** that ensures the translated words come in the right order. Applying SMT to text normalisation can be done at different levels of granularity (i.e. at the token or character level). The advantage of working at the token level is that the high-frequency words and abbreviations can be translated in context, which outperforms a dictionary look-up (Raghunathan and Krawczyk, 2009). A character-based translation module can translate non-standard words that were not

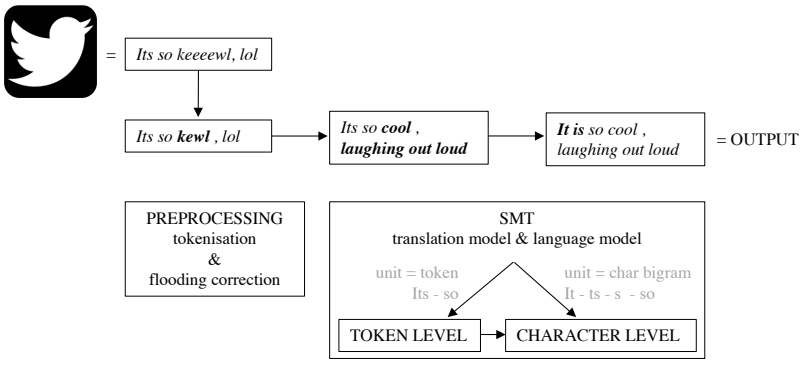


Figure 1. *Illustration of the normalisation process. After preprocessing the tweet, it is normalised at the token level, followed by a normalisation at the character level.*

seen in the training set as it learns patterns of character sequences rather than entire words, which makes the system more robust (Li and Liu, 2012).

For the current research, we propose an SMT-based approach and make use of a system developed by De Clercq *et al.* (2013) for text normalisation of (initially Dutch) user-generated content. The system is based on statistical machine translation and makes use of the SMT system Moses (Koehn *et al.*, 2007). As a target corpus for the language model, which was built with KenLM (Heafield, 2011), parts of the English Open Subtitles Corpus [1] were used, since the data in this corpus are user-generated. The training data for the normalisation system comprises texts from the social networks site ASKfm [2], YouTube comments (Dadvar *et al.*, 2014) and a subset of the Twitter corpus (TWE) compiled by Xue *et al.* (2011). The final corpus comprises about 60,000 tokens and has been manually normalised using annotation guidelines (De Clercq *et al.*, 2014). The system performance was evaluated on three different types of user-generated content, being message board posts (SNS), text messages (SMS), and tweets (TWE). With F_1 scores of up to 0.65 for all three genres, the system has demonstrated its robustness across different genres of user-generated content.

The system, the process of which is depicted in Figure 1, works as follows: prior to performing SMT, the noisy text is tokenised using the Twitter tokeniser by Gimpel *et al.* (2011). Subsequently, a cascaded approach is taken: the system first processes a noisy tweet at the token level to find frequent abbreviations (such as "lol" for "laughing out loud"), after which the output is split into character sequences for normalisation at the character level. The latter step should allow to better resolve character transpo-

1. http://www.opensubtitles.org.
2. http://ask.fm.

sitions, but also problems across the token level, like fusions such as “its” for “it is”. As mentioned earlier, character-based translation models also better generalise since they can learn productive alternations and correct them in words that do not occur in the training data. An important decision to make when applying such a cascaded approach is the level of granularity at the character level, for which we tested unigrams and bigrams.

	original	**normalised**
SMS	Ok then later i **msg u** where i am.	Ok then later i **message you** where i am.
	yah, **wed** i think. But i **noe she not** free on **wed**.	**yah**, **wed** i think. But i noe not free on **wed**.
Twitter	@Manlyboyze I would put Gillard **2nd** best behind Hawke, Rudd **3rd**, Keating 4th, then all the rest, then Howard last #Showdown	@Manlyboyze I would put Gillard **second** best behind Hawke, Rudd **third**, Keating 4th, then all the rest, then Howard last #Showdown

Table 1. *Corpus examples before and after normalisation.*

We empirically verified which cascaded approach (i.e. token+character unigrams or token+character bigrams) works best for the research presented here by selecting 100 random instances from the English SemEval Twitter training corpus (Rosenthal *et al.*, 2014). These 100 tweets (comprising 2,185 tokens all together) were manually normalised by two trained linguists who performed the task independently. In a next phase, a few conflicting normalisations were discussed and resolved, which resulted in a total of 206 tokens that were normalised. Next, the dataset was automatically normalised using two different flavours of the normalisation system and we thus compared the output of the cascaded token+character unigram approach (system A) with that of the cascaded token+character bigram approach (system B). We found that the systems perform comparably in terms of recall, given that system A resolved 41% of the normalisations, whereas system B resolved 40%. When looking at the precision of both systems, however, we found that system A hypernormalised more tokens, which often led to strange additions. As such, precision of the system was 61%, as opposed to 86% of system B. We therefore decided to use system B or the cascaded token+character bigram module for the normalisations performed in the framework of this paper. This finding is in line with previous experiments on Dutch data which revealed that the best performance is indeed achieved with a cascaded bigram approach (De Clercq *et al.*, 2013).

4. Sentiment analysis architecture

This section presents the experimental corpus and feature engineering process, after which the feature filtering and normalisation experiments are described.

4.1. *Data preprocessing and feature engineering*

For the experiments we made use of a training and a held-out test corpus, both containing English social media data. The corpora were distributed in the framework of the SemEval-2014 shared task on sentiment analysis (Rosenthal *et al.*, 2014) and consist of 11,338 and 8,987 instances, respectively (Table 2).

train corpus	**test corpus**		
Twitter	Twitter	SMS	blog
11,338	5,752	2,093	1,142

Table 2. *Number of instances in the experimental corpus.*

The training corpus contains merely tweets, whereas the test corpus comprises a variety of user-generated content, including tweets (regular and sarcastic), text messages (SMS), and LiveJournal [3] blog posts. All instances in the corpora are assigned one out of three class labels, being *positive*, *negative*, and *neutral*, which represent 37%, 16% and 47% of the training data. The class distributions in the test set are comparable.

Prior to extracting features, all tweets in the *norm* and *not norm* corpus were tokenised and PoS tagged using the Carnegie Mellon University Twitter NLP Tool (Gimpel *et al.*, 2011). The tag list comprises regular PoS tags (e.g. "N" for noun, "V" for verb), as well as Twitter-specific ones (e.g. "#" for hashtags, "E" for emoticons). The following paragraphs describe the different features that were extracted to provide the model with relevant information for the task.

Bag-of-words features (BoW): features that represent each message as a "bag" of its words or characters. Unigrams, bigrams and trigrams were extracted at the token level and trigrams and fourgrams at the character level (without crossing token boundaries). N-grams that occurred only once in the training corpus were discarded, which decreased the total number of n-gram features by 80% and the entire feature space by about 50%.

Post length: numeric feature indicating the tweet length in tokens.

Word-shape features: numeric and binary features including (i) the number of tokens with character (e.g. *yaaaay*) and (ii) punctuation flooding (e.g. *??!!*), (iii) whether the last token of the tweet contains punctuation, (iv) the number of tokens in uppercase and (v) the number of hashtags in the tweet. All the numeric features were normalised by dividing them by the tweet length.

3. http://www.livejournal.com.

Syntactic features:

– **Part-of-Speech (PoS) features:** four features for each one of the 25 tags in the PoS tagset, indicating (i) whether the tag occurs in the tweet, (ii) whether the tag occurs zero, one, or two or more times, (iii) the absolute and (iv) relative frequency of the tag. It is important to note that the Twitter-specific PoS tagger distinguishes hashtags that are part of the tweet content from those with a "hashtag function". For instance, in "Foreign commentator award at the #commentawards goes to Marie Colvin", *#commentawards* is tagged as a noun, while in "I hope I get to meet Carmelo Anthony at the Knicks Rally #Hopeful", *#Hopeful* is tagged as a hashtag.

– **Dependency relation features:** four binary features for every dependency relation found in the training data (example 1). The first feature indicates the presence of the lexicalised dependency relations in the test data (hm-lex). For the remaining features, the dependency relation features are generalised in three ways, as proposed by Joshi and Penstein-Rosé (2009): by backing off the head word to its PoS tag (h-bo), the modifier to its PoS tag (m-bo), and both the head and modifier (hm-bo). The dependency parser we made use of is not adapted to Twitter data, hence hashtags are treated like other words.

(1) I had such a **great time** tonight that I've decided to keep celebrating! → hm-lex: (*time*, *great*), h-bo: (N, *great*), m-bo: (*time*, A), hm-bo: (N, A)

Named entity features: four features indicating the presence of named entities in a tweet: one binary feature and three numeric features, indicating (i) the number of NEs in the tweet and (ii) the number and (iii) frequency of tokens that are part of a NE.

PMI features: two numeric features based on PMI (*pointwise mutual information*) obtained from (i) word-sentiment associations in the training data, and (ii) an existing PMI lexicon (Mohammad *et al.*, 2013). A positive PMI value indicates positive sentiment, a negative score indicates negative sentiment. The higher the absolute value, the stronger the word's association with the sentiment. PMI values were calculated by subtracting a word's association score with a negative sentiment from the word's association score with a positive sentiment, as shown by the following equation: $PMI(w) = PMI(w, positive) - PMI(w, negative)$.

Sentiment lexicon features: four sentiment lexicon features were implemented to capture the semantic orientation of a tweet. We consulted existing lexicons including AFINN (Nielsen, 2011), General Inquirer (GI) (Stone *et al.*, 1966), MPQA (Wilson *et al.*, 2005), the NRC Emotion Lexicon (Mohammad and Turney, 2010), Liu's opinion lexicon (Hu and Liu, 2004), Bounce (Kökciyan *et al.*, 2013) and a manually created emoticon list based on the training data. Four features were extracted per lexicon: the number of positive, negative and neutral lexicon words averaged over text length, and the overall polarity (i.e. the sum of the values of the identified sentiment words). These features were extracted by looking at (i) all the tokens in the instance and (ii) hashtag tokens only (e.g. *win* from *#win*). Negation cues were considered by

flipping the polarity of a sentiment word if it occurred within a window of three words left or right to a negation word from a manually composed list (e.g. "neither", "without", "cannot"). To minimise ambiguity, a word was only attributed a sentiment value if its PoS tag matched that of the dictionary entry (when available).

4.2. *Experimental setup*

Our two main research objectives are (i) to build a sentiment classifier that makes optimal use of a varied feature set (cf. section 4.1) and (ii) to investigate the impact of lexical normalisation when included as a preprocessing step (cf. section 4.2.3). For the experiments we made use of a support vector machine as implemented in the LIBSVM library (Chang and Lin, 2011), since the algorithm has been successfully implemented with large feature sets and its performance for similar tasks has been recognised (Zhu *et al.*, 2014). Prior to constructing models, all feature vectors were scaled so that each variable fits in the range [0, 1]. As the evaluation metrics, we report accuracy, and macro-averaged precision, recall and F_1 score. Macro-averaging was preferred over micro-averaging as it attributes equal weight to the different classes in the evaluation (Sokolova and Lapalme, 2009). We recall that these classes are positive, negative and neutral (cf. section 4.1).

4.2.1. *Baselines*

Three standard baseline classifiers were implemented against which to compare the models' performance, being the majority and random class baseline and a unigram-based model (w1gr) relying solely on word unigram features. For this model, the LIBSVM classifier was applied in its default parameter settings. Evaluation was done using ten-fold cross validation.

baseline	**accuracy**	**precision**	**recall**	**F_1**
majority	46.97%	15.66%	33.33%	21.30%
random	38.91%	33.58%	33.58%	33.58%
w1gr	46.97%	15.66%	33.33%	21.30%

Table 3. *Cross-validated results for the majority, random and unigram baseline.*

As can be deduced from the table, scores for the baselines are low. The (unoptimised) unigram baseline scores equally to the majority baseline as it consistently predicts the neutral class, which demonstrates the importance of hyperparameter optimisation for our LIBSVM classifier.

