UNIVERSITÉ DU QUÉBEC À MONTRÉAL

RÉSEAUX RÉCURRENTS ET DÉPISTAGE DE LA DÉPRESSION SUR LES RÉSEAUX SOCIAUX

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
DIEGO MAUPOMÉ

JUILLET 2019

REMERCIEMENTS

J'aimerais remercier toutes les personnes qui se sont intéressées de près ou de loin à mon travail. Je remercie particulièrement mes parents à qui je dédie ce mémoire. Je remercie mon frère, Gabrielle, Karen et Adèle pour leur soutien et leur rôle unique dans ma vie. Je tiens à remercier également mes collègues de laboratoire et amis, spécialement Antoine, Hayda, Marc, Sara et Stéfanny, sans qui les heures au laboratoire auraient été vastement plus ternes et beaucoup moins inspirantes. Finalement, j'aimerais remercier la professeure Marie-Jean Meurs, sans qui ce travail de recherche n'aurait été possible. Elle a su me guider et me motiver à travers ces années et je lui en serai toujours reconnaissant.

TABLE DES MATIÈRES

LISTE DES TABLEAUX vii
LISTE DES FIGURES ix
RÉSUMÉ xi
INTRODUCTION 1
CHAPITRE I ÉTAT DE L'ART 5
1.1 Réseaux récurrents 5
1.1.1 Réseaux neuronaux 5
1.1.2 Réseaux récurrents simples 11
1.1.3 Disparition du gradient et mémoire à long terme 13
1.2 Modèles de langue 14
1.2.1 Introduction 14
1.2.2 Évaluation 16
1.2.3 Modèles de langue récurrents 17
1.3 Dépistage de la dépression par le texte 20
1.3.1 Données 20
1.3.2 Approches 21
CHAPITRE II MODÈLES DE LANGUE RÉCURRENTS MULTIPLICATIFS 25
2.1 Contexte et références 25
2.2 Publication 26
2.2.1 Introduction 26
2.2.2 Recurrent neural networks 27
2.2.3 Multiplicative RNNs 29
2.2.4 Sharing intermediate states 31

2.2.5 Experiments in character-level language modeling 35
2.2.6 Conclusion 39
CHAPITRE III MODÈLES D'ATTENTION POUR L'ESTIMATION DU RISQUE DE DÉPRESSION 41
3.1 Contexte et références 41
3.2 Publication 42
3.2.1 Introduction 43
3.2.2 Dataset 44
3.2.3 Models 45
3.2.4 Related Work 50
3.2.5 Methodology 51
3.2.6 Results 53
3.2.7 Conclusion 54
CONCLUSION 55
GLOSSAIRE 57
ACRONYMES 59
NOTATION 61
RÉFÉRENCES 63

LISTE DES TABLEAUX

Tableau Page

2.1 Test set error on Penn Treebank and parameter counts in character-level language modeling 37

2.2 Test set error on Text8 and parameter counts in character-level language modeling 38

3.1 Statistics on the eRisk 2018 task dataset 45

3.2 Parameter counts, precision, recall and f1-score (%) on the adapted test set for the eRisk 2018 corpus 54

LISTE DES FIGURES

Figure		Page
1.1	Le perceptron	6
1.2	Exemple de prédiction par un perceptron	7
1.3	La fonction logistique	7
1.4	Un perceptron multicouches en activation (biais omis)	8
1.5	Les neurones de la couche suivante font la somme pondérée des activations précédentes	8
1.6	Ils envoient leur activation à la prochaine couche	9
1.7	Les corrections à apporter son propagés la dernière couche	10
1.8	Les neurones intermédiaires récoltent leur part de correction	10
1.9	Les corrections se propagent jusqu'au début du réseau	11
1.10	Un réseau récurrent	12

RÉSUMÉ

La souffrance mentale est un enjeu de santé publique majeur. Or, l'accès aux soins est souvent difficile. Avec la place croissante que les médias sociaux ont dans la vie des personnes de tous les milieux, une piste de solution potentielle est l'utilisation de contenus de médias sociaux afin de faciliter le dépistage et le suivi de troubles de la santé mentale. Il s'agit en effet d'un domaine de recherche en plein essor. Nous nous intéressons en particulier à la détection précoce de la dépression à partir de contenus textuels dans les médias sociaux à l'aide de réseaux neuronaux dits récurrents.

Les réseaux neuronaux récurrents sont de puissants outils de modélisation de données séquentielles. Ils sont spécialement intéressants en traitement automatique de la langue car ils préservent l'ordre des mots et peuvent traiter des écrits de longueur variable. Ces réseaux peuvent toutefois rencontrer des limitations. Par exemple, la façon dont leur état interne est mis à jour peut résulter en des difficultés à quitter des états erronés.

Nous commençons par nous pencher sur une amélioration possible, les *réseaux récurrents multiplicatifs*. Nous examinons leur performance en modélisation de la langue naturelle. Les résultats obtenus sont compétitifs par rapport à l'état de l'art.

Ensuite, nous utilisons ces approches en détection précoce de la dépression. Nous étudions le cas où on dispose de plusieurs écrits indépendants par patient. Nous mettons de l'avant des approches s'appuyant sur les réseaux récurrents pour analyser les écrits d'abord individuellement, puis, en les agrégeant. Les résultats obtenus sont prometteurs, en particulier lorsqu'on intègre le mécanisme de pondération appelé mécanisme d'attention.

MOTS-CLÉS : Réseaux neuronaux, réseaux récurrents, analyse de sentiments, détection du risque, santé mentale, dépistage de la dépression, médias sociaux, apprentissage automatique, traitement automatique du langage naturel.

ARTICLES PRÉSENTÉS DANS CE MÉMOIRE

Multiplicative Models for Recurrent Language Modeling

Diego Maupomé, Marie-Jean Meurs

Proceedings of The 20th International Conference on Computational Linguistics and Intelligent Text Processing, CICLing 2019

Springer Nature, Lecture Notes in Conputer Science (LNCS)

Inter and Intra Document Attention for Depression Risk Assessment

Diego Maupomé, Marc Queudot, Marie-Jean Meurs

Proceedings of The 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019

Springer Nature, LNCS Lecture Notes in Artificial Intelligence #11489

INTRODUCTION

En 2015, 5 millions de Canadiens âgés de plus de 15 ans ont éprouvé un besoin de soins en santé mentale ; deux millions ont rapporté que leurs besoins n'ont pas été satisfaits (Statistics Canada, 2017). Or, les troubles de la santé mentale constituent une des premières causes d'invalidité (Lim *et al.*, 2008). La souffrance mentale peut réduire l'espérance de 7 à 24 ans (Chesney *et al.*, 2014). L'accès a un diagnostic adéquat est difficile pour différentes raisons telles que la stigmatisation (Rodrigues *et al.*, 2014) et la prévalence de diagnostics erronés (Vermani *et al.*, 2011). La détection et le traitement des troubles en santé mentale constituent donc un enjeu de santé publique majeur.

Pour faire face à ce défi, il existe un intérêt naissant pour la détection automatique du risque en santé mentale en utilisant les techniques d'Intelligence Artificielle (IA). Nous nous intéressons en particulier à la détection précoce de la dépression à partir de la langue écrite dans l'intention de l'appliquer aux médias sociaux. Ce champ d'intérêt relève du domaine du Traitement Automatique de la Langue Naturelle (TALN), plus particulièrement, de l'**analyse de sentiments**. Historiquement, le domaine a été centré autour de l'analyse d'opinions, aussi appelée analyse de polarité. Une tâche d'analyse de polarité tentera, par exemple, de déterminer le *sentiment* général d'un document, soit positif ou négatif. Plus précisément, il sera question de déterminer une fonction qui associe un document à la catégorie – appelée la *classe* – qui lui correspond selon l'annotation dont on dispose. Cette fonction peut être construite à la main ou apprise, c'est-à-dire, déterminée à partir d'un ensemble d'observations. Par exemple, on peut s'intéresser

au décompte de certains mots dits «porteurs de sentiment», tels que «bon, mauvais, excellent, terrible», et décider de la polarité du document en suivant certains critères préétablis. Ceci constituera une approche manuelle. On pourrait considérer plus de mots, par exemple, les mots apparaissant dans une certaine plage de fréquence dans le corpus. On pourra tenter d'inférer à partir des données comment les fréquences et concomitances éventuelles de mots peuvent aider à faire une prédiction sur la polarité d'un document. Une telle fonction serait donc apprise et l'approche relèverait du domaine de l'apprentissage automatique. Évidemment, si l'on veut considérer des facteurs plus complexes tels l'ordre des mots, la fonction à apprendre sera elle aussi plus complexe et inférer de la fonction finale à partir d'observations sera plus difficile.

Les deux fonctions, celle construite à la main et celle qu'on aura apprise, seront des algorithmes d'analyse de sentiments. Dans un cas comme dans l'autre, la justesse, la *performance*, des approches sera quantifiée à l'aide d'une métrique et d'un ensemble d'exemples dits de test. Le choix de la métrique utilisée est tout aussi important que le choix de la fonction ou, le cas échéant, de la méthode d'inférence. Certaines métriques peuvent donner des résultats qui seront trompeurs si l'on n'est pas attentif. Par exemple, un algorithme qui classe tout document comme étant positif peut atteindre, sur un ensemble de données composé à 90% de documents positifs, un taux d'erreur de 10%, ce qui peut sembler un bon résultat hors contexte. Dans tous les cas, les métriques seront toujours réductrices : il est simplement question de faire un choix éclairé sur ces réductions.

D'un point de vue technique, l'intérêt accentué pour l'analyse de polarité découle principalement de deux aspects de cette tâche. D'une part, les avancées dans ces tâches ont d'énormes conséquences commerciales, permettant aux entreprises de percevoir et quantifier la popularité de leurs produits et services. D'autre part, puisqu'il s'agit effectivement d'une polarité, la distinction nette entre les deux

pôles simplifie l'abord de la tâche et rend très accessible l'annotation. En effet, si le problème de classification était plus complexe, il serait indispensable d'élaborer des hypothèses plus sophistiquées. Il serait également plus difficile d'annoter des documents.

C'est le cas de la détection de troubles de la santé mentale à partir de ressources textuelles. Il s'agit d'une tâche naissante en TALN qui semble nécessiter des approches sophistiquées pour arriver à des bons résultats. Or, ces modèles plus puissants seront souvent lourds. Nous nous intéressons donc à la question de recherche suivante :

Comment élaborer des approches légères et efficaces de détection de la dépression en s'appuyant sur les résultats en modélisation de la langue.

Le présent mémoire cherche donc à répondre à cette question. Il est structuré comme suit.

Dans l'état de l'art proposé au Chapitre 1, nous évoquons quelques unes des difficultés qui surgissent dans la récolte de données et leur utilisation à des fins de prédictions, ainsi que des difficultés propres à l'outil principal utilisé au courant de notre recherche, les réseaux récurrents. Nous présentons au Chapitre 2 une publication parue à la conférence CICLing 2019 sur nos travaux de recherche sur les réseaux récurrents multiplicatifs et leur performance en modélisation de langues. Le Chapitre 3 est composé d'une publication parue à la conférence Canadian AI 2019 sur la détection précoce de la dépression et différentes méthodes d'agrégation des écrits d'usagers de médias sociaux. Finalement, nous concluons ce mémoire avec des perspectives sur la portée des présentes recherches sur la détection de la dépression, de la suicidalité et d'autres troubles comportementaux ainsi que des pistes de travail futur.

CHAPITRE I

ÉTAT DE L'ART

Nous présentons dans ce chapitre un état de la littérature pertinente dans les champs d'intérêt que nous avons étudiés. Nous entamons par une brève introduction aux réseaux neuronaux et plus particulièrement aux réseaux récurrents. Nous introduisons également les modèles de langue ainsi que les modèles de langue récurrents et discutons des considérations pratiques qui en engendrent des améliorations structurales. Finalement, nous présentons une application au domaine du dépistage de la dépression dans le texte, la récolte et annotation de leurs données ainsi que les approches prises en prédiction et analyse.

1.1 Réseaux récurrents

1.1.1 Réseaux neuronaux

Les réseaux neuronaux constituent une famille d'algorithmes utilisés dans l'approximation de fonctions complexes, en particulier en apprentissage automatique.

Le premier réseau neuronal est un classifieur binaire appelé le perceptron (Rosenblatt, 1958), illustré à la Figure 1.1. Il comporte un paramètre, également appelé poids, par composante des données ainsi qu'un paramètre de biais. Pour une observation donnée, il fera l'addition du biais des composantes de l'observa-

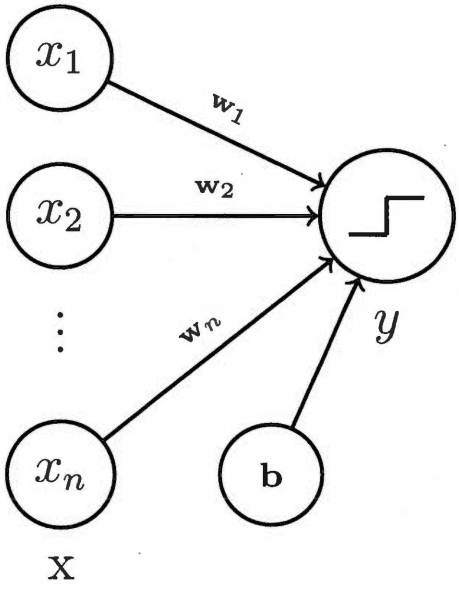

Figure 1.1 Le perceptron

tion pondérées par les poids. Cette somme doit dépasser un seuil fixe pour donner une prédiction positive.

$$\text{perceptron(x)} = \begin{cases} 1 & \text{si} \quad \mathbf{w}^\top \mathrm{x} + \mathbf{b} \geq 1 \\ 0 & \text{sinon} \end{cases} \tag{1.1}$$

La Figure 1.2 montre un exemple de prédiction par un perceptron.