4.2.2. *Building an optimal sentiment classifier exploiting a rich feature set*

It has been shown that sentiment classification benefits from a variety of information sources including bags of words, syntactic features, sentiment lexicon features, negation and modality clues, and so on (Zhu *et al.*, 2014). Implementing such a rich

feature set seems promising, since many individual features may have a substantial predictive power when they are combined. However, it is probable that not all individual features are equally informative for the decision-making algorithm. For instance, bag-of-word features have shown to perform well for sentiment analysis (Rosenthal *et al.*, 2015), but easily suffer from sparseness as the corpus size increases (Saif *et al.*, 2012). High-dimensional data with a large number of features may include irrelevant and redundant information, which could degrade the performance of learning algorithms (Yu and Liu, 2003). hat is why in our first round of experiments, we disentangle which types of information sources described in section 4.1 lead to optimal classification performance by means of feature filtering and hyperparameter optimisation using a genetic algorithm (see further). It is important to note that, in this experimental round, normalisation is not included as a preprocessing step.

Dimensionality reduction of the feature space is done in two phases, including (i) feature filtering based on information gain (Daelemans *et al.*, 2009) and (ii) wrapped feature selection.

(i) **Feature filtering** is done based on information gain (IG). IG is the difference in entropy, i.e. the uncertainty about a class label given a set of features when the feature is present or absent in a feature vector representation. Given the considerable number of individual features, we decided to apply feature filtering prior to joint optimisation using a wrapper method. We experimentally determined a threshold of 0.001, meaning that features with an IG value below 0.001 were discarded. This reduced our initial feature space by approximately 99.5% (i.e. from 430,980 to 1,852 features).

(ii) **Wrapper methods**, as opposed to statistical feature filtering (see (i)), conduct a search for informative features using the induction algorithm itself as part of the evaluation function (Kohavi and John, 1997). The selected features are the ones that have the most predictive power given the particular task and classification algorithm. The advantage of wrapped feature selection is that it tests features in combination to detect the possible interactions between them. An important disadvantage of wrapper methods is that they are computationally expensive for data with numerous features (Chandrashekar and Sahin, 2014). Two measures were taken to prevent the search space from becoming too large. First, the feature space was filtered using information gain, as explained in the previous paragraph. Second, the remaining individual features were combined into feature groups for feature selection, since 1,852 individual features would increase the search space considerably due to combinatorial explosion.

Another challenging but necessary (see the results of the unoptimised unigram model in Table 3) task when applying machine learning algorithms is defining good hyperparameter settings as it allows a classifier to better fit the training data. Out of the variety of hyperparameters that can be set for a LIBSVM classifier, we chose to optimise kernel-specific settings including t (the kernel type), d (the kernel function degree), g (the kernel function gamma), and the classification cost value, C. Hoste (2005), among other researchers, demonstrated the importance of *joint optimisation*, meaning that feature- and hyperparameter selection are performed at the same time

so that their mutual influence can be evaluated. To this purpose, we made use of the Gallop (Genetic Algorithms for Linguistic Learner Optimisation) toolbox (Desmet *et al.*, 2013). Optimisation was done using k-fold cross validation on the training set and the number of individuals to be tested per generation was set to 100. Given the size of the training corpus, k was set to 3. Both the feature groups and hyperparameters were defined so as to maximise macro-averaged F_1 to assign equal weight to each class in the evaluation.

4.2.3. *Normalisation experiments*

We hypothesised earlier that data noise could (i) increase sparseness in the feature space and (ii) lower the coverage of sentiment lexicons. On the positive side, however, character flooding or extensive capitalisation may provide hints for automatic sentiment analysis. To assess the actual impact of normalisation on this task, we compared the performance of our most optimal sentiment classifier when trained on not-normalised versus normalised input. Before presenting the experimental results, we already provide some insights into the qualitative analysis of the input data before and after normalisation.

Related to the first hypothesis, we observed that token n-grams in the normalised corpus were more sparse: 54,326 token n-grams versus 53,458 in the not-normalised corpus. A qualitative analysis revealed that, while the normaliser decreased sparseness by normalising spelling variations like "yesssss" and "yesss" to "yes", it increased sparsity through (i) correct normalisations that resulted in more tokens than the original word (e.g. "btw" $\rightarrow$ "by the way", "gonna" $\rightarrow$ "going to"), but also through strange operations and hypercorrections (e.g. "#Illini"$\rightarrow$ "#I willini"). Also, given that n-grams with a frequency less than two in the training corpus were discarded during preprocessing, it is likely that normalisation increased the number of tokens added to the bag-of-words. For instance, "Jerry's" and "Jerrys" both occur once in the corpus and were discarded from the bag-of-words in the not-normalised corpus, but during normalisation, "Jerrys" was corrected to "Jerry's", and occurring more than once now, the word was added to the bag-of-words. Normalisation did, however, cause a slight decrease in sparsity in the character-level n-grams (–1%) and the dependency features (–0.5%).

Regarding our second hypothesis, we observed that after normalising the input text, the coverage of the sentiment lexicons increased by 0.22%. This would indicate that normalisation has a rather limited effect on sentiment-bearing words, which could be because (i) their noise cannot be resolved by the normaliser or (ii) they barely need correction. The results of a qualitative analysis of the corpus are in favour of the latter assumption, revealing that syntactic words (e.g. "wiv u" $\rightarrow$ "with you", "ima" $\rightarrow$ "I am going to") appeared much more prone to deviant spelling in our corpus than sentiment-bearing words.

5. Results

In section 4.2.2 we described the steps towards our most optimal classifier. Now we discuss its results and take a closer look at the selected features and hyperparameter settings (sections 5.1 and 5.2.1). Furthermore, we investigate the impact of normalisation as preprocessing on the classification performance. To this end, a second optimised classifier was built after the SMT-based normalisation system was applied to the dataset, leading to a more standard text input corpus. We then discuss the results of our optimised classifier before and after normalisation (section 5.2). Finally, this section zooms in on the performance of our optimised classifiers on a held-out test set containing different genres of user-generated content (section 5.2.2).

5.1. *The effect of classification optimisation*

Observing poor results by the unoptimised LIBSVM classifier, and hypothesising that not all individual features contribute equally to the classification performance (cf. section 4.2.2), we investigate the effect of **joint optimisation**. Table 4 compares the results of two classifiers before and after joint optimisation. A unigram-based model is compared to that of a rich-feature-based model to investigate the benefits of a rich feature set for this task.

model features	optimised?	accuracy	F_1	precision	recall
1) unigram BoW	-	46.97%	21.30%	15.66%	33.33%
2) rich feature set	-	46.97%	21.30%	15.66%	33.33%
3) unigram BoW	✓	67.87%	63.00%	66.98%	61.58%
4) rich feature set	✓	**79.20%**	**75.29%**	**77.55%**	**73.94%**

Table 4. *Cross-validated results for the unigram and rich feature-based models, before and after joint optimisation.*

As can be deduced from the table, when applied in its default parameter settings, the classifier consistently predicts the majority class, independently of the features that are used to construct the model (cf. systems 1 and 3). This demonstrates once more (cf. Table 3) that LIBSVM's default kernel type (RBF) is sensitive to a high-dimensional feature space if no suitable hyperparameter settings are defined (Hsu *et al.*, 2003). Joint optimisation clearly pays off, as it increases the performance of the unoptimised classifiers by 42% and 54%. The results further demonstrate that, when optimised, the classifier benefits from a rich feature set, yielding an F_1 score of 75.29%, as opposed to F_1= 63% when only word unigram features are used.

5.2. *The effect of normalisation on sentiment classification*

Having in place an optimised sentiment classifier, we now investigate the impact of integrating SMT-based normalisation as preprocessing on the classification performance. Table 5 presents the three-fold cross-validation results obtained for both optimised models, trained on normalised and not-normalised tweets (hereafter referred to as the "norm" and "not norm" setup). Similarly to the setup without normalisation as preprocessing, optimisation of the classifier involves IG-based feature filtering as well as joint feature selection plus hyperparameter optimisation using Gallop. Feature filtering decreased the feature space by approximately 99.5% (from 400,872 to 2,002 features), which is similar to the feature space reduction we observed for the corpus without normalisation (cf. section 4.2.2).

	overall score				*F_1 score per class*		
	accuracy	**F_1**	**precision**	**recall**	**F_1(pos.)**	**F_1(neg.)**	**F_1(neu.)**
TWE not norm	**79.20%**	**75.29%**	**77.55%**	**73.94%**	**82.78%**	61.22%	**81.87%**
TWE norm	77.77%	74.17%	76.00%	73.02%	80.39%	**61.39%**	80.73%

Table 5. *Cross-validated results of the optimised classifier with and without normalisation as preprocessing.*

As shown in Table 5, with F_1 scores of 75.29% and 74.17%, the optimised *not norm* and *norm* models outperform the baselines and unoptimised models with more than 50%. Surprisingly, the model constructed from the raw (i.e. not normalised) corpus slightly outperforms the *norm* model. As described in section 3, the precision of our normaliser is high (i.e. 86%), hence it is unlikely that its error rate exceeds the amount of noise in the corpus. Results on the held-out test set (see further) could, however, provide more insights into the performance of both models. The scores per sentiment class indicate that both classifiers perform better on the positive and neutral classes compared to the negative class. This could be explained by the lower number of negative class instances in the training data (cf. section 4.1).

In what follows, we evaluate the performance of the *norm* and *not norm* models on the held-out test set comprising a variety of user-generated content genres. Assuming that they each contain a different level of noise, we take a closer look at the classifier performance for each genre. Before discussing the results, we zoom in on the feature groups that were selected during joint optimisation.

5.2.1. *Selected features in the normalised and not-normalised setups*

In this section, we scrutinise which feature groups and hyperparameter settings were chosen after optimisation of the *norm* and *not norm* models by means of the genetic algorithm Gallop (Desmet *et al.*, 2013). We recall that the feature groups were created from the remaining features after filtering the feature space using information gain (cf. section 4.2.2). For the *not norm* model, all features were grouped into 36 different feature groups depending on their nature (e.g. token unigrams, token bigrams,

binary NE features, NRC sentiment lexicon features, and so on). Since all hm-bo dependency features (cf. section 4.1) were discarded after feature filtering using Information Gain [4], only 35 feature groups were retained for the *norm* setup. Hence, respectively 36 and 35 feature groups were considered for wrapped feature selection in the *not norm* and *norm* setup. During the optimisation, different combinations of hyperparmeter settings and feature group activations were evaluated by measuring the classifier performance through a three-fold cross validation experiment. At the end of an optimisation run, the highest fitness score indicated the classifier performance given a particular combination of hyperparameter settings and feature group activations. For both our models, this score is shown in Table 5.

As this score may be shared by other individuals with different feature group activations or hyperparameters (for a detailed explanation, see Desmet (2014)), it is advisable to not only discuss the individual with the highest fitness score, but also a number of individuals that obtained a comparable fitness score. For the analysis of the selected features and hyperparameter settings, we therefore discuss the k-nearest fit solution set as proposed by Desmet (2014), i.e. by rounding the scores to three decimals and selecting the top three fitness scores out of these.

Based on the top fitness scores, we carefully analysed the hyperparameter settings and selected the feature groups that were activated in the seven and six best individuals of the TWE *norm* and *not norm* models, respectively. In section 4.2 we defined which LIBSVM parameters are relevant to this task. The output of the optimisation process revealed that the linear kernel (t=0) was always selected as the most suited kernel type for the *norm* model, whereas the sigmoid kernel (t=3) was selected for the *not norm* model. In effect, when looking at the results of the optimised models, we see a substantial improvement over the results obtained with the classifier in its default kernel configuration (Table 5), demonstrating that defining a suited kernel is instrumental to LIBSVM's good performance with large feature spaces. For the normalised model, the optimal cost value C of the k-nearest individuals varied between 0.25 and 1, where it varied between 256 and 1,024 for *not norm*. The d parameter [5] is not relevant here given that a linear and sigmoid kernel were defined. The γ parameter [6] is not relevant when using a linear kernel, but it was set to an optimal value of 0.0039 for the *not norm* model.