Il est à noter que la nature discontinue de ce seuil rend difficile l'inférence des paramètres du perceptron. Pour cette raison, on utilisera plutôt des **fonctions d'activation**, qui sont différentiables partout. Un exemple courant est la fonction logistique,

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{1.2}$$

qui agit comme le seuil présenté précédemment mais de façon plus lisse, tel qu'illustré par sa courbe à la Figure 1.3. L'utilisation de **fonctions d'activation** permet de rendre différentiable l'ensemble de la fonction calculée par le perceptron, ce qui permet son inférence par **descente de gradient** (Cauchy, 1847)

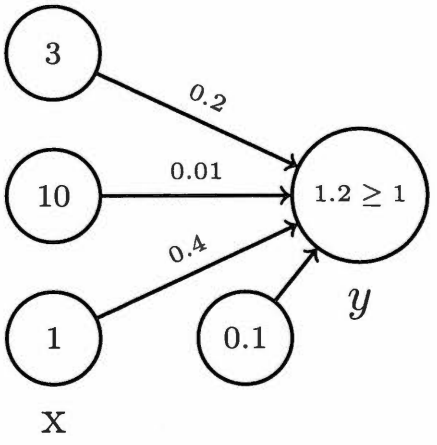

Figure 1.2 Exemple de prédiction par un perceptron

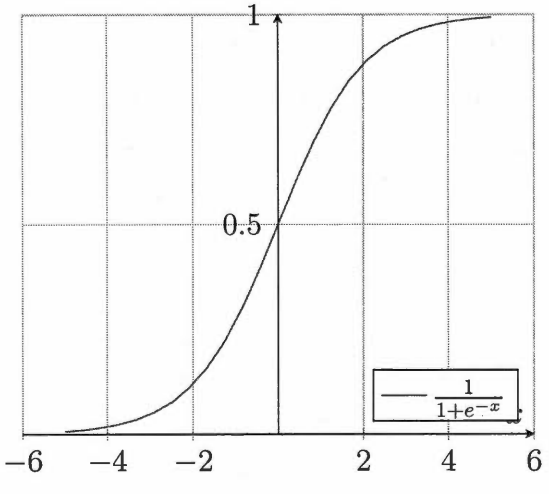

Figure 1.3 La fonction logistique

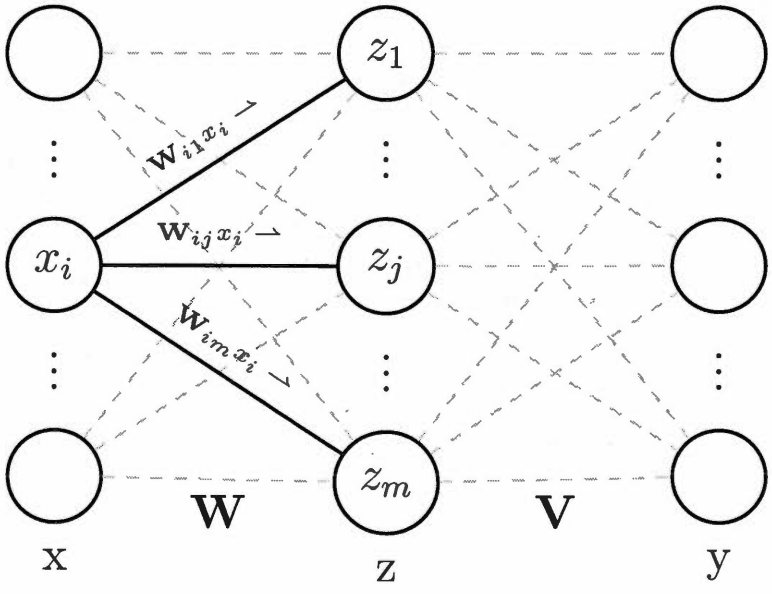

Figure 1.4 Un perceptron multicouches en activation (biais omis)

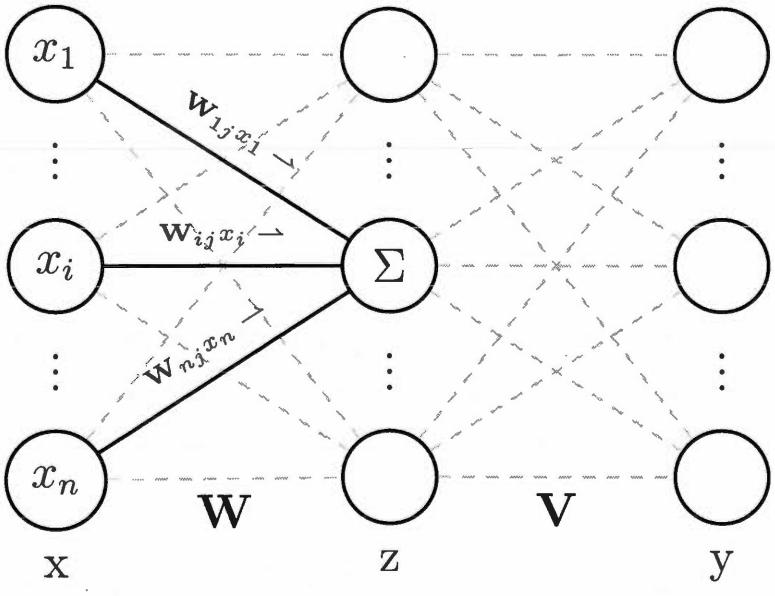

Figure 1.5 Les neurones de la couche suivante font la somme pondérée des activations précédentes

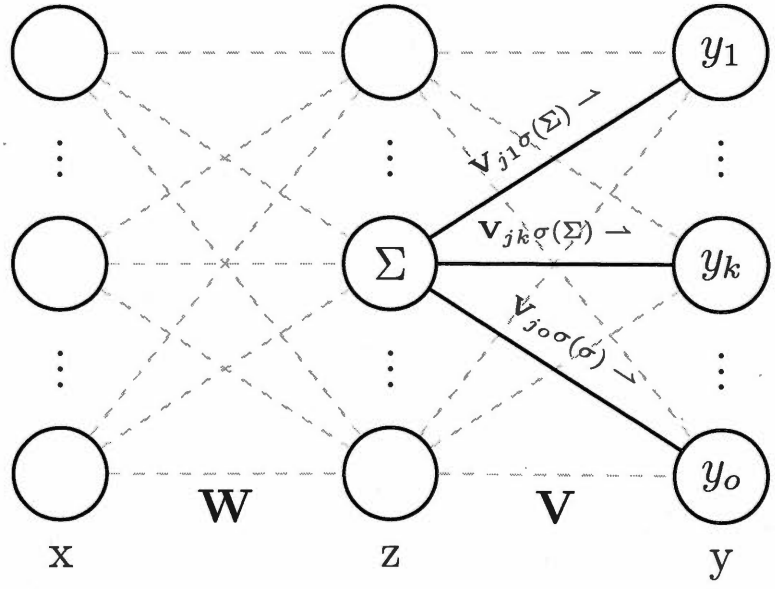

Figure 1.6 Ils envoient leur activation à la prochaine couche

Il s'avère que plusieurs de ces perceptrons, ou neurones, peuvent être arrangés en couches dans des réseaux complexes, tels celui à la Figure 1.4. Chaque neurone fait état des contributions des neurones qui le précèdent et envoie son activation à son tour aux neurones qui le suivent. Ce fonctionnement est illustré par les Figures 1.4, 1.5 et 1.6.

Afin de rendre l'inférence des paramètres de réseaux complexes plus efficace, on utilise l'algorithme de **rétropropagation** (Rumelhart *et al.*, 1988), basé sur la règle de dérivation en chaîne. Puisque les gradients des paramètres en début de réseau dépendent des gradients des paramètres qui les suivent, on calculera les gradients en sens inverse du réseau. Les Figures 1.7, 1.8 et 1.9 illustrent ce principe.

De plus, il se trouve que les réseaux neuronaux à une seule couche intermédiaire sont capables d'approximation universelle (Hornik, 1991). Cela veut dire que pour toute fonction calculable, il est possible de l'approcher de façon arbitrairement juste avec un réseau neuronal à une couche cachée suffisamment grande. Cette

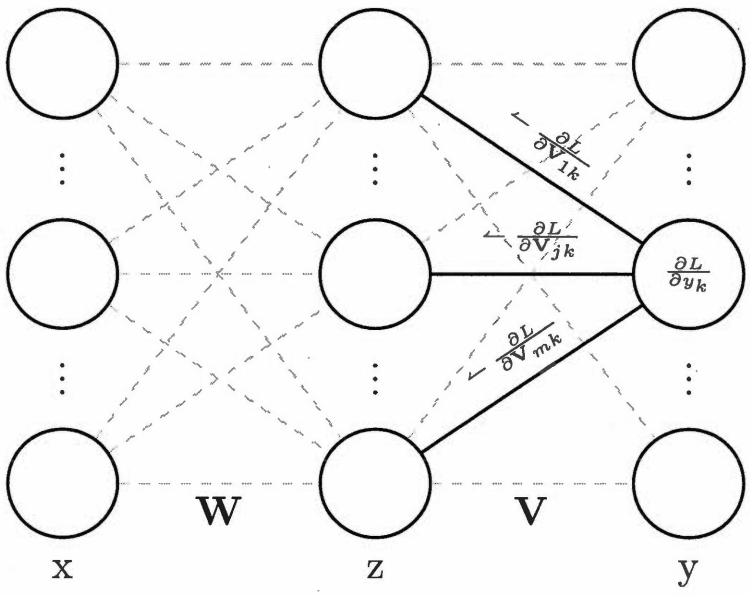

Figure 1.7 Les corrections à apporter son propagés la dernière couche

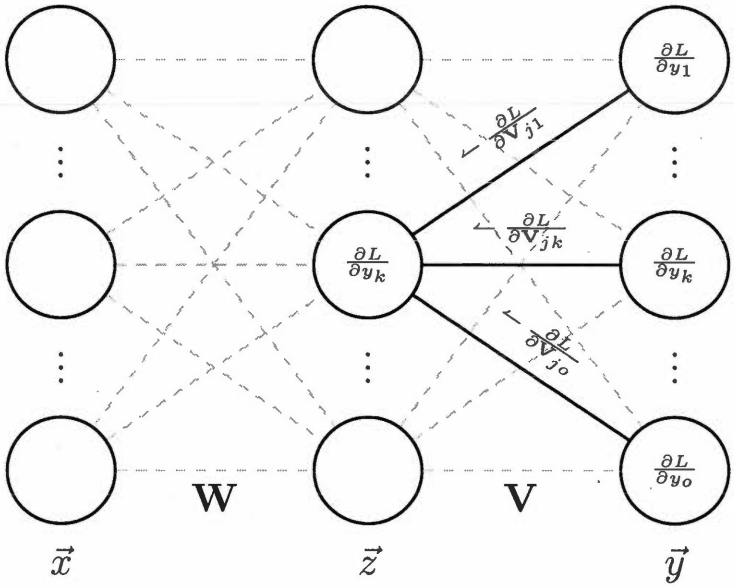

Figure 1.8 Les neurones intermédiaires récoltent leur part de correction

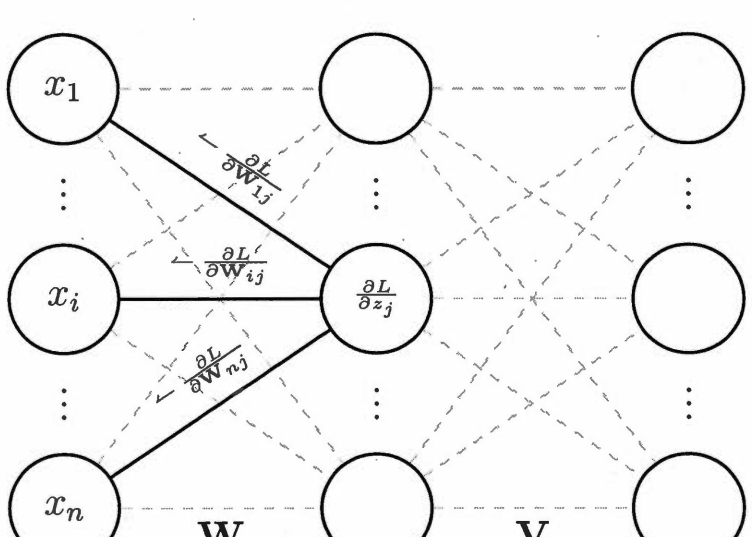

Figure 1.9 Les corrections se propagent jusqu'au début du réseau

notion est toutefois sujette à des interprétations inexactes. Ce n'est pas tout réseau qui peut approcher n'importe quelle fonction. Il n'est pas question non plus de la capacité à inférer une fonction calculable quelconque.

En pratique, certains réseaux seront mieux adaptés à traiter certains types de données. Par exemple, lorsqu'on traite des données séquentielles, un réseau dont toutes les opérations sont de taille fixe pourra difficilement appréhender leur longueur variable. Il est préférable de traiter ces données par des réseaux dits récurrents, qui seront présentés à la prochaine sous-section.