Figure 2 visualises the activation of the different feature groups in the three-nearest fit individuals for both models. Feature groups that are consistently retained in the fittest individuals can be considered the most important given the nature of the classification task. In both figures the right column contains information about the selection status of the feature group, represented as a percentage and with a colour range. The

4. In the same way, for both *norm* and *not norm*, sentiment features based on hashtag tokens were discarded for the Bounce and SemEval (se) lexicons after IG-filtering. This makes intuitive sense as both are emoticon lexicons and do not consider regular words.

5. d is the degree of freedom of the polynomial kernel.

6. Intuitively, the gamma parameter defines how far the influence of a single training example on the model reaches.

Feature group activation NOT NORM

postLength	100%
countFloodedTokens	0%
countCapitalizedTokens	66.67%
countFloodedPunctuationTokens	100%
countHashtags	100%
punctuationLastToken	66.67%
token1gramFeatures	100%
token2gramFeatures	100%
token3gramFeatures	50%
character3gramFeatures	0%
character4gramFeatures	100%
dependency_hm-lex	0%
dependency_h-bo	0%
dependency_m-bo	0%
PoSBinaryFeatures	100%
PoSTernaryFeatures	83.33%
PoSAbsoluteFeatures	100%
PoSFrequencyFeatures	100%
NEBinary	66.67%
NEAbsolute	66.67%
NETokenAbsolute	66.67%
NETokenFrequency	66.67%
nrc	33.33%
gi	33.33%
liu	33.33%
mpqa	100%
afinn	100%
bounce	66.67%
se	33.33%
nrc-hashtokens	100%
gi-hashtokens	66.67%
liu-hashtokens	33.33%
mpqa-hashtokens	100%
afinn-hashtokens	83.33%
nrc-PMIScore	66.67%
se-PMIScore	100%

Feature group activation NORM

postLength	71.43%
countFloodedTokens	71.43%
countCapitalizedTokens	0%
countFloodedPunctuationTokens	57.14%
countHashtags	57.14%
punctuationLastToken	71.43%
token1gramFeatures	71.43%
token2gramFeatures	100%
token3gramFeatures	28.57%
character3gramFeatures	28.57%
character4gramFeatures	100%
dependency_hm-lex	0%
dependency_h-bo	0%
dependency_m-bo	28.57%
dependency_hm-bo	0%
PoSBinaryFeatures	28.57%
PoSTernaryFeatures	85.71%
PoSAbsoluteFeatures	100%
PoSFrequencyFeatures	100%
NEBinary	100%
NEAbsolute	0%
NETokenAbsolute	71.43%
NETokenFrequency	0%
nrc	100%
gi	71.43%
liu	0%
mpqa	100%
afinn	100%
bounce	0%
se	0%
nrc-hashtokens	85.71%
gi-hashtokens	85.71%
liu-hashtokens	85.71%
mpqa-hashtokens	0%
afinn-hashtokens	85.71%
nrc-PMIScore	85.71%
se-PMIScore	100%

Figure 2. *Gallop feature group activation of the models with (*norm*) and without (*not norm*) normalisation as preprocessing.*

higher the percentage of individuals where the feature group is selected, the darker the tone. The figures are split in such a way that feature sets of the same nature or related feature sets are grouped together. The upper part comprises the word-shape features, the post length, and hashtag count. The second part shows the bag-of-word features at the token and character level. The third part contains the dependency relation features,

the PoS tag features and the named entity features. After that, the sentiment lexicon features are presented per lexicon (NRC, General Inquirer (gi), SemEval (se), etc.), for regular and hashtag tokens. At the bottom of the table are the PMI-based features. Overall, we notice that more features groups are retained in the *not norm* setup (31 out of 36) than in the *norm* setup (27 out of 37). If we zoom in on the differences between *norm* and *not norm*, we observe that **social media-specific features**, i.e. features that capture creative writing like punctuation flooding and capitalisation, are more often activated in the *not norm* than in the *norm* model. Interestingly, this is not the case for *countFloodedTokens*, a feature that captures the number of tokens with character flooding, which is removed during normalisation. A qualitative analysis revealed, however, that not all types of flooding were removed during normalisation. As such, important features were derived from words like "yummmm" and "xxxxx" where the flooded characters are at the end of the word instead of in the middle. These words were, as opposed to "yaaay" and "looooooll", not corrected by the normaliser. When looking at the sentiment features, we see that information provided by the MPQA and AFINN lexicons results in good features for both models, since they show a 100% activation across the best individuals. The AFINN lexicon also shows a high activation when applied to hashtag tokens in both models. This is in line with the work by Özdemir and Bergler (2015), revealing that AFINN outperforms other popular lexicons for sentiment analysis tasks.

Overall, the top three most important feature groups (i.e. with the highest overall activation) for the *not norm* model include PMI-based features (i.e. nrc-PMIScore and se-PMIScore), sentiment lexicon features (i.e. nrc-hashtokens to afinn-hashtokens) and word-based features (i.e. postLength to punctuationLastToken). For the *norm* model, the first and second most important feature groups (i.e. PMI and sentiment lexicon features) are the same, but the third most important group are n-gram features (i.e. token1gramFeatures to character4gramFeatures). The analysis further reveals that features based on dependency relations in the training data are the least informative, given that only one out of four dependency features (i.e. dependency m-bo) is activated in *norm*, but no dependency features were activated in the fittest *not norm* models.

5.2.2. *Results on the held-out test set*

In this section, we report the results of our optimised sentiment classifiers (i.e. trained on our corpus with and without normalisation as preprocessing) on a varied held-out test set. The test set has not been part of the corpus during training and optimisation and therefore allows to draw conclusions about the robustness of our system. It was distributed for the SemEval Task 9 (Rosenthal *et al.*, 2014) and consists of 8,987 messages from a variety of genres including Twitter (64%), SMS (23%), and blogs (13%). Table 6 displays the scores obtained by the two models. As evaluation metrics, we report accuracy and (macro-averaged) precision, recall and F_1 score.

We can observe that overall, the best scores are obtained with the *norm* model, reaching an F_1 score of 68.90% on the full test set. This is opposite to our findings in the cross-validation experiments (cf. Table 5) and indicates that the *norm* model

	overall score				*F_1 score per class*		
	accuracy	**F_1**	**precision**	**recall**	**F_1(pos.)**	**F_1(neg.)**	**F_1(neu.)**
TWE not norm	70.26%	68.23%	**73.02%**	67.22%	66.91%	**63.40%**	74.38%
TWE norm	**70.94%**	**68.90%**	72.55%	**67.80%**	**69.22%**	62.95%	**74.54%**

Table 6. *Results for each optimised model on the held-out test set, for all classes and for each class label separately.*

is more robust, allowing to perform better on unseen data, even when it contains different genres of UGC. We observe that both systems perform well on all three classes, although they show an important difference in distribution in the training data (cf. section 4.1). The negative class, for instance, only represents 16% of the training corpus. We can conclude from this that our optimisation strategy to take macro-averaged F_1 score as fitness criterion during optimisation pays off. Judging from the raw performance results, sentiment classification of user-generated content benefits from lexical normalisation as a preprocessing step. In a next step, we investigated whether the results of our normalised system are **significantly** better than those of the not-normalised system. To this end, we applied a t-test (α= 0.05) after bootstrap resampling (Noreen, 1989). More specifically, we drew samples (n= 5,000) with replacement from the output of each system (i.e. the classified instances) and of equal size of the test set (n= 8,987). For each resample, macro-averaged F_1 score was calculated and we subsequently applied a paired samples t-test to compare the mean scores for both systems, which revealed a significant difference ($p< 0.001$) between the results of the *norm* and *not norm* models.

In the above paragraphs, we demonstrated the benefits of normalisation as preprocessing for our sentiment classifier tested on an unseen and varying corpus of user-generated content. In what follows, we take a closer look at both models' performance for each data genre in the test set, the results of which are presented in Table 7.

	SMS2013	**TWE2013**	**TWE2014**	**TWE2014 sarc**	**LJ2014**
	2,093 inst.	3,813 inst.	1,853 inst.	86 inst.	1,142 inst.
TWE not norm	**71.57%**	66.89%	65.73%	46.38%	68.63%
TWE norm	70.31%	**68.26%**	**67.37%**	**50.42%**	**69.67%**

Table 7. *Macro-averaged F_1 scores for each optimised model per genre in the held-out test set.*

Interestingly, the results indicate that both models, which are trained on Twitter data (TWE), perform best on the SMS and LiveJournal genres. Not surprisingly, Twitter sarcasm seems to be the most difficult genre, but it also benefits most from the model where normalisation is included as preprocessing. The figures indicate that the *norm* model performs best for all genres except SMS. In fact, a qualitative analysis revealed that the genre is much more noisy than Twitter (as can be observed in the examples in Table 1 and which was also shown by De Clercq *et al.* (2013)). Hence,

it is not surprising that the *not norm* model better fits the SMS genre, as compared to *norm*.

It is noteworthy, however, that the performance of the normaliser on the SMS set is comparable to its performance on Twitter. In fact, a manual normalisation of 50 instances from the SMS test set showed that recall of the system was 39% and precision 87%, which is comparable to its performance on tweets (cf. section 3). To better evaluate the benefits of normalisation for sentiment analysis on other genres than Twitter, further experiments are necessary where such other genres are included in the training corpus of our sentiment model. In sum, the results in the table indicate that the *norm* model outperforms *not norm* on unseen data, and it generalises well to other data genres (e.g. LiveJournal) or variations (e.g. sarcastic tweets), except SMS.

As the experiments were run using SemEval data, we also compared the performance of our sentiment classifier in the light of the SemEval shared tasks on sentiment analysis. We focus particularly on the SemEval-2014 T9 (Rosenthal *et al.*, 2014) run in which we participated and where participants were required to classify (social media) messages as positive, negative or objective/neutral. Table 8 presents the scores of TeamX (Miura *et al.*, 2014), the Task 9 winning team, and compare them to the results of our two models (i.e. *LT3 norm* and *LT3 not norm*). The TeamX system is based on a supervised text categorisation approach by means of logistic regression. The features that are exploited include bags-of-words, sentiment lexicon features, cluster features, and word sense features. As preprocessing steps the authors included rule-based normalisation, spelling correction, PoS tagging, word sense disambiguation, and negation detection.

It is important to note that Table 8 displays scores obtained by two different evaluations. Firstly, we report the scores obtained by TeamX and LT3 on the three sentiment classes, being positive, negative and neutral. Secondly, we present the results of the SemEval-2014 competition, which scored the systems solely by their performance on the positive and negative class (Rosenthal *et al.*, 2014).

	SMS2013	**TWE2013**	**TWE2014**	**TWE2014sarc**	**LJ2014**	**Full test**
	2,093 inst.	3,813 inst.	1,853 inst.	86 inst.	1,142 inst.	8,987 inst.
TeamX	53.62%	**71.72%**	**69.34%**	**55.06%**	58.11%	65.40%
LT3 (not norm)	**71.57%**	66.89%	64.73%	46.38%	68.63%	68.23%
LT3 (norm)	70.31%	68.26%	67.37%	50.42%	**69.67%**	**68.90%**
Official results SemEval-2014 Task 9						
TeamX	57.36%	72.12%	70.96%	56.50%	69.44%	65.63%
LT3	64.78%	65.56%	65.47%	47.76%	68.56%	60.60%

Table 8. *Comparative results on sentiment classification: the SemEval 2014 winning team versus the current system (with and without normalisation as preprocessing) for the classification of pos/neg/neu sentiment and the official SemEval-2014 Task 9 scores for the classification of pos/neg sentiment.*

As shown in Table 8, with a top F_1 score of 68.90% obtained by our *norm* model, our system compares favourably with the top-performing TeamX system, which obtained an F_1 score of 65.40% on the complete test set. Looking at the different genres in the test set, we observe that TeamX obtained the best results on the Twitter genre, whereas our system performed particularly well on the SMS and LiveJournal (LJ) genre. This demonstrates our system's robustness to data genres other than Twitter, while TeamX' system was particularly tuned to Twitter data, making use of eleven different types of n-gram features (contiguous and not-contiguous, token-based and character-based, both with different levels of granularity) and word clusters generated from a 56M Twitter corpus [7].

Looking at the official SemEval-2014 Task 9 results, we observe that the TeamX system outperformed ours on the positive and negative sentiment class, except for SMS2013. Both systems (i.e. TeamX and LT3) were, however, trained on the three sentiment classes. A possible explanation is that the TeamX system integrated weighting of the predicted labels to handle with the class imbalance in the test set (Miura *et al.*, 2014). Overall, comparing these scores with the performance of our current system clearly demonstrates the beneficial effects of classifier optimisation and normalisation, which allowed us to outperform the Task 9 wining team with 3.5 points (i.e. 68.90% vs. 65.40%).

6. Conclusions and future work

The present research presents the development of an **optimised** sentiment analysis pipeline exploiting a **rich feature set** and investigated the effect of **complex normalisation** as a preprocessing step. Different feature types were exploited, including bags of words, word shape features, syntactic features and sentiment lexicon features. We performed a series of experiments that were evaluated against three baselines (i.e. random, majority class and word unigram), which were all improved at each step of our experimental setup. In a first step, we explored the effect of classifier optimisation by means of joint hyperparameter and feature selection using genetic algorithms. The results revealed that, when applied in its default parameter settings, the LIBSVM model consistently predicted the majority class. Optimisation of the model increased its performance from F_1= 21.30% to F_1= 63%, and eventually to F_1=75.29% when a rich set of features was exploited instead of bags of words.

Having in place an optimised and rich-feature-based sentiment analysis architecture, our focus was on investigating the usefulness of lexical normalisation as a preprocessing step to sentiment classification. Cross-validation experiments on the development data showed that the *norm* model did not outperform the *not norm* model, but experiments on an unseen and varied test set revealed that normalisation did benefit the classification performance. In fact, including normalisation as preprocessing not only improved the classification results for tweets, but also for LiveJournal blog posts,

7. http://www.cs.cmu.edu/ ark/TweetNLP.

a different type of user-generated content. This demonstrates that the *norm* model performs well on unseen data and that it is able to generalise to other genres of UGC than Twitter. For the noisier SMS genre, however, the *not norm* model performed better.

In a last step, we compared the performance of our optimised sentiment classifier (with and without normalisation as preprocessing) to that of the SemEval-2014 winning team and observed that our model performs best overall, and proves more robust to other genres of user-generated content than Twitter.

In future research, it will be interesting to explore "targeted" normalisation, which focuses on increasing the coverage of words that are relevant to the task (i.e. sentiment-bearing words). As was demonstrated in section 4.2.3, besides errors, normalisation also corrects character flooding and inconsistent capitalisation, which could hint at a specific sentiment. Hence, it will be interesting to investigate whether these forms of creative writing can be left unchanged and how this would affect the system performance. Exploring deep learning using neural networks for sentiment classification is another important direction for future work, as several top-ranked systems in the SemEval-2015 to -2017 competitions were based on deep learning. Since our approach involves an intensive process of feature engineering, it would be interesting to see whether competitive or even better results can be achieved when features are automatically deduced from the data using neural networks. Finally, we plan to port our sentiment analysis system to other languages, starting with Dutch.

Acknowledgements

We thank the researchers of TeamX for kindly sharing their SemEval-2014 submission with us, allowing us to compare the performance of our sentiment classifier to their approach.

The work presented in this paper was carried out in the framework of the AMiCA (Automatic Monitoring for Cyberspace Applications) IWT SBO-project 120007.

7. References

Agarwal A., Xie B., Vovsha I., Rambow O., Passonneau R., "Sentiment Analysis of Twitter Data", *Proceedings of the Workshop on Languages in Social Media (LSM 2011)*, ACL, p. 30-38, 2011.

Barbosa L., Feng J., "Robust Sentiment Detection on Twitter from Biased and Noisy Data", *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, ACL, p. 36-44, 2010.

Chandrashekar G., Sahin F., "A survey on feature selection methods", *Computers & Electrical Engineering*, vol. 40, n° 1, p. 16-28, 2014.

Chang C.-C., Lin C.-J., "LIBSVM: A library for support vector machines", *ACM Transactions on Intelligent Systems and Technology*, vol. 2, n° 3, p. 27:1-27:27, 2011.

Dadvar M., Trieschnigg D., de Jong F., "Experts and Machines against Bullies: A Hybrid Approach to Detect Cyberbullies", *Advances in Artificial Intelligence – 27th Canadian Conference on Artificial Intelligence*, p. 275-281, 2014.

Daelemans W., Zavrel J., van der Sloot K., van den Bosch A., TiMBL: Tilburg Memory Based Learner, version 6.2, Reference Guide, Technical Report nº 09-01, ILK Research Group, 2009.

Davidov D., Tsur O., Rappoport A., "Enhanced Sentiment Learning Using Twitter Hashtags and Smileys", *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, ACL, Stroudsburg, PA, USA, p. 241-249, 2010.

De Clercq O., Desmet B., Schulz S., Lefever E., Hoste V., "Normalization of Dutch user-generated content", *Proceedings of the 9th International Conference on Recent Advances in Natural Language Processing (RANLP 2013)*, INCOMA Ltd., p. 179-188, 2013.

De Clercq O., Schulz S., Desmet B., Hoste V., "Towards shared datasets for normalization research", *in* N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, S. Piperidis (eds), *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2014)*, ELRA, p. 1218-1223, 2014.

Deerwester S., Dumais S. T., Furnas G. W., Landauer T. K., Harshman R., "Indexing by latent semantic analysis", *Journal of the American Society for Information Science*, vol. 41, nº 6, p. 391-407, 1990.

Desmet B., Finding the online cry for help: automatic text classification for suicide prevention, PhD thesis, Ghent University, 2014.

Desmet B., Hoste V., Verstraeten D., Verhasselt J., Gallop Documentation, Technical Report nº LT3 13-03, University of Ghent, 2013.

Eisenstein J., "What to do about bad language on the internet", *Proceedings of the North American Chapter of the ACL : Human Language Technologies (NAACL-HLT)*, ACL, p. 359-369, 2013.

Finn A., Kushmerick N., Smyth B., "Genre Classification and Domain Transfer for Information Filtering", *Proceedings of the 24th BCS-IRSG European Colloquium on IR Research: Advances in Information Retrieval*, Springer-Verlag, London, UK, UK, p. 353-362, 2002.

Gimpel K., Schneider N., O'Connor B., Das D., Mills D., Eisenstein J., Heilman M., Yogatama D., Flanigan J., Smith N. A., "Part-of-speech Tagging for Twitter: Annotation, Features, and Experiments", *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies (HLT 2011) – Volume 2*, ACL, p. 42-47, 2011.

Haddi E., Liu X., Shi Y., "The Role of Text Pre-processing in Sentiment Analysis", *Procedia Computer Science*, vol. 17, p. 26-32, 2013.

Han B., Cook P., Baldwin T., "Lexical Normalization for Social Media Text", *ACM Transactions on Intelligent Systems and Technology*, vol. 4, nº 1, p. 5:1-5:27, February, 2013.

Heafield K., "KenLM: Faster and Smaller Language Model Queries", *Proceedings of the 6th Workshop on Statistical Machine Translation (WMT 2011)*, ACL, p. 187-197, 2011.

Hoste V., Optimization Issues in Machine Learning of Coreference Resolution, PhD thesis, Antwerp University, 2005.

Hsu C.-W., Chang C.-C., Lin C.-J., A Practical Guide to Support Vector Classification, Technical report, Department of Computer Science, National Taiwan University, 2003.

Hu M., Liu B., "Mining and summarizing customer reviews", *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD04*, ACM, p. 168-177, 2004.

Joshi M., Penstein-Rosé C., "Generalizing Dependency Features for Opinion Mining", *Proceedings of the ACL-IJCNLP 2009 Conference*, ACL, p. 313-316, 2009.

Kobus C., Yvon F., Damnati G., "Normalizing SMS: Are Two Metaphors Better Than One?", *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008) – Volume 1*, ACL, p. 441-448, 2008.

Koehn P., Hoang H., Birch A., Callison-Burch C., Federico M., Bertoldi N., Cowan B., Shen W., Moran C., Zens R., Dyer C., Bojar O., Constantin A., Herbst E., "Moses: Open Source Toolkit for Statistical Machine Translation", *Proceedings of the ACL 2007 Demo and Poster Sessions*, p. 177-180, 2007.

Kohavi R., John G. H., "Wrappers for Feature Subset Selection", *Artificial Intelligence*, vol. 97, nº 1-2, p. 273-324, December, 1997.

Kökciyan N., Çelebi A., Özgür A., Üsküdarli S., "BOUNCE: Sentiment Classification in Twitter using Rich Feature Sets", *Proceedings of the 2nd Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, ACL, Atlanta, Georgia, p. 554-561, 2013.

Li C., Liu Y., "Normalization of text messages using character-and phone-based machine translation approaches", *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

Liu B., *Sentiment Analysis – Mining Opinions, Sentiments, and Emotions*, Cambridge University Press, 2015.

Liu X., Zhang S., Wei F., Zhou M., "Recognizing named entities in tweets", *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies (HLT 2011)*, ACL, p. 359-367, 2011.

Miura Y., Sakaki S., Hattori K., Ohkuma T., "TeamX: A Sentiment Analyzer with Enhanced Lexicon Mapping and Weighting Scheme for Unbalanced Data", *Proceedings of the International Workshop on Semantic Evaluation (SemEval 2014)*, ACL and Dublin City University, p. 628-632, August, 2014.

Moens M.-F., Li J., Chua T.-S., *Mining User Generated Content*, Chapman & Hall/CRC, 2014.

Mohammad S. M., "Sentiment Analysis: Detecting Valence, Emotions, and Other Affectual States from Text", *in* H. Meiselman (ed.), *Emotion Measurement*, Elsevier, p. 201-226, 2016.

Mohammad S. M., Kiritchenko S., Zhu X., "NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets", *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, ACL, Atlanta, Georgia, USA, p. 321-327, June, 2013.

Mohammad S. M., Turney P. D., "Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon", *Proceedings of the NAACL HLT Workshop on Computational Approaches to Analysis and Generation of Emotion in Text (CAAGET 2010)*, ACL, p. 26-34, 2010.

Mosquera A., Moreda P., "Improving Web 2.0 Opinion Mining Systems Using Text Normalisation Techniques", *Recent Advances in Natural Language Processing (RANLP) 2013*, 2013.

Nakov P., Ritter A., Rosenthal S., Sebastiani F., Stoyanov V., "SemEval 2016 Task 4: Sentiment Analysis in Twitter", *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, ACL, p. 1-18, 2016.

Nenkova A., "Automatic Text Summarization of Newswire: Lessons Learned from the Document Understanding Conference", *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, AAAI Press, p. 1436-1441, 2005.

Nielsen F. A., "A New ANEW: Evaluation of a Word List for Sentiment Analysis in Microblogs", *in* M. Rowe, M. Stankovic, A.-S. Dadzie, M. Hardey (eds), *Proceedings of the ESWC2011 Workshop on "Making Sense of Microposts": Big things come in small packages*, vol. 718, CEUR-WS.org, p. 93-98, 2011.

Noreen E. W., *Computer-Intensive Methods for Testing Hypotheses: An Introduction*, Wiley-Interscience, April, 1989.