1.1.2 Réseaux récurrents simples

Les Réseaux Récurrents (RNN) (Jordan, 1986) sont un type de réseaux neuronaux spécialement adaptés à faire des prédictions sur des données séquentielles. Ceci est accompli en combinant chaque composante temporelle des données à un vec-

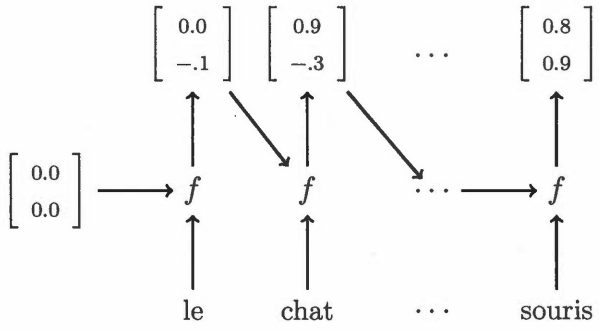

Figure 1.10 Un réseau récurrent

teur appelé l'**état caché** du RNN, qui doit résumer les composantes temporelles précedentes. Soit une observation séquentielle, $x_1, \ldots, x_T$, indicée par t. Un RNN sera donc décrit ainsi,

$$h_t = f(h_{t-1}, x_t) \tag{1.3}$$

$$h_0 = 0, \tag{1.4}$$

où h_t est la valeur de l'**état caché** après avoir lu l'entrée x_t. Un exemple est présenté à la Figure 1.10. f est dénommée la **fonction de transition** du RNN. C'est elle qui s'occupe de produire la nouvelle valeur de l'état caché. Ce seront donc ses paramètres qui seront modifiés pour bien modéliser les séquences à l'étude. La fonction f pourra être plus ou moins complexe selon les besoins du problème. On peut s'intéresser également aux valeurs de l'état caché à différents moments selon la tâche à réaliser. Par exemple, si l'on cherche à classifier des séquences, on ne pourrait s'intéresser qu'à la dernière valeur, c'est-à-dire celle calculée après avoir lu tous les éléments de la séquence.

Un choix particulier de la forme de f est aussi dénommé l'**architecture** du RNN. Une architecture est donc une famille de fonctions de cette forme particulière. Le mot *modèle* est, lui, réservé à la fonction apprise pour une tâche ou un ensemble de données particuliers : un choix particulier dans cette famille. Différentes architec-

tures de RNN sont proposées constamment en vue d'obtenir des améliorations ou des généralisations de modèles précédents (Yang *et al.*, 2017; Wu *et al.*, 2016; Sutskever *et al.*, 2011). L'**architecture** la plus simple et, historiquement, la première proposée (Jordan, 1986) est la suivante :

$$f(h_{t-1}, x_t) = \tanh(\mathbf{U}x_t + \mathbf{W}h_{t-1} + \mathbf{b}). \tag{1.5}$$

$\mathbf{U}$ et $\mathbf{W}$ sont des matrices de paramètres appris, b est le vecteur de biais appris. Avec un système dynamique non-linéaire et un **état caché** de dimension arbitraire, on peut théoriquement modéliser des données séquentielles arbitrairement complexes (Siegelmann et Sontag, 1991). Toutefois, plusieurs considérations numériques entravent cette promesse. Par exemple, un problème bien connu qui se présente lors de l'inférence de RNN par descente de gradient est celui de la **disparition ou explosion du gradient** (Bengio *et al.*, 1994).

1.1.3 Disparition du gradient et mémoire à long terme

Lorsqu'on rétropropage le gradient pour une prédiction sur une donnée séquentielle, on devra corriger les poids qui ont contribué à la plus récente valeur de l'**état caché** tout comme ceux qui ont contribué à ses valeurs précédentes. On dénomme cette idée rétropropagation à travers le temps (BPTT) car la rétropropagation devra reculer dans le temps pour faire les corrections appropriées dès le début de l'observation. Soit donc deux étapes de temps quelconques $i, j : i < j$. On peut facilement voir que le gradient de l'erreur par rapport à h_i dépendra de h_j. Toutefois puisque f demeure la même à travers toutes les étapes de temps, on aura $\mathbf{W}$ multipliant h_j à chaque étape de temps, ce qui mènera à l'exponentiation de $\mathbf{W}$, $\mathbf{W}^{j-1}$. Lorsque i est beaucoup plus petit que j et à cause de la nature exponentielle de ce calcul, le terme en question deviendra très grand (explosera) lorsque la valeur propre dominante de $\mathbf{W}$ est plus grande que 1 ou très petit (disparaîtra)

lorsqu'elle est plus petite que 1. Alors que dans un contexte de précision infinie, on pourrait argumenter que ces valeurs très grandes ou très petites reflètent la nature des ajustements à faire, en contexte numérique, ces ajustements pourront tout simplement ne pas avoir lieu. Si le gradient disparaît avant de s'être propagé jusqu'au début du réseau, les ajustements à faire en début de réseau seront nuls.

En particulier, en langue naturelle, il existe des dépendances à long terme, c'est-à-dire que des mots qui font référence les uns aux autres se trouvent à plusieurs mots d'écart. C'est dans de tels contextes que la disparition du gradient pourra nuire à l'apprentissage de la tâche, puisque l'information pertinente à un mot ne se propagera pas nécessairement à un mot lointain. Plusieurs solutions architecturales à ce problème ont été proposées. Les plus connues sont celles employant des mécanismes dit de **portes**, telles les Long Short-Term Memory networks (LSTM) (Hochreiter et Schmidhuber, 1997). Ces réseaux permettent d'apprendre ce genre de dépendances en tâchant les portes (dont les valeurs sont elles-mêmes apprises) de gérer la taille du gradient.

Ayant expliqué le principe de fonctionnement ainsi que quelques architectures de RNN propices au traitement de langues, nous abordons à la prochaine section les modèles de langue et leurs versions récurrentes.

1.2 Modèles de langue

1.2.1 Introduction

Un **modèle de langue** est une distribution de probabilité sur un ensemble de productions langagières. Il s'agit d'une manière de formaliser et relaxer la notion de langue naturelle. Soit un ensemble de mots, V, dénommé le vocabulaire. Une phrase composée de mots de V s'écrit comme une séquence $x_1, \ldots, x_T$. Nous avons

donc que $x_1, \ldots, x_T \in V^*$, où V^* est l'ensemble de toutes les séquences de mots possibles. Un **modèle de langue** sur V^*, f, assignera une masse de probabilité à toute phrase, $f : V^* \to [0, 1]$. Le modèle doit être donc capable de lire une séquence, aussi incohérente soit elle, et déterminer à quel point elle est probable dans la langue en question. Évidemment, le but est que le modèle assigne une probabilité plus élevée aux séquences qui sont vraisemblables et une probabilité faible voire nulle à celles qui sont improbables ou impossibles.

Bien que rien n'empêche l'élaboration manuelle d'un modèle de langue, ceux-ci seront le plus souvent inférés à partir d'un corpus. Il appartient au modèle de langue d'attribuer une probabilité à toute séquence de mots. La nature exacte du phénomène linguistique n'étant pas connue et V^* étant vraisemblablement très grand, le modèle de langue passera par des simplifications choisies soigneusement selon les ressources dont on dispose et les besoins du problème. Par exemple, un modèle unigramme maintiendra une table de probabilité des mots individuels. La masse de probabilité d'une phrase $x_1, \ldots, x_T$ sera approchée de la façon suivante :

$$p(x_1, \ldots, x_T) \approx \prod_{i=1}^{T} p(x_i) \tag{1.6}$$

Il suffit donc simplement de maintenir une table de probabilité marginale des mots et multiplier ces probabilités pour élaborer une prédiction pour une phrase. Un tel modèle est simple d'implémentation et rapide d'exécution. Néanmoins, il sera plutôt naïf car l'intuition veut que les mots constituant une phrase ne sont pas indépendants les uns des autres : observer un certain mot devrait affecter la probabilité d'en observer un autre. De plus, l'ordre des mots n'est pas pris en compte.

On pourrait tenter d'améliorer à cette situation en prenant des paires ou triplets de mots, appelés respectivement bigrammes et trigrammes. Toutefois, plus les parties de phrase que l'on considère deviennent longues, plus elles deviendront rares.

L'espace de représentation deviendra creux, ce qui menera à des modèles surajustés, un phénomène communément appelé **le fléau de la dimension** (Chen, 2009).

1.2.2 Évaluation

Puisqu'un modèle de langue est une distribution de probabilité sur les productions langagières possibles, on voudrait l'évaluer en le comparant à la vraie distribution, p, utilisant l'**entropie croisée**.

$$H(p, \hat{p}) = -\sum_{x} p(x) \log \hat{p}(x) \tag{1.7}$$

Toutefois, la vraie distribution n'étant pas connue, la distribution estimée sera plutôt comparée à un corpus de test. En raison de la nature **ergodique** de la langue, on peut s'approcher arbitrairement de la vraie entropie croisée avec un corpus assez grand (Manning *et al.*, 1999) :

$$H(p, \hat{p}) \approx -\frac{1}{n} \sum_{x} \log \hat{p}(x), \tag{1.8}$$

où n est la taille du corpus. Cette approximation repose aussi sur la supposition de stationnarité de la langue. En réalité, la langue évolue à travers le temps. On peut toutefois considérer cette approche valide pour un court moment dans l'histoire de la langue ou pour un corpus particulier (Manning *et al.*, 1999).

L'unité de mesure de l'entropie croisée est le bit ou le nat selon la base du logarithme utilisé. Puisqu'elle est normalisée par la taille du corpus, il sera question de Bits par Caractère (BPC) ou par mot. Cette mesure est intuitive lorsqu'on découpe le corpus en caractères puisque leur encodage, quel qu'il soit, est de taille fixe. Cependant, les mots étant de taille variable, cette notion est moins intuitive lorsqu'on découpe en mots. On parlera alors de **perplexité** par mot, dénotée par

$b^{H(p,\hat{p})}$,

$$b^{H(p,\hat{p})} = b^{-\sum_x p(x) \log_b \hat{p}(x)} \tag{1.9}$$

où b est la base du log choisi pour calculer l'entropie croisée (souvent 2 ou la base naturelle). Intuitivement, une **perplexité** de p indique que les prédictions du modèle sont aussi justes que s'il prenait chaque décision au hasard parmi p options en moyenne. Il faut donc regarder cette quantité en comparaison avec le nombre moyen d'options qu'un modèle a à chaque décision. Par exemple, il peut sembler qu'un modèle de langue avec une **perplexité** par mot de 50 ne soit pas performant. Toutefois, si le vocabulaire du modèle est de 20k mots, cela veut dire qu'il arrive à écarter toutes les possibilités excepté 50 à chaque mot. Il faut aussi considérer qu'il s'agit d'une moyenne à travers le corpus de test. Les erreurs ou indécisions du modèle ne seront jamais distribuées uniformément à travers le corpus.

1.2.3 Modèles de langue récurrents

Certains modèles de langue vont chercher à combiner leurs prédictions pour des sous-séquences afin d'arriver à une prédiction pour la séquences que celles-ci constituent. Comme exprimé par l'Équation 1.6, le modèle unigramme fera une agrégation approximative. On peut cependant chercher à agréger de façon exacte, comme le font les **modèles de langue récurrents**. Les **modèles de langue récurrents** sont des **modèles de langue** s'appuyant sur un RNN pour produire une prédiction de probabilité attribuée à la séquence de mots observée. Par la règle de chaîne de probabilité, nous avons que

$$p(x_1, \ldots, x_T) = \prod_{i=1}^{T} p(x_i | x_1, \ldots, x_{i-1}). \tag{1.10}$$

S'il nous est possible de parcourir les mots d'un document dans l'ordre et si nous pouvons calculer la probabilité de chaque mot sachant les mots précédents, alors

la probabilité de la séquence entière pourrait être calculée de façon exacte. Malheureusement, l'explosion combinatoire (Krippendorff, 1986) rend cela inenvisageable. Toutefois, les **modèles de langue récurrents** vont chercher à approcher la probabilité conditionnelle du mot présent étant donné les mots précédents en maintenant une synthèse de ceux-ci. Cette synthèse est l'**état caché** du RNN.

Tel que mentionné à la section 2.2.2, les RNN peuvent rencontrer des difficultés pratiques. Nous avons présenté l'une des plus connues, le **disparition ou explosion du gradient**. Une difficulté rencontrée lors de la modélisation récurrente de langues est celle de rectifier des valeurs erronées de l'état caché. Supposons que nous avons un RNN simple dont la fonction de transition est décrite par l'Équation 1.5. Si le RNN doit apprendre à préserver certaines composantes de l'état caché à travers le temps afin de faire voyager l'information sur de grandes distances], la partie du terme $\mathbf{W}h_{t-1}$ correspondant à cette région devra être grande pour rester invariante à son homologue dans $\mathbf{U}x_t$. Or, si le réseau a eu tort de s'engager dans des telles valeurs pour cette région de l'état caché, il sera difficile pour l'entrée de réajuster la valeur de l'état caché.

Bien entendu, si on augmente la taille de l'état caché, cette situation sera vraisemblablement plus rare car les composantes de l'état caché pourront se spécialiser. Toutefois, la croissance du nombre de paramètres est de l'ordre du carré de la dimension de l'état caché. Il s'avère que le gain en performance en modélisation de langues ne suit pas cette croissance (Sutskever *et al.*, 2011). Une alternative à l'agrandissement de l'état caché est d'utiliser les mécanismes de **portes**, la mémoire à long terme ayant un effet stabilisateur (Graves, 2013).

Néanmoins, le problème est plus fondamental : la nature additive de la fonction de transition en est en cause. Pour produire une nouvelle valeur d'état caché, nous avons besoin d'un accord entre l'entrée et l'état caché. Une addition suggère

une disjonction, c'est-à-dire qu'il suffit de l'accord d'une opérande pour franchir le seuil de la **fonction d'activation**. En revanche, une conjonction aurait besoin de l'accord des deux termes pour faire cette transition, ce qui suggère le besoin d'une interaction multiplicative (Sutskever *et al.*, 2011). Si l'on lit le morphème *appren* suivi du caractère *a*, nous savons que *nt* sont probablement les caractères à suivre. C'est grâce à la conjonction de ces deux informations que nous savons qu'il s'agit du gérondif. On voudrait donc que le calcul du prochain état reflète cela.

Donc, il existe plusieurs **architectures** de RNN faisant appel aux interactions multiplicatives. Elles font souvent intervenir un terme de la forme $\mathbf{U}x_t * \mathbf{W}h_{t-1}$, appelé terme de second ordre, où $*$ représente le produit de Hadamard.

Par exemple, les multiplicative RNN (mRNN) (Sutskever *et al.*, 2011) remplacent le terme $\mathbf{W}h_{t-1}$ par le terme $\mathbf{V}m_t$, où m_t est le terme de second ordre, appelé **état intermédiaire**. En revanche, l'architecture appelée RNN de second ordre (Goudreau *et al.*, 1994) utilise une projection bilinéaire **de l'entrée** et l'état caché pour mettre à jour les composantes de ce dernier, $h_t^{(i)} = \tanh(h_{t-1}\boldsymbol{W}^{(i)}x_t)$.

Une troisième architecture proche de ces deux dernières est le RNN à Intégration Multiplicative (MI-RNN) (Wu *et al.*, 2016). En remplaçant l'addition par le produit de Hadamard dans l'Équation 1.5, on obtient

$$f(h_{t-1}, x_t) = \tanh(\mathbf{U}x_t * \mathbf{W}h_{t-1} + \mathbf{b}). \tag{1.11}$$

Cette modification introduit un terme de second ordre sans ajouter de paramètres. Il existe également une formulation plus générale (Wu *et al.*, 2016),

$$f(h_{t-1}, x_t) = \tanh(\alpha * \mathbf{U}x_t * \mathbf{W}h_{t-1} + \beta_1 * \mathbf{U}x_t + \beta_2 * \mathbf{W}h_{t-1} + \mathbf{b}). \tag{1.12}$$

où $\alpha, \beta_{1,2}$ sont des biais multiplicatifs appris Cette formulation, bien qu'elle introduise plus de paramètres, a également l'avantage de préserver des termes de

premier ordre, à l'instar des mRNN.

Ces trois architectures sont proches, mais elles ne s'équivalent pas. Elles ne comportent pas la même charge paramétrique toutes choses étant égales par ailleurs. Un intérêt particulier des mRNN est que l'état intermédiaire est de dimension arbitraire. Ceci permet de séparer la taille de l'espace où l'interaction multiplicative a lieu de la taille de l'espace de l'état caché. Il est possible d'atteindre des résultats compétitifs en modélisation de langue avec mRNN et MI-RNN. Il est également possible d'ajouter des mécanismes de portes à ces architectures, ce qui est présenté au Chapitre 2.

Dans le contexte de notre projet, nous nous sommes naturellement intéressés à l'aptitude de ces différentes architectures à détecter la dépression dans le texte. Nous abordons cette tâche à la prochaine section en présentant un aperçu de la littérature du domaine.

1.3 Dépistage de la dépression par le texte

1.3.1 Données

Des nombreux travaux existent visant à détecter la dépression depuis des ressources textuelles produites par le patient. Un des premiers jeux de données produits pour cette tâche (Rude *et al.*, 2004) est composé d'essais écrits par des étudiants au sujet de leur expérience universitaire par flux de conscience. L'état mental des étudiants a été évalué par le Beck Depression Inventory (Beck *et al.*, 1961).

Avec l'avènement des médias sociaux, le nombres d'études à ce sujet a crû, ces premiers étant non seulement une source plus vaste de données pour l'inférence mais un milieu où l'application des connaissances acquises serait plus efficace

dans le combat de cette maladie. Ces études s'intéressent souvent à l'identification précoce de la dépression.

Une des principales difficultés de ce champ de recherche est la collecte de données cliniquement fondées. Un diagnostic formel d'épisode dépressif majeur requiert un entretien avec un clinicien (American Psychiatric Association, 2013), ce qui représente des coûts et du temps supplémentaires. Souvent, les données seront recueillies en cherchant des admissions de diagnostic par un clinicien (Losada *et al.*, 2018; Coppersmith *et al.*, 2015). En cherchant, par exemple des expressions de la forme «mon médecin m'a diagnostiqué une dépression» et prélevant les écrits précédent celles-ci, on peut présumer détenir une vérité proche de la vérité clinique. On obtient cependant des faux positifs et, vraisemblablement, un bon nombre de faux négatifs, en particulier si les usagers négatifs sont prélevés dans les mêmes forums de discussion que les positifs. Il est également possible de recruter des usagers de médias sociaux et les évaluer par des outils dont se servent les cliniciens (De Choudhury *et al.*, 2013). Une troisième option plus lointaine de la vérité clinique est l'annotation experte à même les données textuelles. Ce genre d'étude sera plus souvent mené sur des phénomènes comportementaux plus larges et moins cliniquement cernés tels la suicidalité ou l'automutilation (Coppersmith *et al.*, 2015; Shing *et al.*, 2018). Conséquemment, les annotateurs ne sont pas tenus d'être des cliniciens mais peuvent être des intervenants sociaux ou des modérateurs de forums Internet.

1.3.2 Approches

D'un point de vue technique, les approches utilisées couvrent l'ensemble de celles existant en classification de documents. On retrouve par exemple l'usage d'analyse de style linguistique (Rude *et al.*, 2004), ou des approches reposant sur des

dictionnaires ou de catégories psychologiques de mots (Pennebaker *et al.*, 2001; Tausczik et Pennebaker, 2010; De Choudhury *et al.*, 2013). Dans le même ordre d'idées, il peut être aussi pertinent d'utiliser des **modèles de sujets** tels Allocation de Dirichlet Latente (LDA) afin de représenter les usagers par la fréquence à laquelle ils abordent certains sujets dans leurs écrits (Resnik *et al.*, 2013, 2015). Ces approches reposant sur les mots et leurs catégories ont l'énorme avantage d'être plus interprétables et intuitives. Conséquemment, ces études seront menées par la communauté de recherche en psychologie, psycholinguistique ou linguistique computationnelle afin d'améliorer la compréhension des pathologies. Or, les résultats en prédiction seront plus modestes.

À l'inverse, les approches plus sophistiquées produiront de meilleurs prédictions mais moins de possibilités d'analyse. Elles reposent souvent sur des réseaux neuronaux. Par exemple, Yates *et al.* (2017) ont affaire à des usagers représentés par un ensemble d'écrits. Ils utilisent des couches convolutives pour traiter chaque écrit individuellement. Ces représentations sont ensuite concaténées et données en entrée à un réseau neuronal simple qui fournit une prédiction pour l'usager, représenté comme un ensemble d'écrits. À l'inverse, Shing *et al.* (2018) concatènent d'abord les écrits et chargent un réseau convolutif de déterminer les dimensions caractéristiques des mots les plus saillantes. Ces aspects saillants sont agrégés à travers le temps de façon semblable à Kim (2014) pour fournir une prédiction sur l'usager.

Du côté des RNN, Ive *et al.* (2018) classifient des documents pouvant exhiber des troubles de la santé mentale (Gkotsis *et al.*, 2017). Ils utilisent un RNN hiérarchique (Yang *et al.*, 2016) pour traiter les documents. Cette approche combine des RNN au mécanisme attentionnel (Bahdanau *et al.*, 2014) pour traiter d'abord les énoncés qui constituent un document pour ensuite les pondérer et les agréger. Cette approche parvient à des bons résultats en moyenne à travers les diffé-

rentes catégories de troubles comportementaux. Elle a également l'avantage d'être plus interprétable grâce au mécanisme d'attention, qui offre une pondération des phrases les plus importantes du document. Nous présentons, au Chapitre 3, une publication mettant de l'avant plusieurs approches semblables à cette dernière.

Nous avons donc introduit dans ce premier chapitre un aperçu des réseaux récurrents, les modèles de langue et la détection automatique de la dépression. Les deux chapitres suivants présentent des publications mettant de l'avant ces concepts.

CHAPITRE II

MODÈLES DE LANGUE RÉCURRENTS MULTIPLICATIFS

2.1 Contexte et références

Nous avons mentionné à la sous-section 1.2.3 qu'il est possible d'ajouter des interactions multiplicatives à la **fonction de transition** d'un RNN afin de le rendre plus expressif. De plus, il est possible combiner ces modifications à des mécanismes de **portes**. Dans la publication qui suit, nous avons exploré certaines de ces combinaisons. Nous avons testé leur performance en modélisation récurrente de la langue niveau caractère sur deux jeux de données largement répandus dans la littérature. Les résultats obtenus sont à l'état de l'art ou meilleurs. L'article a été présenté par l'auteur de ce mémoire en avril 2019 à la conférence *CICLing 2019, The 20th International Conference on Computational Linguistics and Intelligent Text Processing*, à La Rochelle, France, et sera publié dans les actes de la conférence (Springer, Lecture Notes in Conputer Science LNCS).

L'auteur de ce mémoire a formulé l'hypothèse de recherche ainsi que conçu et implémenté les architectures présentées. Il a également choisit et mené les expériences conformément à l'état de l'art. Finalement, il a effectué la majeure partie de la rédaction de l'article.

Reproductibilité. Le code source associé aux travaux présentés est disponible publiquement ici : `https://gitlab.ikb.info.uqam.ca/diego/cicling2019`

2.2 Publication

Multiplicative Models for Recurrent Language Modeling

Diego Maupomé, Marie-Jean Meurs

Université du Québec à Montréal

Abstract

Recently, there has been interest in multiplicative recurrent neural networks for language modeling. Indeed, simple Recurrent Neural Networks (RNNs) encounter difficulties recovering from past mistakes when generating sequences due to high correlation between hidden states. These challenges can be mitigated by integrating second-order terms in the hidden-state update. One such model, multiplicative Long Short-Term Memory (mLSTM) is particularly interesting in its original formulation because of the sharing of its second-order term, referred to as the intermediate state. We explore these architectural improvements by introducing new models and testing them on character-level language modeling tasks. This allows us to establish the relevance of shared parametrization in recurrent language modeling.

2.2.1 Introduction

One of the principal challenges in computational linguistics is to account for the word order of the document or utterance being processed (Ghodsi et DeNero, 2016). Of course, the numbers of possible phrases grows exponentially with respect to a given phrase length, requiring an approximate approach to summarizing its content. RNN are such an approach, and they are used in various tasks in Natural Language Processing (NLP), such as machine translation (Luong *et al.*, 2015),

abstractive summarization (Paulus *et al.*, 2017) and question answering (Iyyer *et al.*, 2014). However, RNN, as approximations, suffer from numerical troubles that have been identified, such as that of recovering from past errors when generating phrases. We take interest in a model that mitigates this problem, mRNN, and how it has been and can be combined for new models. To evaluate these models, we use the task of *recurrent language modeling*, which consists in predicting the next token (character or word) in a document. This paper is organized as follows : RNN and mRNN are introduced respectively in Sections 2.2.2 and 2.2.3. Section 2.2.4 presents new and existing multiplicative models. Section 2.2.5 describes the datasets and experiments performed, as well as results obtained. Sections 2.2.6 discusses and concludes our findings.

2.2.2 Recurrent neural networks

RNN are powerful tools of sequence modeling that can preserve the order of words or characters in a document. A document is therefore a sequence of words, $x_1, \ldots, x_T$. Given the exponential growth of possible histories with respect to the sequence length, the probability of observing a given sequence needs to be approximated. RNN will make this approximation using the product rule,

$$P(x_1, \ldots, x_T) = P(x_1)P(x_2|x_1) \ldots P(x_T|x_1, \ldots, x_{T-1}),$$

and updating a *hidden state* at every time step. This state is first null,

$$h_0 = \mathbf{0}.$$

Thereafter, it is computed as a function of the past hidden state as well as the input at the current time step,

$$h_t = f(h_{t-1}, x_t),$$

known as the *transition function.* f is a learned function, often taking the form

$$h_t = \tanh(Ux_t + Wh_{t-1}).^{1}$$

This allows, in theory, for straightforward modeling of sequences of arbitrary length.

In practice, RNN encounter some difficulties that need some clever engineering to be mitigated. For example, learning long-term dependencies such as those found in language is not without its share of woes arising from numerical considerations, such as the well-known vanishing gradient problem (Bengio *et al.*, 1994). This can be addressed with gating mechanisms, such as LSTM (Hochreiter et Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho *et al.*, 2014).

A problem that is more specific to generative RNN is their difficulty recovering from past errors (Graves, 2013), which (Krause *et al.*, 2016) argue arises from having hidden-state transitions that are highly correlated across possible inputs. One approach to adapting RNN to have more input-dependent transition functions is to use the multiplicative "trick" (Sutskever *et al.*, 2011). This approximates the idea of having the input at each time synthesize a dedicated kernel of parameters dictating the transition from the previous hidden state to the next. These two approaches can be combined, as in the multiplicative LSTM (mLSTM) (Krause *et al.*, 2016).