Özdemir C., Bergler S., "A comparative study of different sentiment lexica for sentiment analysis of tweets", *Proceedings of Recent Advances in Natural Language Processing (RANLP 2015)*, p. 488-496, 2015.

Pang B., Lee L., "Opinion Mining and Sentiment Analysis", *Foundations and Trends in Information Retrieval*, vol. 2, nº 1-2, p. 1-135, January, 2008.

Pang B., Lee L., Vaithyanathan S., "Thumbs Up?: Sentiment Classification Using Machine Learning Techniques", *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, ACL, p. 79-86, 2002.

Pettersson E., Megyesi B., Tiedemann J., "An SMT approach to automatic annotation of historical text", *Proceedings of the workshop on computational historical linguistics at NODALIDA'13*, nº 87 in *NODALIDA'13*, Linköping University Electronic Press, Oslo, Norway, p. 54-69, 2013.

Raghunathan K., Krawczyk S., CS224N: Investigating SMS Text Normalization using Statistical Machine Translation, Technical report, Stanford University, 2009.

Ritter A., Clark S., Mausam, Etzioni O., "Named Entity Recognition in Tweets: An Experimental Study", *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, ACL, p. 1524-1534, 2011.

Rosenthal S., Farra N., Nakov P., "SemEval 2017 Task 4: Sentiment Analysis in Twitter", *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval 2017)*, ACL, Vancouver, Canada, p. 502-518, 2017.

Rosenthal S., Nakov P., Kiritchenko S., Mohammad S., Ritter A., Stoyanov V., "SemEval 2015 Task 10: Sentiment Analysis in Twitter", *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, ACL, p. 451-463, June, 2015.

Rosenthal S., Ritter A., Nakov P., Stoyanov V., "SemEval 2014 Task 9: Sentiment Analysis in Twitter", *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, ACL and Dublin City University, Dublin, Ireland, p. 73-80, August, 2014.

Roy S., Dhar S., Bhattacharjee S., Das A., "Sentiment in Twitter Events", *Journal of the American Society for Information Science and Technology*, vol. 62, nº 2, p. 406-418, 2011.

Saif H., He Y., Alani H., "Alleviating Data Sparsity for Twitter Sentiment Analysis", *Proceedings of the #MSM Workshop*, vol. 838, CEUR-WS.org, p. 2-9, 2012.

Salton G., *Automatic Text Processing: the Transformation, Analysis and Retrieval of Information by Computer*, Addison Wesley, 1989.

Schneider G., Pettersson E., Percillier M., "Comparing Rule-Based and SMT-Based Spelling Normalisation for English Historical Texts", *Proceedings of the NODALIDA'17 Workshop on Processing Historical Language*, nº 133 in *NODALIDA'17*, Linköping University Electronic Press, p. 40-46, 2017.

Severyn A., Moschitti A., "UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification", *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, ACL, p. 464-469, 2015.

Sharma S., Srinivas P., Balabantaray R. C., "Text normalization of code mix and sentiment analysis", *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, p. 1468-1473, 2015.

Sokolova M., Lapalme G., "A systematic analysis of performance measures for classification tasks", *Information Processing & Management*, vol. 45, nº 4, p. 427-437, 2009.

Stone P. J., Dunphy D. C. D., Smith M. S., Ogilvie D. M., *The General Inquirer: A Computer Approach to Content Analysis*, The MIT Press, 1966.

Wiebe J. M., "Learning Subjective Adjectives from Corpora", *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, AAAI Press, p. 735-740, 2000.

Wiebe J. M., Wilson T., Bruce R., Bell M., Martin M., "Learning Subjective Language", *Computational Linguistics*, vol. 30, nº 3, p. 277-308, 2004.

Wilson T., Wiebe J., Hoffmann P., "Recognizing Contextual Polarity in Phrase-level Sentiment Analysis", *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT 2005)*, ACL, p. 347-354, 2005.

Xue Z., Yin D., Davison B. D., "Normalizing Microtext", *Proceedings of the 5th AAAI Conference on Analyzing Microtext (AAAIWS 2011)*, AAAI Press, p. 74-79, 2011.

Yu L., Liu H., "Feature selection for high-dimensional data: a fast correlation-based filter solution", *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*, p. 856-863, 2003.

Zhu X., Kiritchenko S., Mohammad S. M., "NRC-Canada-2014: Recent Improvements in the Sentiment Analysis of Tweets", *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, ACL and Dublin City University, p. 443-447, 2014.

Notes de lecture

Rubrique préparée par Denis Maurel

Université François Rabelais Tours, LI (Laboratoire d'informatique)

Annelies BRAFFORT. La Langue des Signes Française (LSF). Modélisations, ressources et applications. *ISTE Editions*. 2016. 224 pages. ISBN 978-1-78405-050-4.

Lu par **Stéphanie GOBET**

Université de Poitiers – Laboratoire Forell

L'ouvrage dirigé par Annelies Braffort s'adresse en particulier à des étudiants aguerris, ou des ingénieurs et chercheurs spécialisés en TAL. Il est constitué de cinq chapitres dont le fil conducteur est la langue des signes abordée sous différents aspects : l'histoire, la pédagogie, la modélisation informatique et ses applications. Nous attirons l'attention sur les chapitres 2 et 3 qui exigent une lecture spécialisée en TAL.

Le premier chapitre introduit l'ouvrage en proposant une partie qui a pour vocation de vulgariser les recherches en linguistique des langues des signes afin que le lecteur, non initié à ces langues dont la modalité est visuo-gestuelle, découvre les travaux et leurs applications. Les auteurs, après un rappel sociolinguistique concernant le statut de la LSF et de ses locuteurs, abordent différents domaines, comme la phonologie, la poésie… Comme le rappellent les auteurs tout au long de leur chapitre, la langue des signes est devenue un objet pertinemment scientifique bousculant les modèles classiques, tout en en proposant de nouveaux, comme en TAL par exemple.

Le deuxième chapitre se focalise, de façon très pédagogique, sur la question des corpus en langue des signes. Après avoir rappelé les notions de transcription et d'annotations, tout en évaluant les apports et les difficultés liés à la modalité visuo-gestuelle de la langue des signes, les auteurs exposent la typologie des corpus à partir d'exemples et posent la problématique de la transcription des LS qui peut se réaliser sous forme graphique ou sous forme de gloses. Par conséquent, ce chapitre a pour intérêt de présenter les types de corpus existants, comment ils sont recueillis en langue des signes, avec quel protocole, mais surtout quelles en sont les exploitations possibles. La présentation comparative de différents projets (CREAGEST, TALS, DICTA-SIGN, par exemple) est l'occasion de discuter du traitement des données selon les objectifs du protocole et de l'archivage de ces données visuelles.

Le troisième chapitre traite plus précisément et de manière très technique de la modélisation linguistique appliquée à la langue des signes. Les auteurs définissent dans un premier temps les différents types de données disponibles (brutes, degré

intermédiaire, etc.) pour les chercheurs en TALS (traitement automatique des langues signées). Des exemples illustrent les propos des auteurs, mais surtout permettent de visualiser le type de données et leur traitement. Il existe aujourd'hui différentes technologies et différents logiciels pour les représenter (image, rotoscopie, modèle Zebedee, signeur virtuel, etc.). Comme l'exposent les auteurs, les modèles linéaires ne permettent pas de représenter la multilinéarité spécifique aux langues des signes. Par conséquent, les modèles doivent être repensés et ont donné lieu à de nouveaux travaux concernant la traduction automatique appliquée aux langues des signes.

Les deux derniers chapitres sont des mises en application de la modélisation informatique en langue des signes.

L'avant-dernier chapitre a une visée plus spécifiquement pédagogique. Il s'adresse à tous ceux qui réfléchissent aux supports pédagogiques. Après être revenus sur le statut de la langue des signes au sein du système éducatif et sur le combat mené pour sa reconnaissance par la loi (en France et dans le monde), les auteurs abordent la question de la trace écrite pour les langues à modalité visuo-gestuelle. Comme il est explicitement exposé, le support vidéo est devenu, au-delà d'une simple trace, un outil pédagogique et didactique. Bien que non reconnue par l'Éducation nationale, la vidéo répond aux besoins des enseignants lors de la création de cours, mais est aussi devenue le support de glossaires. Des sites tels que Wikisign ont été conçus comme des ressources lexicales. Les auteurs, à travers cet article, attirent l'attention du lecteur sur la problématique de la langue des signes qui n'a pas d'équivalent écrit, alors que l'écrit est nécessaire en pédagogie et en didactique lors de la transmission du savoir. Face à ce besoin, des ressources pédagogiques ont vu le jour, telles que LogiSignes dont les logiciels sont décrits à la fin du chapitre. Pour toute personne réfléchissant à l'amélioration de ses cours, les auteurs apportent ici des exemples concrets – tout en les analysant –, offrant ainsi aux pédagogues des outils pour améliorer leurs pratiques.

Le cinquième et dernier chapitre traite de l'application de la modélisation informatique des langues des signes au grand public. L'innovation, en termes de logiciels et de ressources, a créé de nouveaux besoins pour la communauté sourde. L'un d'entre eux est la transmission de l'information au grand public. Cette transmission, générée par la vidéo, doit répondre à différentes contraintes exposées par les auteurs et peut se réaliser différemment. Les auteurs expliquent, en particulier, comment la LS peut s'articuler dans un document, soit par incrustation soit par juxtaposition, avec des éléments tels que l'image, le pictogramme, la vidéo, voire la conjonction de plusieurs éléments comme il apparaît sur le site Internet Macif Sourds. Toutefois, la question de cette articulation a soulevé le questionnement dû à la multimodalité des langues des signes. Ces dernières n'étant pas linéaires, la problématique de l'articulation des LS dans le document est confrontée à des problématiques autres telles que l'interaction et la navigation dans l'ensemble des documents. La typologie des LS implique donc de développer de nouveaux systèmes afin que l'accès à l'information pour la communauté sourde soit optimal. Les auteurs signalent tout de même l'existence d'applications à destination du grand public tout en rappelant la difficulté d'évaluer ces outils. Ce chapitre

conclut sur un aspect abordé tout au long de l'ouvrage : le signeur virtuel. L'avatar, dont le processus de création a été présenté dans le chapitre précédent, constitue une avancée dans le domaine public par son confort de visualisation, à condition de répondre à des critères tels que la visibilité ou encore l'esthétique.

Bien que l'ouvrage concerne davantage les spécialistes en TAL, il a été conçu de manière très pédagogique, les illustrations et exemples permettant de visualiser les concepts traités pour tout lecteur souhaitant avoir une première approche des travaux en TAL. Toutefois, les chapitres deux et trois requièrent des prérequis en traitement de corpus et modélisation informatiques.

Jannik STRÖTGEN, Michael GERTZ. Domain-Sensitive Temporal Tagging. *Morgan & Claypool Publishers*. 2016. 133 pages. ISBN 978-1-62705-459-1.

Lu par **Yannis HARALAMBOUS**

IMT – UMR CNRS 6285 Lab-STICC

Le titre de cet ouvrage nous renseigne déjà sur deux de ses particularités : primo, Temporal Tagging *indique qu'il s'agit non pas d'extraction d'informations temporelles en général, mais seulement de balisage temporel. L'ouvrage se concentre donc sur l'extraction, la classification et la normalisation d'expressions temporelles, et ne s'occupe pas de leur interprétation, de l'extraction d'évènements et de relations entre ces évènements. Secundo,* Domain-Sensitive, *c'est-à-dire « spécifique à un domaine donné », représente la véritable innovation de l'ouvrage, puisqu'il s'agit de comparer le balisage temporel de quatre types de documents, de natures assez différentes.*

Le premier auteur, Jannik Strötgen, post-doctorant à l'Institut Max-Planck de Sarrebruck, est un spécialiste de l'annotation temporelle aussi bien du point de vue du Web sémantique, que de celui de l'étude de corpus historiques et littéraires. Le deuxième auteur, Michael Gertz, professeur à l'université Heidelberg et directeur de l'Institut d'informatique de celle-ci, participe à un grand nombre de projets, dans différents domaines du traitement automatique de langue et de la modélisation. D'ailleurs, cet ouvrage est fondé sur la thèse de doctorat du premier auteur, dont le deuxième auteur était directeur de thèse.