We begin by contending that, in making RNN multiplicative, sharing what is

1. Additive biases are omitted throughout the paper for concision

known as the *intermediate state* does not significantly hinder performance when parameter counts are equal. We verify this with existing as well as new gated models on several well-known language modeling tasks.

2.2.3 Multiplicative RNNs

Most recurrent neural network architectures, including LSTM and GRU share the following building block :

$$\tilde{h}_t = Ux_t + Wh_{t-1}. \tag{2.1}$$

$\tilde{h}_t$ is the *candidate* hidden state, computed from the previous hidden state, h_{t-1}, and the current input, x_t, weighted by the parameter matrices W and U, respectively. This candidate hidden state may then be passed through gating mechanisms and non-linearities depending on the specific recurrent model.

Let us assume for simplicity that the input is a one-hot vector (one component is 1, the rest are 0 (Socher *et al.*, 2013) [see p.45]), as it is often the case in NLP. Then, the term Ux_t is reduced to a single column of U and can therefore be thought of as an input-dependent bias in the hidden state transition. As the dependencies we wish to establish between the elements of the sequences under consideration become more distant, the term Wh_t will have to be significantly larger than this input-dependent bias, Ux_t, in order to remain unchanged across time-steps. This will mean that from one time-step to the next, the hidden-to-hidden transition will be highly correlated across possible inputs. This can be addressed by having more input-dependent hidden state transitions, making RNN more expressive.

In order to remedy the aforementioned problem, each possible input i can be given its own matrix $W^{(i)}$ parameterizing the contribution of h_t to $\tilde{h}_t$.

$$\tilde{h}_t = Ux_t + \underbrace{(\sum_i W^{(i)} x_t^{(i)})}_{\mathbf{W}^{(x_t)}} h_{t-1}. \tag{2.2}$$

This is known as a tensor RNN (tRNN) (Sutskever *et al.*, 2011), because all the matrices can be stacked to form a rank 3 tensor, $\mathbf{W}$. The input x_t selects the relevant slice of the tensor in the one-hot case and a weighted sum over all slices in the dense case. The resulting matrix then acts as the appropriate W.

However, such an approach is impractical because of the high parameter count such a tensor would entail. The tensor can nonetheless be approximated by factorizing it (Taylor et Hinton, 2009) as follows :

$$\mathbf{W}^{(x_t)} = V\text{diag}(W_x x_t)W_h, \tag{2.3}$$

where W_x and W_h are weight matrices, and diag is the operator turning a vector v into a diagonal matrix where the elements of v form the main diagonal of said matrix. Replacing $\mathbf{W}^{(x_t)}$ in Equation (2.2) by this tensor factorization, we obtain

$$\tilde{h}_t = Ux_t + Vm_t, \tag{2.4}$$

where m_t is known as the *intermediate state*, given by

$$m_t = (W_x x_t) * (W_h h_{t-1}). \tag{2.5}$$

Here, $*$ refers to the Hadamard or element-wise product of vectors. The intermediate state is the result of having the input apply a learned filter via the new parameter kernel W to the factors of the hidden state. It should be noted that

the dimensionality of m_t is free and, should it become sufficiently large, the factorization becomes as expressive as the tensor. The ensuing model is known as a mRNN (Sutskever *et al.*, 2011).

2.2.4 Sharing intermediate states

While mRNN outperform simple RNN in character-level language modeling, they have been found wanting with respect to the popular LSTM (Hochreiter et Schmidhuber, 1997). This prompted (Krause *et al.*, 2016) to apply the multiplicative "trick" to LSTM resulting in the mLSTM, which achieved promising results in several language modeling tasks (Krause *et al.*, 2016).

mLSTM

Gated RNN, such as LSTM and GRU, use *gates* to help signals move through the network. The value of these gates is computed in much the same way as the candidate hidden state, albeit with different parameters. For example, LSTM uses two different gates, i and f in updating its memory cell, c_t,

$$c_t = f_t * c_{t-1} + i_t * \tanh(\tilde{h}_t). \tag{2.6}$$

It uses another gate, o, in mapping c_t to the new hidden state, h_t,

$$h_t = o_t * \sigma(c_t), \tag{2.7}$$

where σ is the sigmoid function, squashing its input between 0 and 1. f and i are known as forget and input gates, respectively.

The forget gates allows the network to ignore components of the value of the

memory cell at the past state. The input gate filters out certain components of the new hidden state. Finally, the output gates separates the memory cell from the actual hidden state. The values of these gates are computed at each time step as follows :

$$i_t = \sigma(U_i x_t + W_i h_{t-1}) \tag{2.8}$$

$$f_t = \sigma(U_f x_t + W_f h_{t-1}) \tag{2.9}$$

$$o_t = \sigma(U_o x_t + W_o h_{t-1}). \tag{2.10}$$

Each gate has its own set of parameters to infer. If we were to replace each $W_\star$ by a tensor factorization as in mRNN, we would obtain a mLSTM model.

However, in the original formulation of mLSTM, there is no factorization of each would-be $\mathbf{W}_\star$ *individually*. There is no separate intermediate state for each gate, as one would expect. Instead, a single intermediate state, m_t, is computed to replace h_{t-1} in *all* equations in the system, by Equation 2.5. Furthermore, each gate has its own $V_\star$ weighting m_t. Their values are computed as follows :

$$i_t = \sigma(W_i h_{t-1} + V_i m_t) \tag{2.11}$$

$$f_t = \sigma(W_f h_{t-1} + V_f m_t) \tag{2.12}$$

$$o_t = \sigma(W_o h_{t-1} + V_o m_t). \tag{2.13}$$

The model can therefore no longer be understood as as an approximation of the tRNN. Nonetheless, it has achieved empirical success in NLP. We therefore try

to explore the empirical merits of this shared parametrization and apply them to other RNN architectures.

True mLSTM

We have presented the original mLSTM model with its shared intermediate state. If we wish to remain true to the original multiplicative model, however, we have to factorize every would-be $W_\star$ tensor separately. We have :

$$i_t = \sigma(U_i x_t + V_i m_{i,t}) \tag{2.14}$$

$$f_t = \sigma(U_f x_t + V_f m_{f,t}) \tag{2.15}$$

$$o_t = \sigma(U_o x_t + V_o m_{o,t}), \tag{2.16}$$

with each $m_{\star,t}$ being given by a separate set of parameters :

$$m_{\star,t} = (W_{\star,x} x_t) * (W_{\star,h} h_{t-1}). \tag{2.17}$$

We henceforth refer to this model as true mLSTM (tmLSTM). We sought to apply the same modifications to the GRU model, as LSTM and GRU are known to perform similarly (Greff *et al.*, 2016; Chung *et al.*, 2014; Jozefowicz *et al.*, 2015). That is, we build a true multiplicative GRU (tmGRU) model, as well as a multiplicative GRU (mGRU) with a shared intermediate state.

GRU

The GRU was first proposed by (Cho *et al.*, 2014) as a lighter, simpler variant of LSTM. GRU relies on two gates, called, respectively, the *update* and *reset* gates, and no additional memory cell. These gates intervene in the computation of the hidden state as follows :

$$h_t = (1 - z_t)h_{t-1} + z_t \tanh(\tilde{h}_t), \tag{2.18}$$

where the candidate hidden state, $\tilde{h}_t$, is given by :

$$\tilde{h}_t = U_h x_t + W_h(r_t * h_{t-1}). \tag{2.19}$$

The update gate deletes specific components of the hidden state and replaces them with those of the candidate hidden state, thus updating its content. On the other hand, the reset gate allows the unit to start anew, as if it were reading the first symbol of the input sequence. They are computed much in the same way as the gates of LSTM :

$$z_t = \sigma(U_z x_t + W_z h_{t-1}), \tag{2.20}$$

$$r_t = \sigma(U_r x_t + W_r h_{t-1}). \tag{2.21}$$

True mGRU

We can now make GRU multiplicative by using the tensor factorization for z and r :

$$z_t = \sigma(U_z x_t + V_z m_{z,t}), \tag{2.22}$$

$$r_t = \sigma(U_r x_t) + V_r m_{r,t}, \tag{2.23}$$

with each $m_{\star,t}$ given by Eq. 2.17. There is a subtlety to computing $\tilde{h}_t$, as we need to apply the reset gate to h_{t-1}. While h_t itself is given by Eq. 2.4, $m_{h,t}$ is not computed the same way as in mLSTM and mRNN. Instead, it is given by :

$$m_{h,t} = (W_x x_t) * (W_h(r_t * h_{t-1})). \tag{2.24}$$

mGRU with shared intermediate state

Sharing an intermediate state is not as immediate for GRU. This is due to the application of r_t, which we need in computing the intermediate state that we want to share. That is, r_t and m_t would both depend on each other. We modify the role of r_t to act as a filter on m_t, rather than a reset on individual components of h_{t-1}. Note that, when all components of r_t go to zero, it amounts to having all components of h_{t-1} at zero. We have

$$z_t = \sigma(U_z x_t + V_z m_t) \tag{2.25}$$

and

$$r_t = \sigma(U_r x_t + V_r m_t). \tag{2.26}$$

$\tilde{h}_t$ is given by

$$\tilde{h}_t = U_h x_t + V_h(r_t * m_t), \tag{2.27}$$

with m_t the same as in mRNN and mLSTM this time, i.e. Eq.2.5. The final hidden state is computed the same way as in the original GRU (Eq.2.18).

2.2.5 Experiments in character-level language modeling

Character-level language modeling (or character prediction) consists in predicting the next character while reading a document one character at a time. It is a

common benchmark for RNN because of the heightened need for shared parametrization when compared to word-level models. We test mGRU on two well-known datasets, the Penn Treebank and Text8.

Penn Treebank. The Penn Treebank dataset (Marcus *et al.*, 1993) comes from a series of Wall Street Journal articles written in English. Following (Mikolov *et al.*, 2012), sections 0-20 were used for training, 21-22 for validation and 23-24 for testing, respectively, which amounts to 5.1M, 400K and 450K characters, respectively.

The vocabulary consists of 10K lowercase words. All punctuation is removed and numbers were substituted for a single capital N. All words out of vocabulary are replaced by the token `<unk>`.

The training sequences were passed to the model in batches of 32 sequences. Following (Krause *et al.*, 2016), we built an initial mLSTM model of 700 units. However, we set the dimensionality of the intermediate state to that of the input in order to keep the model small. We do the same for our mGRU, tmLSTM and tmGRU, changing only the size of the hidden state so that all four models have roughly the same parameter count. We trained it using the Adam optimizer (Kingma et Ba, 2014), selecting the best model on validation over 10 epochs. We apply no regularization other than a checkpoint which keeps the best model over all epochs. The performance of the model is evaluated using cross entropy in BPC, which is log_2 of perplexity.

All models outperform previously reported results for mLSTM (Krause *et al.*, 2016) despite lower parameter counts. This is likely due to our relatively small batch size. However, they perform fairly similarly. Encouraged by these results, we built an mGRU with both hidden and intermediate state sizes set to that of the original mLSTM (700). This version highly surpasses the previous state of the

Model	Parameter count	Error(BPC)
GRU (Bai *et al.*, 2018)	3M	1.53
mRNN (Mikolov *et al.*, 2012)	-	1.41
LSTM (Cooijmans *et al.*, 2016)	-	1.38
batch-normalized LSTM (Cooijmans *et al.*, 2016)	-	1.32
mLSTM (Krause *et al.*, 2016)	-	1.27
fast-slow LSTM (Mujika *et al.*, 2017)	7.2M	1.19
mLSTM	**292K**	**1.11**
tmLSTM	**292K**	**1.09**
tmGRU	**292K**	**1.08**
mGRU	**292K**	**1.07**
larger mGRU	**2.1M**	**0.98**

Tableau 2.1 Test set error on Penn Treebank and parameter counts in character-level language modeling

art while still having fewer parameters than previous work.

For the sake of comparison, results as well as parameter counts (where available) of our models (bold) and related approaches are presented in Table 2.1. mGRU and larger mGRU, our best models, achieved respectively an error of 1.07 and 0.98 BPC on the test data, setting a new state of the art for this task.

Text8. The Text8 corpus (Hutter, 2006) comprises the first 100M plain text characters in English from Wikipedia in 2006. As such, the alphabet consists of the 26 letters of the English alphabet as well as the space character. No vocabulary restrictions were put in place. As per (Mikolov *et al.*, 2012), the first 90M and 5M

Model	Parameter count	Error (BPC)
GRU (Bai *et al.*, 2018)	5M	1.53
mRNN (Mikolov *et al.*, 2012)	-	1.54
LSTM (Cooijmans *et al.*, 2016)	-	1.43
mLSTM (Krause *et al.*, 2016)	20M	1.42
mLSTM	**133K**	**1.37**
batch-normalized LSTM(Cooijmans *et al.*, 2016)	-	1.36
tmGRU	**133K**	**1.35**
tmLSTM	**133K**	**1.35**
mGRU	**133K**	**1.35**
large mLSTM (Krause *et al.*, 2016)	46M	1.27
larger mGRU	**877K**	**1.21**
LSTM (Krause *et al.*, 2017)*	45M	1.19

Tableau 2.2 Test set error on Text8 and parameter counts in character-level language modeling

characters were used for training and validation, respectively, with the last 5M used for testing.

Encouraged by our results on the Penn Treebank dataset, we opted to use similar configurations. However, as the data is one long sequence of characters, we divide it into sequences of 200 characters. We pass these sequences to the model in slightly larger batches of 50 to speed up computation. Again, the dimensionality of the hidden state for mLSTM is set at 450 after the original model, and that of the intermediate state is set to the size of the alphabet. The size of the hidden state

is adjusted for the other three models as it was for the PTB experiments. The model is also trained using the Adam optimizer over 10 epochs.

The best model as per validation data over 10 epochs achieves 1.40 BPC on the test data, slightly surpassing an mLSTM of smaller hidden-state dimensionality (450) but larger parameter count. Our results are more modest, as are those of the original mLSTM. Once again, results do not vary greatly between models.

As with the Penn Treebank, we proceed with building an mGRU with both hidden and intermediate state sizes set to 450. This improves performance to 1.21 BPC, setting a new state of the art for this task and surpassing a large mLSTM of 1900 units from (Krause *et al.*, 2016) despite having far fewer parameters (45M to 5M).

For the sake of comparison, results as well as parameter counts of our models and related approaches are presented in Table 2.2. It should be noted that some of these models employ *dynamic evaluation* (Graves, 2013), which fits the model further during evaluation. We refer the reader to (Krause *et al.*, 2017). These models are indicated by a star.

2.2.6 Conclusion

We have found that competitive results can be achieved with mRNN using small models. We have not found significant differences in the approaches presented, despite added non-intuitive parameter-sharing constraints when controlling for model size. Our results are restricted to character-level language modeling. While the relevance of statistical language modeling and recurrent language modeling to applied NLP tasks is uncertain (De Mulder *et al.*, 2015), we believe they are a fair means of comparison among RNN architectures. Along this line of thought, previous work on mRNN demonstrated their increased potential when compared to their regular variants (Sutskever *et al.*, 2011; Krause *et al.*, 2016; Radford *et al.*,

2017). We therefore offer other variants as well as a first investigation into their differences. We hope to have evinced the impact of increased flexibility in hidden-state transitions on RNN sequence-modeling capabilities. Further work in this area is required to transpose these findings into applied tasks in NLP.

CHAPITRE III

MODÈLES D'ATTENTION POUR L'ESTIMATION DU RISQUE DE DÉPRESSION

3.1 Contexte et références

Cette publication se penche sur la tâche 1 de la campagne d'évaluation internationale *eRisk 2018*[1] dont l'objectif est de détecter le plus tôt possible le risque de dépression chez les usagers d'un forum Internet. Les données ont été recueillies par Losada *et al.* (2018). Chaque usager est représenté par la séquence de ses écrits et la prédiction de risque de dépression qui y associée. Cette dernière a été établie en cherchant des phrases suggérant un diagnostic par un clinicien Puisque les écrits peuvent être issus de discussions différentes, nous choisissons d'ignorer leur ordre d'apparition. Nous les traitons en parallèle et conjointement en chargeant un RNN de résumer les écrits et un autre d'agréger ces résumés. Nous utilisons le mécanisme attentionnel pour permettre au RNN agrégateur d'accorder plus d'importance à certains écrits selon leur contenu. Nous obtenons des résultats prometteurs.

L'article a été présenté par l'auteur de ce mémoire le 29 mai 2019 à la conférence *Canadian AI 2019, The 32nd Canadian Conference on Artificial Intelligence*, à

1. eRisk 2018 `http://early.irlab.org/2018/index.html`

Kingston, Canada, et publié dans les actes de la conférence, *Springer Nature, LNCS Lecture Notes in Artificial Intelligence #11489.*

L'auteur de ce mémoire a formulé l'hypothèse de recherche ainsi que conçu et implémenté les architectures présentées. Il a également choisit et mené les expériences conformément à l'état de l'art. Finalement, il a effectué la majeure partie de la rédaction de l'article.

Reproductibilité. Le code source associé aux travaux présentés est disponible publiquement ici : `https://gitlab.ikb.info.uqam.ca/diego/canadianai2019`

3.2 Publication

Inter and Intra Document Attention for Depression Risk Assessment

Diego Maupomé, Marc Queudot, Marie-Jean Meurs

Université du Québec à Montréal

Abstract

We take interest in the early assessment of risk for depression in social media users. We focus on the eRisk 2018 dataset, which represents users as a sequence of their written online contributions. We implement four RNN-based systems to classify the users. We explore several aggregations methods to combine predictions on individual posts. Our best model reads through all writings of a user in parallel but uses an attention mechanism to prioritize the most important ones at each timestep.

3.2.1 Introduction

In 2015, 4.9 million Canadians aged 15 and over experienced a need for mental health care; 1.6 million felt their needs were partially met or unmet (Statistics Canada, 2017). In 2017, over a third of Ontario students, grades 7 to 12, reported having wanted to talk to someone about their mental health concerns but did not know who to turn to (Boak *et al.*, 2016). These numbers highlight a concerning but all too familiar notion : although highly prevalent, mental health concerns often go unheard. Nonetheless, mental disorders can shorten life expectancy by 7-24 years (Chesney *et al.*, 2014).

In particular, depression is a major cause of morbidity worldwide. Although prevalence varies widely, in most countries, the number of persons that would suffer from depression in their lifetime falls between 8 and 12% (Kessler *et al.*, 2003). Access to proper diagnosis and care is overall lacking because of a variety of reasons, from the stigma surrounding seeking treatment (Rodrigues *et al.*, 2014) to a high rate of misdiagnosis (Vermani *et al.*, 2011). These obstacles could be mitigated in some way among social media users by analyzing their output on these platforms to assess their risk of depression or other mental health afflictions. The analysis of user-generated content could give valuable insights into the users mental health, identify risks, and help provide them with better support (Ayers *et al.*, 2013; De Choudhury *et al.*, 2013). To promote such analyses that could lead to the development of tools supporting mental health practitioners and forum moderators, the research community has put forward shared tasks like CLPsych (Yates *et al.*, 2017) and the CLEF eRisk pilot task (Losada *et al.*, 2018). Participants must identify users at risk of mental health issues, such as eminent risk of depression, post traumatic stress disorder, or anorexia. These tasks provide participants with annotated data and a framework for testing the performance of their approaches.

In this paper, we present a neural approach to identify social media users at risk of depression from their writings in a subreddit forum, in the context of the eRisk 2018 pilot task. From a technical standpoint, the principal interest of this investigation is the use of different aggregation methods for predictions on groups of documents. Using the power of RNN for the sequential treatment of documents, we explore several manners in which to combine predictions on documents to make a prediction on its author.

3.2.2 Dataset

The dataset from the eRisk2018 shared task (Losada *et al.*, 2018) consists of the written production of `reddit` (Advance Publications Inc., 2018) English-speaking users.

The dataset was built using the writings of 887 users, and was provided in whole at the beginning of the task. Users in the RISK class have admitted to having been diagnosed with depression; NO_RISK users have not. It should be noted that the users' writings, or posts, may originate from different separate discussions on the website. The individual writings, however, are not labelled. Only the user as a whole is labelled as RISK or NO_RISK. The two classes of users are highly imbalanced in the training set with the positive class only counting 135 users to 752 in the negative class. Table 3.1 presents some statistics on the task dataset.

We use this dataset but consider a simple classification task, as opposed to the early-risk detection that was the object of the shared task.

	training		test	
	risk	**control**	**risk**	**control**
# users	135	752	79	741
# writings	49,557	481,837	40,665	504,523
submissions / subject	367.1	640.7	514.7	680.9
words / submission	27.4	21.8	27.6	23.7

Tableau 3.1 Statistics on the eRisk 2018 task dataset

3.2.3 Models

We represent users as sets of writings rather than sequences of writings. This is partly due to the intuition that the order of writings would not be significant in the context of forums, generally speaking. It is also due to the fact that treating writings sequentially would be cumbersome, especially if we consider training on all ten chunks. However, we do consider writings as sequences of words, as this is the main strength of RNN. We therefore write a user u as the set of his m writings, $u = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$. A given writing $\mathbf{x}^{(j)}$, is then a sequence of words, $\mathbf{x}^{(j)} = x_1^{(j)}, \ldots, x_\tau^{(j)}$, with τ being the index of the last word. Thus, $x_t^{(j)}$ is the t-th word of the j-th post for a given user.

Aggregating predictions on writings

Late Inter-Document Averaging We set out to put together an approach that aggregates predictions made individually and sequentially on the writings of a

user. That is, we read the different writings of a user in parallel and take the average prediction on them. This is our first model, Late Inter-Document Averaging (LIDA). Using the RNN architecture of our choice, we read each word of a post and update its *hidden state*,

$$h_t^{(j)} = f(x_t^{(j)}, h_{t-1}^{(j)}; \theta_{post}). \tag{3.1}$$

f is the *transition function* of the chosen RNN architecture, θ_{post} is the set of parameters of our particular RNN model and the initial state is set to zero,

$$h_0 = \mathbf{0}.$$

In practice, however, we take but a sample of users' writings and trim overlong writings (see Sec.3.2.5). LIDA averages over the final state of the RNN, $h_\tau^{(j)}$, across writings,

$$a = \frac{1}{m} \sum_{j=1}^{m} h_\tau^{(j)} \tag{3.2}$$

This average is then projected into a binary prediction for the user,

$$p = \sigma(\mathbf{u}^\top \begin{bmatrix} a \\ 1 \end{bmatrix}), \tag{3.3}$$

using σ, the standard logistic sigmoid function, to normalize the output and a vector of parameters, $\mathbf{u}$. By averaging over all writings, rather than taking the sum, we ensure that the number of writings does not influence the decision. However, we suspect that regularizing on the hidden state alone will not suffice, as the

problem remains essentially the same : gradient correction information will have to travel the entire length of the writings regardless of the corrections made as a results of other writings.

Continual Inter-Document Averaging. Our second model, Continual Inter-Document Averaging (CIDA), therefore aggregates the hidden state across writings at *every* time step, as opposed to only the final one. A first RNN, represented by its hidden state h_t, reads the writings as in Eq. 3.1. The resulting hidden states are averaged across writings and then fed as the input to a second RNN, represented by g_t,

$$a_t = \frac{1}{m} \sum_{j=1}^{m} h_t^{(j)}, \tag{3.4}$$

$$g_t = f(a_t, g_{t-1}; \theta_{user}). \tag{3.5}$$

g_τ is used to make a prediction similarly to Eq.3.3.

Inter-document attention

It stands to reason that averaging over the ongoing summary of each document would help in classifying a group of documents. Nonetheless, one would suspect that some documents would be more interesting than others to our task. Even if all documents were equally interesting, their interesting parts might not align well. Because we are reading them in parallel, we should try and prioritize the documents that are interesting at the current time step.

CIDA does not offer this possibility, as no weighting of terms is put in place in Eq.3.4. Consequently, we turn to the *attention mechanism* (Bahdanau *et al.*,

2014) to provide this information. While several manners of both applying and computing the attention mechanism exist (Luong *et al.*, 2015; Cheng *et al.*, 2016; Xu *et al.*, 2015), we compute the variant known as *general* attention (Luong *et al.*, 2015), which is both learned and content-dependent. In applying it, we introduce Inter-Document Attention (IDA), which will provide a weighted average to our previous model.

The computation of $h_t^{(j)}$, the post-level hidden state, remains the same, i.e. Eq.3.1. However, these values are compared against the previous user-level hidden state to compute the relevant *energy* between them, $\hat{\alpha}_{jt}$

$$\tilde{\alpha}_t^{(j)} = g_{t-1}\mathbf{W}_{att}h_t^{(j)}, \tag{3.6}$$

where $\mathbf{W}_{att}$ is a matrix of parameters that learns the compatibility between the hidden states of the two RNN. The resulting energy scalars, $\hat{\alpha}^{(j)}t$ are mapped to probabilities by way of softmax normalization,

$$\alpha_t^{(j)} = \frac{e^{\tilde{\alpha}_t^{(j)}}}{\sum_{k=1}^{m} e^{\tilde{\alpha}_t^{(k)}}}. \tag{3.7}$$

This probability is then used to weight the appropriate h_t,

$$a_t = \sum_{j=1}^{m} \alpha_t^{(j)} h_t^{(j)}. \tag{3.8}$$

g_t is given by Eq.3.5. Through the use of this probability weighting, we can understand a_t as an expected document summary at position t when grouping documents together. As in the previous model, a prediction on the user is made from g_τ.

Intra-document Attention

We extend our use of the attention mechanism in the aggregation to the parsing of individual documents. Similarly to our weighting of documents in aggregation dependent on the current aggregation state, we compare the current input to past inputs to evince a *context* for it. This is known in the literature as *self-attention* (Cheng *et al.*, 2016). We therefore modify the computation of h_t from Eq.3.1 by adding a context vector, $c_t^{(j)}$, corresponding to the ongoing context in document j at time t :

$$h_t^{(j)} = f(x_t^{(j)}, c_t^{(j)}, h_{t-1}^{(j)}; \theta_{post}). \tag{3.9}$$

This context vector is computed by comparing past inputs to the present document-level hidden state,

$$\tilde{\alpha}_{t,t'}^{(j)} = h_t^{(j)} \mathbf{W}_{intra} x_{t'}^{(j)}, \tag{3.10}$$

This weighting is normalized by softmax and used in adding the previous inputs together. We refer to this model as Inter- and Intra-Document Attention (InIDA).