Structure de l'ouvrage

L'ouvrage est divisé en six chapitres, dont les trois premiers sont introductifs, le quatrième traite du cœur de la question (spécificités du balisage temporel selon le domaine d'application), le cinquième énumère un certain nombre d'outils, méthodes et projets, et le sixième conclut avec des directions de recherches futures. L'ouvrage peut paraître court (133 pages en tout, dont seulement 38 pour le quatrième chapitre, qui est le plus important), mais il est dense et contient énormément d'informations, notamment sur les ressources externes, les projets, et les défis.

Chapitres 1 à 3

L'introduction réussit sans trop de mal à nous convaincre de l'importance et de la complexité de l'extraction d'informations temporelles, notamment en illustrant

cette tâche par une page Wikipédia munie d'un ensemble de flèches enchevêtrées, reliant les expressions temporelles contenues dans le texte et les points ou intervalles de la frise chronologique correspondante.

Le deuxième chapitre présente de manière très simple la mesure standard du temps et les quatre types d'expressions temporelles : explicite (« je suis né le 16 février 1962 »), sous-spécifiée (« j'ai terminé mes lectures vendredi »), relative (« ils se sont vus hier »), et implicite (« les vacances commencent le lundi de Pâques de cette année »). Ce chapitre est tellement clair et concis, que l'on regrette presque l'abondance encyclopédique à laquelle un autre auteur (comme Umberto Eco[1], pour ne pas le nommer) se serait certainement prêté : il n'est question ni de calendrier hégirien (avec ces mois de vingt-neuf à trente jours), ni de calendrier républicain (avec des décades à la place des semaines), ni des turpitudes du passage du calendrier julien au calendrier grégorien, passage différent dans chaque pays, qui s'est étalé sur cinq siècles, ni de la célèbre interprétation du mot « maintenant » par Michel Serres comme « Main tenant, tenant en main le monde »... Les auteurs n'étaient certainement pas obligés de parler de tout cela, mais cela aurait pu être une agréable surprise pour le lecteur.

Le troisième chapitre est très hétérogène. Il commence par une description technique et détaillée des normes XML afférentes (TIDES et TimeML), se poursuit par une section sur l'évaluation des outils d'analyse, enchaîne naturellement en énumérant les différentes compétitions, et enfin, donne une liste de corpus de type journalistique existant dans différentes langues.

Chapitre 4

Le quatrième chapitre part du constat que l'écrasante majorité des textes analysés sont des textes journalistiques (en anglais : *news style*), textes pour lesquels l'annotation temporelle est cruciale, et qui sont rédigés de manière assez similaire. Les auteurs définissent un domaine comme un « groupe de documents ayant les mêmes caractéristiques vis-à-vis de la tâche de balisage temporel » et énumèrent des corpus de documents temporellement balisés de domaines autres que le domaine journalistique : on y trouve des textes d'histoire, des collections de pages Wikipédia, des SMS, ainsi que des récits cliniques et des rapports de pathologie.

Pour mettre de l'ordre dans cette nébuleuse de méthodes de représentation et d'utilisation de la temporalité, ils décrivent quatre grands domaines (au sens du balisage temporel) :

1) *les documents journalistiques* : on y trouve un grand nombre d'expressions temporelles sous-spécifiées et relatives ; le temps de référence est souvent le temps de création du document ; les temps des verbes sont importants pour normaliser les expressions sous-spécifiées ; néanmoins, le temps du verbe peut induire en erreur,

1 En effet, dans le *Pendule de Foucault*, chap. 71, Eco décrit un rendez-vous manqué entre Rose-Croix anglais et français, le même jour étant pour les premiers le 13 juin 1584 et pour les deuxièmes le 23 juin, puisque la réforme du calendrier a aboli dix jours en 1583 en France, mais n'a été adoptée en Angleterre qu'en 1752.

ainsi dans la phrase « la réunion a été reportée à lundi de la semaine prochaine », le verbe est au passé, mais la date donnée est dans le futur (sans doute parce que l'information implicite est « la réunion a été reportée et aura lieu le lundi de la semaine prochaine », où le verbe est bien au futur) ;

2) *les documents narratifs*, l'exemple paradigmatique étant les pages Wikipédia : on y trouve un grand nombre d'expressions explicites, mais aussi des expressions relatives à ces dernières ; le temps de référence change assez souvent et on est amené à déterminer les expressions sous-spécifiées par rapport à celui-ci ; de par cette relativité, des erreurs peuvent se propager, et on trouve parfois des phénomènes qui peuvent être modélisés par des automates de type fini, par exemple les ères « av. J.-C. » et « apr. J.-C. » : l'auteur ayant donné une date av. J.-C. aura tendance à ne plus spécifier explicitement l'ère jusqu'à l'occurrence d'une date dans l'autre ère, et *vice-versa* ;

3) *les documents en langage informel ou familier*, l'exemple paradigmatique donné étant des corpus de SMS : il faut interpréter les parophonies (« 2m1 » pour « demain ») et détecter les erreurs ; le temps de référence est le temps de publication ; le contexte est souvent manquant ; les temps des verbes ne sont pas toujours cohérents (« j'arrive demain », le verbe est au présent alors que l'action se situe au futur) ;

4) *les documents autonomiques* : le terme *autonomique* provient de la médecine et se réfère à des processus biologiques *involontaires* par opposition à des processus *indépendants* qui sont qualifiés d'*autonomes* (par exemple, le système nerveux autonome est appelé *autonomic* en anglais au lieu d'*autonomous*). Les auteurs désignent par ce terme les documents contenant une majorité d'expressions temporelles qui ne peuvent pas être normalisées dans un cadre temporel absolu, mais doivent rester dans un cadre temporel local (exemple : les fameux « T = 0, T + 3, T + 6, T + 12 » que l'on trouve dans les plannings). En guise d'exemple, ils mentionnent les textes scientifiques et les récits cliniques. Dans ce type de documents, il s'agit de détecter et de déterminer le « temps zéro » et d'interpréter la sémantique des expressions relatives.

Les auteurs comparent ces quatre domaines, et donnent des stratégies d'analyse circonstanciées.

Chapitres 5 et 6

Le cinquième chapitre nous offre un panorama des outils existants et de différentes compétitions du domaine, y compris celles comportant des corpus multilingues. Le sixième chapitre comprend une conclusion et une liste de directions de recherches futures proposées : la comparaison des annotations temporelles dans les différentes langues, l'identification du temps de référence, la compréhension des cadres temporels locaux, l'adaptation à des disciplines spécifiques (par exemple, la paléontologie ou le sport), l'identification automatique du domaine, l'évaluation de la confiance des expressions temporelles, le traitement des zones horaires, les références temporelles implicites (« pendant la campagne présidentielle 2017 », « à la saint-glinglin »), etc.

Conclusion

Les points positifs de cet ouvrage sont nombreux : il est concis et extrêmement clair, et il fourmille de liens vers des outils, des ressources, des projets et des compétitions.

En guise de point négatif, on peut noter que le choix des « domaines » opéré par les auteurs peut sembler un peu arbitraire et que les auteurs ne font appel à aucune théorie linguistique pour fonder cette classification. Quoi qu'il en soit, les exemples donnés sont intéressants et le lecteur ayant parcouru le quatrième chapitre est tout à fait en mesure d'appliquer les mêmes méthodes à d'autres types de documents auxquels il peut être confronté[2]. Pour conclure, nous considérons que cet ouvrage est bien indiqué à ceux qui ont des expressions temporelles à baliser, et même à ceux qui souhaitent s'y investir et participer aux diverses compétitions.

Karën FORT. Collaborative Annotation for Reliable Natural Language Processing: Technical and Sociological Aspects. *Wiley-Iste*. 2016. 164 pages. ISBN 978-1-84821-904-5.

Lu par **Aurélien BOSSARD**

Université Paris 8 – Laboratoire d'informatique avancée de Saint-Denis

---

L'autrice part du constat que les ressources textuelles annotées sont un élément décisif pour tout système de traitement automatique du langage. Elles sont en effet nécessaires à la fois pour l'entraînement des systèmes et pour leur évaluation. L'autrice définit ce qu'est une annotation et les enjeux généraux de l'annotation puis décrit le sujet, les enjeux spécifiques, et dresse un état des lieux de l'annotation participative et/ou collaborative.

L'ouvrage d'environ cent vingt pages plus l'appendice a pour objectif de décrire les méthodologies et les outils existants d'annotation collaborative et participative pour le traitement automatique du langage ainsi que d'en montrer les forces et les faiblesses. L'annotation collaborative et sa petite sœur, l'annotation participative, sont des enjeux majeurs du TAL.

Dans une introduction très documentée et illustrée, l'autrice définit ce qu'est une annotation, en replaçant le processus d'annotation dans une chronologie qui court des débuts de l'écriture à nos jours.

---

2 Sans oublier, en tant qu'exemple extrême, les textes à frise chronologique circulaire, comme les récits de voyages temporels, qui sont un véritable sous-genre de la science-fiction, et dont H. G. Wells (*La machine à explorer le temps*), J. Finney (*Le voyage de Simon Morley*), R. A. Heinlein (*Vous les zombies*), R. Silverberg (*Les temps parallèles*), C. Webb (*Les quinze premières vies d'Harry August*) et notre cher R. Barjavel national (*Le voyageur imprudent*) sont des illustres représentants.

Le chapitre 1 est dédié à l'annotation collaborative. Y sont évoqués les enjeux ainsi que les méthodologies de la constitution collaborative de données annotées. Y sont également décrites clairement différentes méthodologies ainsi que leurs points communs et leurs différences, depuis l'identification des acteurs de l'annotation, en passant par l'écriture de guides d'annotation, jusqu'aux différents outils d'évaluation de la qualité d'annotation.

Ce chapitre met également en évidence, après avoir présenté les formats d'annotation et les outils existants, les défis posés à ces derniers : la généricité, la prise en compte de l'aspect collaboratif et les biais associés, ainsi que la place du processus d'annotation dans une campagne d'annotation.

Le deuxième et dernier chapitre est consacré à l'annotation participative dont l'autrice avait déjà montré qu'elle est une évolution naturelle de l'annotation collaborative et qui remet la participation citoyenne au centre d'un processus scientifique. Après avoir défini l'annotation collaborative, l'autrice montre les spécificités de ce type d'annotation : nécessité d'évaluer et de prendre en compte la compétence des participants, problématiques liées aux quantités des participants et de leur contribution, motivation de ces derniers (financière comme pour *Amazon Mechanical Turk*, ludique au travers de jeux sérieux...).

Ce chapitre passe en revue les différentes utilisations que l'on peut faire d'un participant dans le cadre des jeux sérieux, de la simple utilisation des capacités innées jusqu'à celle des capacités d'apprentissage des joueurs. Le chapitre clôt avec les problèmes éthiques liés aux jeux sérieux et aux plateformes participatives rémunérées et démontre la nécessité de l'utilisation d'une charte éthique pour les scientifiques et les financeurs.

L'ouvrage comporte également un appendice qui référence, de manière très exhaustive, les outils existants dans les domaines de l'annotation collaborative et de l'annotation participative, les classe en trois catégories (génériques, orientés tâches et spécifiques au TAL) et les décrit ainsi que leurs points forts et leurs faiblesses.

L'ouvrage se termine par un glossaire réduit au strict minimum et une bibliographie très complète.