This last attention mechanism arises from practical difficulties in learning long-range dependencies by RNN. While RNN are theoretically capable of summarizing sequences of arbitrary complexity in their hidden state, numerical considerations make learning this process through gradient descent difficult when the sequences are long or the state is too small (Bengio *et al.*, 1994). This can be addressed in different manners, such as gating mechanisms (Hochreiter et Schmidhuber, 1997; Cho *et al.*, 2014) and the introduction of multiplicative interactions (Sutskever *et al.*, 2011). Self-Attention is one such mechanism where the context vector acts

as a reminder of past inputs in the form of a learned expected context. It can be combined to other mechanisms with minimal parameter load.

3.2.4 Related Work

Choudhury et al. (De Choudhury *et al.*, 2013) used a more classical approach to classify Twitter users as being at risk of depression or not. They first manually crafted features that describe users' online behavior and characterize their speech. The measures were computed daily, so a user is represented as the time series of the features. Then, the training and predictions were done by a Support Vector Machine (SVM) with PCA for dimensionality reduction.

More similarly to our approach, Ive et al. (Ive *et al.*, 2018) used Hierarchical Attention Networks (Yang *et al.*, 2016) to represent user-generated documents. Sentence representations are learned using a RNN with an attention mechanism and are then used to learn the document's representation using the same network architecture. The computation of the attention weights they use is different from ours as it is non-parametric. Their equivalent of equation 3.6 would be

$$\tilde{\alpha}_t^{(j)} = h_t^{(j)\top} g_t^{(j)} \tag{3.11}$$

This means that the RNN learn the attention weights along with the representation of the sequences themselves. This attention function has been introduced in (Luong *et al.*, 2015) under that name of *dot*.

The *location-based function* (Luong *et al.*, 2015) is a simpler version of the *general attention* that we used, that only takes into account the target hidden state. It is stated as such :

$$\tilde{\alpha}_t^{(j)} = W_{att} g_t^{(j)} \tag{3.12}$$

The *additive* function introduced in (Bahdanau *et al.*, 2014), has been improved in (Luong *et al.*, 2015). Luong et al. use a concatenation layer to combine the information of the hidden state and the context vector.

$$\tilde{\alpha}_t^{(j)} = \tanh(W_{att} g_t^{(j)} + W_{att} h_t^{(j)}) \tag{3.13}$$

Content-based addressing was developed as part of Neural Turing Machines (Graves *et al.*, 2014), where the attention is focused on inputs that are similar to the values in memory.

$$\tilde{\alpha}_t^{(j)} = \text{cosine}[g_t^{(j)}, h_t^{(j)}] \tag{3.14}$$

3.2.5 Methodology

Preprocessing

As previously mentioned, documents are broken into words. The representation of these words is learned from the entirety of the training documents, all chunks included, using the skip-gram algorithm (Mikolov *et al.*, 2013). All words were turned to lowercase. Only the 40k most frequent words were kept. The embedded representation learned is of size 40, using a window of size five. The embeddings are are shared by all models.

Documents are trimmed at the end at a length of 66 words, which is longer than 90% of the posts in the dataset. The number of documents varies greatly across user classes. We take small random samples without replacement of 30 documents

per user at every iteration (epoch). We contend that sampling the user at every iteration allows us to train for longer as it is harder for the models to overfit when the components that make up each instance keep changing.

Model configurations

We use the mLSTM (Krause *et al.*, 2016) architecture as the post-level and user-level RNN, where applicable. The flexibility of the transition function in mLSTM has shown to be capable of arriving at highly abstract features on its own and can achieve competitive results in sentiment analysis (Radford *et al.*, 2017). Due to the limited number of examples, smaller models are required to avoid overfitting. We therefore set the embedded representation at 20 and the size of the hidden state of both RNN to 80. Parameter counts are shown in Table 3.2.

Training

For our experiments, we reshuffle the original eRisk 2018 dataset, as the training and test sets do not have the same proportions among labels. To provide our models with more training examples, we divide the dataset 9 :1, stratifying across labels. We use 10% of the training set as validation.

We train the models using the Adam optimizer (Kingma et Ba, 2014), making use of 10% of the training data for validation. Having posited random intra-user sampling as a means of training longer, we set the training time to 30 epochs, taking the best model on validation over all epochs. As noted, the two classes are highly imbalanced. We use inverse class weighting to counteract this.

Evaluation

The nature of the task, which is to prioritize finding positive users, and the class imbalance in the dataset, we use the f1-score as a first metric in validation and in the final testing phase. The f1-score is useful to assess the quality of classification between unbalanced two unbalanced classes, one of which is designated as the positive class. It is defined as the harmonic mean between *precision* (out of all the positive examples, how many are correctly classified as positive) and *recall* (out of all examples classified as positive, how many were indeed in the positive class).

Using True Positive (TP) as the number of positive examples correctly classified, False Positive (FP) the number of examples in the positive class incorrectly classifed, and True Negative (TN) and False Positive (FP) for the negative class, we have the following equations.

$$precision = \frac{TP}{TP + FP} \tag{3.15}$$

$$recall = \frac{TP}{TP + FN} \tag{3.16}$$

$$f1\text{-}score = 2 \times \frac{precision \times recall}{precision + recall} \tag{3.17}$$

We evaluate our models on the best result on a validation set of 10% of the training data. These best results are selected over 30 epochs.

3.2.6 Results

Our preliminary results in validation are in accordance with our hypotheses. That is, continual aggregation surpasses late aggregation but falls short of the more sophisticated attention model. Moreover, the noticeable difference in performance has little to no cost in terms of parameter count.

model	p.c.	precision	recall	f1-score
LIDA	31k	39.7	51.2	45.6
CIDA	95k	41.7	69.8	52.2
IDA	101k	45.6	73.2	56.2
InIDA	175k	47.4	72.8	57.4

Tableau 3.2 Parameter counts, precision, recall and f1-score (%) on the adapted test set for the eRisk 2018 corpus

3.2.7 Conclusion

In this paper, we have put forward three RNN-based models that aggregate documents to make a prediction on their author. We applied this model to the eRisk 2018 dataset, which associates a user, as a sequence of online forum posts, to a binary label that identifies them as being at risk for depression or not.

With the goal of using RNN to read the individual documents, we tested four methods of combining the resulting predictions, LIDA, CIDA, IDA and InIDA. We also introduced the inter-document attention mechanism. Our preliminary results show promise and confirm the parameter efficiency of the attention mechanism.

Future work could involve the use of dot-product alone, which, despite adding no parameters, has been found to be more effective for global attention (Luong *et al.*, 2015). An investigation into using late attention aggregation for all hidden states produced across all documents is also necessary.

CONCLUSION

Dans ce mémoire, nous avons présenté des **architectures** de réseaux récurrents multiplicatifs à **portes** et leur performance en modélisation récurrente de la langue. Nous avons également démontré comment différentes approches basées sur des réseaux récurrents peuvent être utlisées pour prédire le risque de dépression chez les usagers de réseaux sociaux en agrégeant leurs écrits. Nous avons obtenu des résultats à l'état de l'art voire meilleurs sur deux corpus connus en modélisation de la langue au niveau caractère. De plus, nous avons obtenu des bons résultats en détection de la dépression en intégrant le mécanisme d'attention pour mettre de l'avant les écrits les plus importants d'un usager. Ces résultats en modèles de langue et détection de la dépression ont été obtenus avec des modèles bien plus petits que ce qui est couramment utilisé, ce qui est prometteur en vue de leur portabilité. En effet, des petits modèles pourraient notamment être exécutés à même le téléphone portable de l'usager.

L'utilité des modèles de langue peut ne pas être apparente. Connaître la masse de probabilité d'une phrase n'est pas une application répandue en soi. Toutefois, comme démontré par ce travail, les modèles de langue peuvent servir à guider le choix d'approches à des tâches appliquées. Qui plus est, la modélisation de langue peut servir comme stabilisateur pour faciliter l'apprentissage en contexte supervisé (Erhan *et al.*, 2010; Dai et Le, 2015). Lorsqu'un modèle apprend à reconnaître les productions textuelles réelles, il apprend tacitement à reconnaître les caractéristiques de celles-ci. Cette capacité peut s'avérer très utile puisqu'on peut utiliser ces caractéristiques pour détecter les risques en santé mentale. De plus, alors que la détection des risques nécessite l'utilisation de données annotées

qui sont difficiles à obtenir, les modèles de langue peuvent apprendre du texte brut, qui est abondant. Nous avons démontré au Chapitre 2 que les mRNN peuvent obtenir des résultats compétitifs en modèles de langue. Une prochaine étape de travail serait donc d'explorer les approches de pré-entraînement en modélisation de la langue et leur impact en détection des troubles de la santé mentale.

Comme remarqué au Chapitre 1, la plupart des études existantes dans la littérature ont peu de fondement clinique. Une prochaine étape de recherche est donc l'élaboration de corpus en collaboration avec des professionnels de la santé. Ces derniers seraient en mesure de fournir un diagnostic juste et précis du patient. Ils seraient également en mesure d'obtenir le contenu de médias sociaux des patients ayant préalablement établi un consentement libre et éclairé. Notamment, Padrez *et al.* (2016) ont trouvé que 71% des patients souffrant de problèmes de santé mentale étaient prêts à partager leur contenu de médias sociaux.

Il est aussi primordial d'aider les cliniciens dans le traitement de la souffrance mentale, ce qui nécessite notamment l'usage d'approches *interprétables*. En effet, le clinicien a besoin d'une évaluation aussi détaillée de chaque cas afin d'établir un traitement approprié. Nous considérons que l'intégration du mécanisme attentionnel comme nous l'avons fait au Chapitre 3 constitue un premier pas dans cette direction. Il permet notamment d'orienter le clinicien vers les éléments les plus importants dans la décision diagnostique prise par l'algorithme. Cela aide à connaître la justesse de la prédiction ainsi qu'à mettre de l'avant les doléances principales du patient. Il est également primordial de soutenir la recherche fondamentale dans ce domaine. Nous espérons que le présent travail encouragera la recherche dans le domaine de l'intelligence artificielle et la santé mentale.

GLOSSAIRE

état caché vecteur maintenu par les réseaux récurrents pour résumer l'observation séquentielle en entrée. 11, 12, 13, 17

état intermédiaire vecteur issu de la combinaison multiplicative de l'entrée et l'état caché courant dans les réseaux récurrents multiplicatifs. 19

analyse de sentiments le domaine du Traitement Automatique de la Langue Naturelle qui s'intéresse à la reconnaissance d'émotions dans la langue écrite. 1

architecture choix particulier de fonctions de transition. 12, 19, 55

descente de gradient technique d'optimisation cherchant à trouver les optima en effectuant itérativement les petits changements résultant en la meilleure amélioration immédiate. 6

disparition ou explosion du gradient phénomène se produisant dans certaines architectures de réseaux récurrents où les corrections à apporter sont anéanties ou suramplifiées pour les observations longues. 13, 18

entropie croisée mesure de similarité entre deux distributions de probabilité. 16

ergodique se dit d'un processus stochastique dont les statistiques peuvent être inférées à partir d'une seule réalisation suffisamment longue. 16

fonction de transition dans les réseaux récurrents, fonction dictant la nouvelle valeur de l'état caché en fonction de sa valeur précédente et l'entrée la plus récente. 12, 25

fonction d'activation fonction différentiable partout appliqué à la somme des entrées d'un neurone dans un réseau de neurones. 6, 18

le fléau de la dimension Phénomènes nuisible à l'analyse des données lorsque celles-ci sont représentés dans des espaces de grande dimension. 15

modèle de sujets modèle qui associe les mots ou leurs séquences à des catégories thématiques latentes, des sujets. 21

modèle de langue récurrent modèle de langue qui effectue ses prédictions en faisant des prédictions pour chaque mot de la séquence, lue en ordre.. 17

modèle de langue modèle qui associe une masse de probabilité à une séquence de mots. 14, 17

perplexité mesure de l'aptitude d'un modèle à prédire un échantillon. 16, 17

porte mécanisme se servant d'un vecteur appris dont les valeurs tendent vers 0 ou 1 pour annuler ou maintenir certaines valeurs en entrée. 14, 18, 25, 55

rétropropagation algorithme qui facilite la descente de gradient en calculant les gradients des paramêtres en sens inverse de l'ordre d'exécution. 9

ACRONYMES

BPC Bits par Caractère. 16, 36, 37, 39

BPTT rétropropagation à travers le temps. 13

CIDA Continual Inter-Document Averaging. 47, 53, 54

GRU Gated Recurrent Unit. 28, 29, 31, 33, 34, 35, 37, 38, 59

IA Intelligence Artificielle. 1

IDA Inter-Document Attention. 47, 53, 54

InIDA Inter- and Intra-Document Attention. 49, 53, 54

LDA Allocation de Dirichlet Latente. 21

LIDA Late Inter-Document Averaging. 45, 46, 53, 54

LSTM Long Short-Term Memory network. 14, 28, 29, 31, 33, 34, 37, 38, 59

mGRU multiplicative GRU. 33, 35, 36, 37, 38, 39

MI-RNN RNN à Intégration Multiplicative. 19, 20

mLSTM multiplicative LSTM. 28, 31, 32, 33, 34, 35, 36, 37, 38, 39, 52, 60

mRNN multiplicative RNN. 19, 20, 26, 30, 31, 32, 34, 35, 37, 38, 39, 55

NLP Natural Language Processing. 26, 29, 32, 39

RNN Réseau Récurrent. 11, 12, 13, 14, 17, 18, 19, 22, 25, 26, 27, 28, 29, 30, 31, 32, 35, 39, 41, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54, 59, 60

SVM Support Vector Machine. 50

TALN Traitement Automatique de la Langue Naturelle. 1, 3

tmGRU true multiplicative GRU. 33, 36

tmLSTM true mLSTM. 33, 36

tRNN tensor RNN. 30, 32

NOTATION

Nombres et variables

a un scalaire.

a un vecteur.

A une matrice.

A un tenseur.

I_n la matrice identité de dimension n.

$e^{(i)}$ le i-ème vecteur de la base canonique.

$\dot{a}$ une variable aléatoire.

Mots et Documents

x un mot, souvent représenté par un indice dans un vocabulaire.

$e^{(x)}$ un mot en représentation orthonormale.

x un mot en représentation vectorielle quelconque.

$x_1, \ldots, x_T$ un document, explicité comme une séquence de T mots.

Paramètres

$\mathbf{w}$ un vecteur de paramètres appris.

$\mathbf{W}$ une matrice de paramètres appris.

$\boldsymbol{W}$ un tenseur de paramètres appris.

$\mathbf{b}$ un vecteur de biais appris.

RÉFÉRENCES

Advance Publications Inc. (2018). Reddit. `https://www.reddit.com/`. Accessed July 6, 2018.

American Psychiatric Association. (2013). *Diagnostic and Statistical Manual of Mental Disorders : Dsm-5*. Diagnostic and Statistical Manual of Mental Disorders. Amer Psychiatric Pub Incorporated. Récupéré de `https://books.google.ca/books?id=EIbMlwEACAAJ`

Ayers, J. W., Althouse, B. M., Allem, J.-P., Rosenquist, J. N. et Ford, D. E. (2013). Seasonality in seeking mental health information on Google. *American Journal of Preventive Medicine (AJPM)*, *44*(5), 520–525.

Bahdanau, D., Cho, K. et Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, *abs/1409.0473*. Récupéré de `http://arxiv.org/abs/1409.0473`

Bai, S., Kolter, J. Z. et Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, *abs/1803.01271*. Récupéré de `http://arxiv.org/abs/1803.01271`

Beck, A. T., Ward, C. H., Mendelson, M., Mock, J. et Erbaugh, J. (1961). An inventory for measuring depression. *Archives of general psychiatry*, *4*(6), 561–571.

Bengio, Y., Simard, P. et Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157–166.

Boak, A., Hamilton, H. A., Adlaf, E. M., Henderson, J. L. et Mann, R. E. (2016). *The mental health and well-being of Ontario students, 1991-2017 : Detailed Findings from the Ontario Student Drug Use and Health Survey.* CamhOSDUHS.

Cauchy, A. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, *25*(1847), 536–538.

Chen, L. (2009). Curse of dimensionality. *Encyclopedia of Database Systems*, 545–546.

Cheng, J., Dong, L. et Lapata, M. (2016). Long short-term memory-networks for machine reading. Dans *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 551–561.

Chesney, E., Goodwin, G. M. et Fazel, S. (2014). Risks of all-cause and suicide mortality in mental disorders : a meta-review. *World Psychiatry*, *13*(2), 153–160.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. et Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv :1406.1078.*

Chung, J., Gulcehre, C., Cho, K. et Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv :1412.3555.*

Cooijmans, T., Ballas, N., Laurent, C. et Courville, A. C. (2016). Recurrent Batch Normalization. *CoRR, abs/1603.09025.* Récupéré de `http://arxiv.org/abs/1603.09025`

Coppersmith, G., Dredze, M., Harman, C., Hollingshead, K. et Mitchell, M. (2015). CLPsych 2015 shared task : Depression and PTSD on Twitter. Dans *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology (CLPsych) : From Linguistic Signal to Clinical Reality*, 31–39.

Dai, A. M. et Le, Q. V. (2015). Semi-Supervised Sequence Learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, et R. Garnett (dir.), *Advances in Neural Information Processing Systems 28* 3079–3087. Curran Associates, Inc.

De Choudhury, M., Gamon, M., Counts, S. et Horvitz, E. (2013). Predicting depression via social media. Dans *Seventh international AAAI conference on weblogs and social media.*

De Mulder, W., Bethard, S. et Moens, M.-F. (2015). A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, *30*(1), 61–98.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P. et Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, *11*(Feb), 625–660.

Ghodsi, A. et DeNero, J. (2016). An analysis of the ability of statistical language models to capture the structural properties of language. Dans *Proceedings of the 9th International Natural Language Generation conference*, 227–231.

Gkotsis, G., Oellrich, A., Velupillai, S., Liakata, M., Hubbard, T. J., Dobson, R. J. et Dutta, R. (2017). Characterisation of mental health conditions in social media using informed deep learning. *Scientific reports*, *7*, 45141.

Goudreau, M. W., Giles, C. L., Chakradhar, S. T. et Chen, D. (1994). First-order versus second-order single-layer recurrent neural networks. *IEEE Transactions on Neural Networks*, *5*(3), 511–513.

Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. *CoRR, abs/1308.0850*. Récupéré de `http://arxiv.org/abs/1308.0850`

Graves, A., Wayne, G. et Danihelka, I. (2014). Neural Turing Machines. *arXiv preprint arXiv :1410.5401*.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. et Schmidhuber, J. (2016). LSTM : A search space odyssey. *IEEE transactions on neural networks and learning systems*.

Hochreiter, S. et Schmidhuber, J. (1997). Long Short-term Memory. *Neural computation*, *9*(8), 1735–1780.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, *4*(2), 251–257.

Hutter, M. (2006). Human Knowledge Compression Contest. Récupéré de `http://prize.hutter1.net/`

Ive, J., Gkotsis, G., Dutta, R., Stewart, R. et Velupillai, S. (2018). Hierarchical neural model with attention mechanisms for the classification of social media text related to mental health. Dans *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology : From Keyboard to Clinic*, 69–77., New Orleans, LA. Association for Computational Linguistics. `http://dx.doi.org/10.18653/v1/W18-0607`. Récupéré de `https://www.aclweb.org/anthology/W18-0607`

Iyyer, M., Boyd-Graber, J., Claudino, L., Socher, R. et Daumé III, H. (2014). A neural network for factoid question answering over paragraphs. Dans *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 633–644.

Jordan, M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. Dans *Proc. of the Eighth Annual Conference of the Cognitive Science Society (Erlbaum, Hillsdale, NJ), 1986.*

Jozefowicz, R., Zaremba, W. et Sutskever, I. (2015). An empirical exploration of recurrent network architectures. Dans *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2342–2350.

Kessler, R., Berglund, P., Demler, O. et et al (2003). The epidemiology of major depressive disorder : Results from the national comorbidity survey replication (ncs-r). *JAMA*, *289*(23), 3095–3105.

Kim, Y. (2014). Convolutional neural networks for sentence classification. Dans *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746–1751. Association for Computational Linguistics. `http://dx.doi.org/10.3115/v1/D14-1181`. Récupéré de `http://www.aclweb.org/anthology/D14-1181`

Kingma, D. P. et Ba, J. (2014). Adam : A Method for Stochastic Optimization. *CoRR, abs/1412.6980*. Récupéré de `http://arxiv.org/abs/1412.6980`

Krause, B., Kahembwe, E., Murray, I. et Renals, S. (2017). Dynamic evaluation of neural sequence models. *CoRR, abs/1709.07432*. Récupéré de `http://arxiv.org/abs/1709.07432`

Krause, B., Lu, L., Murray, I. et Renals, S. (2016). Multiplicative LSTM for Sequence Modelling. *arXiv preprint arXiv :1609.07959*.

Krippendorff, K. (1986). Web dictionary of cybernetics and systems. *A Dictionary of Cybernetics*.

Lim, K.-L., Jacobs, P., Ohinmaa, A., Schopflocher, D. et Dewa, C. S. (2008). A new population-based measure of the economic burden of mental illness in canada. *Chronic Dis Can*, *28*(3), 92–98.

Losada, D. E., Crestani, F. et Parapar, J. (2018). Overview of eRisk – Early Risk Prediction on the Internet. Dans *Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018)*, Avignon, France.

Luong, T., Pham, H. et Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. Dans *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412–1421.

Manning, C. D., Manning, C. D. et Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Marcus, M. P., Marcinkiewicz, M. A. et Santorini, B. (1993). Building a large annotated corpus of English : The Penn Treebank. *Computational linguistics*, *19*(2), 313–330.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. et Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, et K. Q. Weinberger (dir.), *Advances in Neural Information Processing Systems 26* 3111–3119. Curran Associates, Inc.

Mikolov, T., Sutskever, I., Deoras, A., Le, H.-S., Kombrink, S. et Cernocky, J. (2012). Subword language modeling with neural networks. *preprint (http ://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf).*

Mujika, A., Meier, F. et Steger, A. (2017). Fast-slow recurrent neural networks. Dans *Advances in Neural Information Processing Systems*, 5917–5926.

Padrez, K. A., Ungar, L., Schwartz, H. A., Smith, R. J., Hill, S., Antanavicius, T., Brown, D. M., Crutchley, P., Asch, D. A. et Merchant, R. M. (2016). Linking social media and medical record data : a study of adults presenting to an academic, urban emergency department. *BMJ Qual Saf*, *25*(6), 414–423.

Paulus, R., Xiong, C. et Socher, R. (2017). A deep reinforced model for abstractive summarization. *CoRR*, *abs/1705.04304*. Récupéré de `http://arxiv.org/abs/1705.04304`

Pennebaker, J. W., Francis, M. E. et Booth, R. J. (2001). Linguistic inquiry and word count : Liwc 2001. *Mahway : Lawrence Erlbaum Associates*, *71*(2001), 2001.

Radford, A., Józefowicz, R. et Sutskever, I. (2017). Learning to Generate Reviews and Discovering Sentiment. *CoRR*, *abs/1704.01444*.

Resnik, P., Armstrong, W., Claudino, L., Nguyen, T., Nguyen, V.-A. et Boyd-Graber, J. (2015). Beyond LDA : exploring supervised topic modeling for depression-related language in Twitter. Dans *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology : From Linguistic Signal to Clinical Reality*, 99–107.

Resnik, P., Garron, A. et Resnik, R. (2013). Using topic modeling to improve prediction of neuroticism and depression in college students. Dans *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1348–1353.

Rodrigues, S., Bokhour, B., Mueller, N., Dell, N., Osei-Bonsu, P. E., Zhao, S., Glickman, M., Eisen, S. V. et Elwy, A. R. (2014). Impact of stigma on veteran treatment seeking for depression. *American Journal of Psychiatric Rehabilitation*, *17*(2), 128–146.

Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Rude, S., Gortner, E.-M. et Pennebaker, J. (2004). Language use of depressed and depression-vulnerable college students. *Cognition & Emotion*, *18*(8), 1121–1133.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. *et al.* (1988). Learning representations by back-propagating errors. *Cognitive modeling*, *5*(3), 1.

Shing, H.-C., Nair, S., Zirikly, A., Friedenberg, M., Daumé III, H. et Resnik, P. (2018). Expert, crowdsourced, and machine assessment of suicide risk via online postings. Dans *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology : From Keyboard to Clinic*, 25–36., New Orleans, LA. Association for Computational Linguistics. `http://dx.doi.org/10.18653/v1/W18-0603`. Récupéré de `https://www.aclweb.org/anthology/W18-0603`

Siegelmann, H. T. et Sontag, E. D. (1991). Turing computability with neural nets. *Applied Mathematics Letters*, *4*(6), 77–80.

Socher, R., Bengio, Y. et Manning, C. (2013). Deep Learning for NLP. *Tutorial at Association of Computational Logistics (ACL), 2012, and North American Chapter of the Association of Computational Linguistics (NAACL)*.

Statistics Canada (2017). Accessing Mental Health Care in Canada. Récupéré le 2019-01-29 de `https://www150.statcan.gc.ca/n1/pub/11-627-m/11-627-m2017019-eng.htm`

Sutskever, I., Martens, J. et Hinton, G. E. (2011). Generating text with recurrent neural networks. Dans *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 1017–1024.

Tausczik, Y. R. et Pennebaker, J. W. (2010). The psychological meaning of words : Liwc and computerized text analysis methods. *Journal of language and social psychology*, *29*(1), 24–54.

Taylor, G. W. et Hinton, G. E. (2009). Factored conditional restricted Boltzmann machines for modeling motion style. Dans *Proceedings of the 26th annual international conference on machine learning*, 1025–1032. ACM.

Vermani, M., Marcus, M. et Katzman, M. A. (2011). Rates of detection of mood and anxiety disorders in primary care : a descriptive, cross-sectional study. *The primary care companion to CNS disorders*, *13*(2).

Wu, Y., Zhang, S., Zhang, Y., Bengio, Y. et Salakhutdinov, R. R. (2016). On Multiplicative Integration with Recurrent Neural Networks. Dans *Advances in Neural Information Processing Systems*, 2856–2864.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R. et Bengio, Y. (2015). Show, attend and tell : Neural image caption generation with visual attention. Dans F. Bach et D. Blei (dir.). *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 de *Proceedings of Machine Learning Research*, 2048–2057., Lille, France. PMLR. Récupéré de `http://proceedings.mlr.press/v37/xuc15.html`

Yang, Z., Dai, Z., Salakhutdinov, R. et Cohen, W. W. (2017). Breaking the softmax bottleneck : A high-rank RNN language model. *CoRR, abs/1711.03953*. Récupéré de `http://arxiv.org/abs/1711.03953`

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A. et Hovy, E. (2016). Hierarchical Attention Networks for Document Classification. Dans *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, 1480–1489.

Yates, A., Cohan, A. et Goharian, N. (2017). Depression and self-harm risk assessment in online forums. *CoRR, abs/1709.01848*. Récupéré de `http://arxiv.org/abs/1709.01848`