Commentaires

Cet ouvrage, extrêmement bien documenté et complet, est, à ma connaissance, la seule référence à l'heure actuelle pour l'annotation collaborative et participative dans le domaine du traitement automatique du langage. Il repose sur des années de recherche et de collaborations de la part de l'autrice. L'ouvrage, par l'effort pédagogique auquel a consenti l'autrice, est accessible à toute personne qui s'intéresse au traitement automatique du langage, mais également utile, par la qualité et la complétude de l'état de l'art et le recul de l'autrice sur le sujet, à un chercheur expérimenté.

Le traitement automatique du langage est fortement dépendant de la quantité et de la qualité des ressources annotées, à une heure à laquelle les systèmes d'apprentissage supervisés prennent une place de plus en plus prépondérante. Ainsi,

constituer et disposer de ressources annotées fiables est essentiel. C'est pourquoi nous ne pouvons que recommander la lecture de cet ouvrage à tout chercheur qui se prépare soit à mettre en place une campagne d'annotation collaborative, soit à utiliser des données issues de l'annotation collaborative et/ou participative. En effet, tous les problèmes, qu'ils soient d'ordres méthodologique, quantitatif, qualitatif ou éthique ne sont évidents ni à cerner, ni à résoudre.

Xavier-Laurent SALVADOR. XML pour les linguistes. *L'Harmattan*. 2016. 189 pages. ISBN 978-2-34309-956-9.

Lu par **Lydia-Mai Ho-Dac**

Université de Toulouse – CLLE-ERSS

« XML pour les linguistes » *se donne pour objectif de présenter le langage XML et ses applications à des chercheurs en lettres et sciences humaines et sociales à même de produire des ressources langagières dans un format moins explicite et partagé que l'XML. L'objectif est clairement de rassurer un public parfois frileux devant la technologie informatique et de le convaincre que le langage XML est adapté aux recherches en linguistique. La présentation commence par les éléments historiques dont hérite le langage XML pour introduire de façon quelque peu littéraire le fonctionnement du langage XML. Viennent ensuite des parties plus didactiques qui expliquent la syntaxe XML, l'utilité et la création de DTD et schémas XSD, les normes XML (TEI, RDF, OWL) et les langages de requête et de transformation disponibles (Xpath, XSL, XQuery). La dernière partie présente trois projets menés par l'auteur mettant en jeu des ressources XML variées (corpus littéraires, dictionnaires, transcriptions) et l'outil Isilex développé par l'auteur.*

Cet ouvrage se donne pour objectif de présenter le langage XML et son potentiel à des étudiants et chercheurs en lettres et sciences humaines et sociales. Tout en définissant les règles de base du langage XML, l'auteur justifie, documente et illustre l'utilisation de cette norme pour des études en linguistique et plus largement des études manipulant du matériau langagier.

Les notions informatiques présentées sont régulièrement reliées à des notions historiques (le chevron, caractère utilisé par les copistes bibliques pour distinguer des éléments textuels différents, le codex comme éternelle base de représentation d'un texte, etc.) dans un souci de montrer que finalement rien n'est vraiment nouveau. Cette attention semble directement adressée à un public frileux à l'égard des technologies informatiques et que l'auteur souhaite convaincre de la simplicité et du bien-fondé du langage XML.

Les avantages du langage XML pour la recherche en lettres et sciences humaines et sociales sont documentés et illustrés à travers une variété d'exemples de ressources en XML (dictionnaires, lexiques, corpus de textes bruts et annotés, etc.) et de projets collaboratifs.

Organisation et contenu

L'ouvrage commence par une introduction à certaines notions fondamentales pour manipuler des formats numériques (le « plasma numérique » selon l'auteur) : octets, caractères et encodage, documents numérisés *vs* électroniques, formats de fichiers, etc.

Le chapitre 2 décrit le langage XML et un certain nombre de normes associées. Il commence par un aperçu des origines du langage XML en tant que convention typographique, puis de la syntaxe XML présentée comme permettant de décrire et déclarer des objets langagiers, à la manière des didascalies (« ceci est un paragraphe »). Après une définition des espaces de nommage, décrits comme des dialectes de XML, une large partie de chapitre 2 est dédiée à la nécessité de normaliser la description et l'encodage des éléments d'une ressource, ainsi qu'aux moyens utilisés pour assurer cette normalisation et, à terme, une pérennisation de la ressource. Sont parcourus les formalismes XML utilisés par les outils de traitement de texte (Open Office et MSWord), la gestion et la création de DTD et de schémas XML, la norme TEI pour l'encodage de ressources textuelles, le modèle RDF pour la structuration des données du Web sémantique, et le format OWL pour l'encodage de ressources terminologiques et les ontologies du Web.

Le troisième chapitre est consacré aux langages de requête et de transformation XPath, XSL et XQuery. La présentation est orientée de façon à montrer toute la valeur ajoutée d'une ressource structurée en langage XML. Le chapitre commence par les bases de la syntaxe d'une requête XPath et d'une transformation XSLT, avant de décrire le langage de requête et de transformation XQuery, langage qui sera utilisé dans les applications et projets présentés dans la suite de l'ouvrage. Le choix pour le XQuery est justifié par son potentiel (c'est un langage de programmation en tant que tel) et la possibilité de l'utiliser *via* le logiciel BaseX, une application multiplateforme *open source* développée à l'université de Konstanz. La fin de ce chapitre illustre le potentiel de la combinaison XML, Xpath et XQuery par des « exemples de manipulations de corpus en XQuery » avec le logiciel BaseX : exploration de corpus bruts ou annotés, annotation de corpus et mise en place d'une interface en ligne pour la consultation et l'interrogation d'une ressource XML. Le chapitre fini sur une liste éclectique d'outils permettant la création, la manipulation et/ou l'exploitation de ressources XML : Oxygen. ToolBox, TXM, Transcriber, PRAAT, ELAN, et Isilex, développé par l'auteur et utilisé dans les projets décrits dans le chapitre 4.

Le quatrième et dernier chapitre présente trois projets menés par X.-L. Salavador avec le logiciel Isilex. Ces projets illustrent des cas de (1) détection automatique de thématiques dans des textes littéraires ; (2) de visualisation d'annotations sous forme de graphes ; (3) de construction et gestion collaborative du dictionnaire Crealscience, dictionnaire de français scientifique médiéval ; (4) de transformation du format XML vers le format LaTeX ; (5) de transcription collaborative pour une version numérique et consultable de l'Exode de la *Bible historiale*.

Commentaire

« *XML pour les linguistes* » est conçu comme un livre qui se lit plus qu'un manuel qui s'utilise. L'auteur met l'accent sur le fait que le langage XML hérite de concepts issus de la tradition de l'édition et de l'écriture depuis ses origines, parfois au détriment d'indications simples pour la bonne prise en main du langage XML par un public néophyte.

Le caractère didactique est toutefois présent dans les trois premiers chapitres, notamment grâce à un lexique qui fournit des définitions complètes des termes utilisés dans l'ouvrage et également à un certain nombre d'encadrés offrant des résumés didactiques (principales règles de syntaxe XML, procédures pour construire un projet collaboratif de base de données XML, etc.) et des mini-tutoriels (gérer des fichiers XML en ligne de commande, etc.). On regrettera cependant le peu d'outils pour faciliter la navigation à l'intérieur du document : pas de renvoi depuis le lexique aux pages pertinentes, absence de table des encadrés, des figures et exemples utilisés, faible nombre de titres de section pour pointer sur des aspects spécifiques du langage XML.

Cet ouvrage s'adresse donc à des étudiants et chercheurs en lettres et sciences humaines et sociales qui ne cherchent pas un manuel, mais plutôt une sorte de cours à lire pour comprendre pourquoi et comment utiliser le langage XML. La première partie peut tout à fait être utilisée en support de cours par des néophytes. La partie consacrée à la manipulation de ressources XML par le langage XQuery et le logiciel BaseX est davantage construite comme un tutoriel adressé à un public plus averti ayant certaines compétences en programmation et gestion de systèmes informatiques, des « ingénieurs du texte » selon les propos de l'auteur. Le dernier chapitre sert, quant à lui, à présenter des projets menés par l'auteur, sans spécialement adopter une orientation didactique.

Yoav GOLDBERG. Neural Network Methods for Natural Language Processing. *Morgan & Claypool Publishers*. 2017. 287 pages. ISBN 978-1-62705-298-6.

Lu par **Franck Burlot**

Limsi – CNRS

L'ouvrage propose une initiation à la fois aux méthodes neuronales et aux différentes architectures qui constituent l'état de l'art en traitement automatique des langues (TAL). Il est destiné aussi bien aux étudiants qu'aux ingénieurs et chercheurs professionnels en TAL et ne suppose que quelques connaissances mathématiques basiques. Les qualités didactiques déployées par l'auteur permettent d'aborder progressivement des architectures complexes et d'assimiler les connaissances nécessaires pour être indépendant dans l'exploration de la littérature spécialisée. Les modèles principaux sont exposés (perceptron, réseaux convolutifs et récurrents) et appliqués de manière pratique à différentes tâches de classification et de

génération (analyse des sentiments, étiquetage, traduction automatique). L'importance des modèles neuronaux dans le TAL ne cesse de grandir et cet ouvrage est une ressource unique pour se former efficacement à l'apprentissage profond.

L'ouvrage consiste en une introduction aux réseaux de neurones appliquée au traitement automatique des langues (TAL). Il évoque les tâches caractéristiques du domaine, pour lesquelles on observe, depuis quelques années seulement, le succès grandissant des méthodes dites « d'apprentissage profond ». L'auteur a publié en 2016 dans la revue « *Journal of Artificial Intelligence Research* » un rapport de soixante-seize pages sur le même sujet, intitulé *A Primer on Neural Network Models for Natural Language Processing*, destiné à la communauté du TAL. L'ouvrage présenté ici reprend ce rapport, l'étend, et s'ouvre ainsi à un public plus large en manifestant un effort didactique considérable et en ajoutant une présentation assez détaillée du TAL. S'il peut donc être intéressant pour un spécialiste d'apprentissage automatique, son propos principal reste le fonctionnement des modèles neuronaux auxquels il s'agit d'initier les spécialistes, ou simplement amateurs, du TAL. En effet, outre quelques connaissances très basiques en algèbre linéaire (opérations de matrices) et en statistiques (chaînes de Markov), la connaissance d'aucun domaine particulier n'est nécessaire. L'auteur précise en préface qu'il ne s'agit nullement d'une introduction à l'apprentissage automatique, néanmoins il ne fait référence à aucune connaissance particulière à avoir acquise avant la lecture de l'ouvrage, qui s'ouvre sur un chapitre simple et clair décrivant les modèles linéaires.

L'ouvrage est publié dans la collection « *Cours de synthèse sur les technologies du langage humain* » et il s'agit bien là d'un cours, dont la valeur pédagogique est indéniable. Les différents modèles neuronaux présentés sont inclus dans le cadre d'une progression systématique, où chaque nouveau détail fait référence à ce qui a été présenté précédemment. Ainsi, après la description de la régression linéaire, le perceptron multicouche est décrit comme une simple « superposition de modèles linéaires séparés par une fonction de non-linéarité ». Cette simplicité dans la formulation est constante et conduit naturellement à des architectures très complexes en fin d'ouvrage.

Le cours présente la théorie nécessaire à la compréhension des modèles neuronaux, sans toutefois noyer le lecteur dans des descriptions abstraites, et introduit régulièrement des notions très pratiques. Ainsi, une section est consacrée à la notion de graphe computationnel, qu'il est essentiel d'acquérir puisqu'elle est employée dans la plupart des modules d'implémentation de réseaux de neurones (Theano, TensorFlow, DyNet). Par ailleurs, l'auteur donne régulièrement des conseils pratiques concernant des détails d'implémentation ou d'utilisation. Il met par exemple le lecteur en garde contre l'utilisation telle quelle de représentations continues de mots préentraînées et propose plusieurs méthodes efficaces pour les intégrer dans un nouveau modèle. Ces détails sont précieux, puisqu'ils sont parfois omis dans la littérature et traités comme allant de soi. C'est là une importante ambition de l'auteur : rendre le lecteur capable d'aborder la littérature technique et scientifique sur le sujet. En effet, certains articles sont parfois elliptiques et omettent des détails importants, comme la présence d'une fonction d'activation ou d'un *softmax*. C'est dans ce même souci qu'il s'arrête rigoureusement sur les flous

terminologiques courants dans un domaine qui connaît actuellement un développement spectaculaire. Ainsi, la « log-vraisemblance négative » est aussi désignée comme « l'entropie croisée », une « fonction non linéaire » comme une « fonction d'activation », une « convolution » comme un « filtre », etc.

La bibliographie est abondante. L'auteur y inclut des ouvrages et articles fondateurs, ainsi que des descriptions de modèles très récents au moment de la publication, au risque qu'ils soient dépréciés rapidement.

La rédaction de l'ouvrage est assez claire, mais comprend un très grand nombre de coquilles qui ne gênent toutefois la compréhension que rarement. Ces négligences dans le travail d'édition semblent courantes chez Morgan & Claypool Publishers. La structure en quatre parties est judicieuse et permet une progression efficace.

La première partie introduit les notions de base (perceptron, rétropropagation, optimisation). L'auteur se désolidarise à juste titre de l'analogie entre les réseaux de neurones artificiels et l'activité cérébrale humaine, qui connaît un heureux succès dans la presse, et présente la notation mathématique qui sera utilisée dans tout l'ouvrage. Cette partie est probablement celle qui nécessitera le plus d'effort de la part du lecteur peu familiarisé à l'apprentissage automatique.

La deuxième partie propose un tour d'horizon des tâches les plus célèbres du TAL, qui concentreront cette fois l'attention du lecteur spécialiste d'apprentissage automatique non initié au domaine. Elle aborde également la question des enjeux liés aux différentes représentations symboliques et continues de la langue. La tâche de modélisation de la langue est présentée en détail et conduit aux algorithmes les plus employés pour apprendre des représentations continues de mots, ainsi que les différentes manières d'utiliser ces représentations. La partie se clôt sur la présentation complète d'un modèle d'inférence du sens des phrases, qui met en application de nombreuses notions abordées plus tôt.

La troisième partie évoque les deux types d'architectures couramment utilisées en TAL : les réseaux convolutifs et récurrents, avec une interprétation très intuitive des différentes composantes d'un LSTM. La performance des réseaux récurrents dans la génération de texte est mise en valeur et le modèle d'encodeur-décodeur avec mécanisme d'attention est présenté, là encore, de manière très progressive et intuitive, ce qui permet de dresser une description assez détaillée du nouvel état de l'art en traduction automatique, dont certaines composantes sont apparues moins d'un an avant la publication de l'ouvrage.

La quatrième partie est plus disparate et évoque des sujets plus avancés. Après une présentation des réseaux récursifs pour l'encodage des arborescences, l'auteur reprend les questions liées à l'étape du décodage et à l'exploration de l'espace de recherche dans une sortie structurée, qui relèvent plutôt de l'apprentissage automatique général. Cette section est brève et apparaîtra probablement comme complexe au lecteur non familiarisé au domaine. L'ouvrage se clôt sur la combinaison de différents modèles et introduit notamment l'apprentissage multitâche.

À l'issue de cette lecture, le spécialiste de TAL qui découvre les modèles neuronaux est en mesure d'aborder une grande partie de la littérature de l'état de l'art et n'a plus qu'à choisir un module pour se lancer dans ses premières implémentations. Cette ambition rendue réaliste fait de l'ouvrage l'entrée la plus efficace dans le domaine de l'apprentissage profond.

Jeffrey HEINZ, Colin de la HIGUERA, Menno van ZAANEN. Grammatical Inference for Computational Linguistics. *Morgan & Claypool Publishers*. 2015. 136 pages. ISBN 978-1-60845-977-3.

Lu par **Isabelle TELLIER**

Université Paris 3 – Sorbonne Nouvelle – Laboratoire LaTTiCe, UMR 8094

Ce livre propose un panorama synthétique et pédagogique de l'inférence grammaticale, domaine qui se consacre à l'apprentissage automatique de modèles de langages (comme les automates finis ou les grammaires formelles) à partir d'exemples. Ses composantes « théoriques » aussi bien qu'« empiriques » sont abordées, et autant que possible illustrées par des exemples empruntés au traitement automatique des langues.

L'inférence grammaticale est une branche de l'apprentissage automatique qui étudie comment il est possible d'acquérir par programme une « grammaire » ou un autre modèle formel de langage[3] (un automate fini, par exemple) à partir de données (en général des séquences de symboles, mais aussi éventuellement des arbres d'analyse) qu'il génère (ou pas). Ses principes et ses résultats sont souvent mal connus, parce qu'elle vise l'identification d'*objets symboliques structurés* et ne rentre pas, de ce fait, dans le cadre classique de l'apprentissage statistique. Ce sont d'ailleurs souvent des chercheurs venant de l'informatique théorique, de la « théorie des langages » et de la combinatoire qui ont le plus contribué à son développement. Quant aux spécialistes du traitement automatique des langues, à qui s'adresse prioritairement le livre, s'ils connaissent en général les grammaires formelles, peu sont familiers avec l'inférence grammaticale. Les problèmes auxquels elle s'attaque pourraient pourtant intéresser nombre d'entre eux. L'ouvrage arrive donc à point nommé pour combler un fossé entre différentes communautés. La tâche est délicate, car leurs cultures et leurs références ont largement divergé. Mais le pari du rapprochement vaut d'être tenté.

Le premier chapitre de l'ouvrage introduit les spécificités de l'inférence grammaticale, pour laquelle la lisibilité et l'interprétabilité de la structure cible sont fondamentales. Le champ est extrêmement vaste ; les auteurs reconnaissent avoir dû faire des choix. Ils tiennent toutefois à préserver une certaine variété représentative et s'attacheront donc à présenter à la fois les aspects « formels » (modèles théoriques

3 J'utilise dans ce texte le terme « langage » (qui est une traduction littérale de l'anglais « language ») dans son sens informatique et non linguistique, comme une combinatoire de symboles régulés par des règles explicites.

d'« apprenabilité » et résultats associés) et « empiriques » (comportement de certains algorithmes sur certaines données) du domaine. Mais c'est sur la branche « formelle », historiquement la première et scientifiquement la plus féconde, que l'accent est d'abord mis. Les préliminaires indispensables de la théorie des langages (langages, grammaires, automates finis...) figurent ainsi également dans ce chapitre.

Le suivant est consacré aux modèles d'apprentissage utilisés en inférence grammaticale « formelle » et à quelques résultats associés. Dans ce domaine, en effet, on ne juge pas du succès d'un programme de la même façon qu'en apprentissage statistique. Un modèle d'apprentissage est caractérisé par un « scénario » qui précise les conditions dans lesquelles un algorithme accède à ses données d'entrée (une par une, toutes d'un coup ou *via* des requêtes, par exemple) et par l'ensemble des critères qu'il doit satisfaire. Ces critères peuvent combiner la qualité du résultat produit avec d'autres propriétés (par exemple de convergence, de complexité...). Plusieurs de ces « modèles formels » (modèle de Gold, « PAC learning »...) sont ainsi détaillés. Chacun peut être vu comme une manière de délimiter précisément les frontières de ce qui est « apprenable » par programme. Quelques résultats saillants d'apprenabilité au sens de ces modèles sont ensuite évoqués. Ils concernent les automates finis (déterministes ou non), les expressions régulières, les HMM (*Hidden Markov Models* ou « chaînes de Markov cachées »), les transducteurs et les grammaires algébriques (ou « *context free* ») ainsi que leur version probabiliste, quand cela a un sens. Pour les linguistes ou les spécialistes du TAL qui les découvriront, ces résultats seront à la fois impressionnants et frustrants : impressionnants parce que ce sont des théorèmes mathématiques qui formalisent des notions habituellement laissées à l'intuition, frustrants parce que, comme tous les « modèles », ils fixent des conditions simplificatrices dans lesquelles les situations humaines ont du mal à rentrer.

Le troisième chapitre se concentre sur la famille de langages ayant donné lieu au plus grand nombre de travaux en inférence grammaticale : les langages réguliers (ou rationnels). C'est l'occasion de se pencher de plus près sur le fonctionnement de quelques algorithmes de référence, capables d'identifier un automate fini à partir de séquences qu'il reconnaît ou produit (ce que l'on appelle des « exemples positifs ») et, éventuellement, de séquences qu'il ne reconnaît pas (« exemples négatifs »). C'est l'occasion aussi de s'attarder sur une opération qui joue un rôle fondamental dans la plupart de ces algorithmes : la « fusion d'états ». Fusionner deux états d'un automate fini a pour effet de *généraliser* le langage reconnu. À condition d'être appliqué à bon escient (c'est-à-dire en évitant de *surgénéraliser*), c'est une clé possible du processus *d'induction* que doivent opérer ces algorithmes. Les intérêts et limites de cette opération sont longuement discutés dans ce chapitre. Ils sont notamment illustrés sur la tâche originale de reconnaissance automatique des schémas accentuels (« *stress patterns* » en anglais) mis en œuvre dans deux langues exotiques (le pintupi et le kwak'wala), à partir d'exemples de suites de syllabes accentuées. Cette application, qui fera plaisir aux linguistes, est une des rares, inspirée par les langues naturelles, qui figure dans les premiers chapitres du livre. Les schémas accentuels sont, en effet, caractérisables par un nombre restreint de

symboles et suivent des règles simples (supposées régulières au sens de la théorie des langages), ce qui les rend bien adaptés à ce type d'approche.

Le quatrième et dernier chapitre aborde les langages non réguliers (principalement algébriques ou « *context free* »), dans une perspective d'inférence grammaticale « empirique ». C'est celui qui devrait intéresser le plus les spécialistes du TAL, car il évoque des travaux en lien avec l'analyse syntaxique des langues naturelles. Avant de présenter quelques systèmes, les principes sur lesquels ils sont fondés sont étudiés. Intervient notamment alors le critère fondamental de *substituabilité*, également familier des linguistes, qui permet ici d'identifier les séquences ou les sous-structures pouvant être produites par le même symbole non terminal. Les systèmes eux-mêmes (Emile, ABL, Adios, CCM, DMV, U-DOP) sont souvent peu connus en dehors du domaine, et ont des capacités très variables. Le problème de leur évaluation donne d'ailleurs lieu à des développements détaillés. On peut regretter que des approches plus récentes, comme Mate ou MaltParser, qui visent à apprendre non pas une grammaire formelle, mais directement un analyseur syntaxique (un « *parser* ») à partir d'un corpus arboré, ne soient pas mises en regard de ces systèmes.

Une brève conclusion, enfin, récapitule l'ensemble et signale quelques éléments qui n'ont pu être développés. Il est dommage, par exemple, que certains travaux cherchant à intégrer au sein de l'inférence grammaticale des notions linguistiques (la sémantique, par exemple) n'aient pas pu être évoqués faute de place. Au final, le livre réalise un certain tour de force : il balaie un spectre très large de travaux souvent assez difficiles d'accès, en esquivant les preuves formelles pour se focaliser sur les principes sous-jacents. Ce parti pris lui permet de rester accessible, compact et néanmoins rigoureux. Comme la plupart des ouvrages de sa collection, il se veut plus une porte d'entrée d'un domaine qu'un exposé exhaustif. Avec ses schémas, ses exemples, ses « encadrés », ses résumés en fin de chapitre et sa bibliographie, il jouera parfaitement son rôle pédagogique d'initiateur, aussi bien que d'« aiguilleur » pour les lecteurs curieux d'en savoir plus. Il n'est pas certain que cet effort éditorial suffise pour autant à convertir les praticiens du TAL à l'inférence grammaticale. Mais le paysage scientifique ainsi dévoilé vaut néanmoins le détour.