Liste des acronymes, sigles et symboles

AREQ	Assemblée des Responsables des Équipes
BPC	*Bytes Per Character* (octets par caractère, mesure de qualité de compression)
CNRS	Centre National de la Recherche Scientifique
CSV	*comma-separated values* (valeurs séparées par des virgules)
Emb.	module d'*embedding*
EUREKA	aucune description de ce sigle n'a été trouvée
GiB	*gibibyte* (gigaoctet informatioque, 1024 MiB)
GMSNN	*Growing Multi-Scale Recurrent Neural Network* (réseau de neurones récurrents multi-échelles croissant)
GPU	*Graphical Processing Unit* (processeur graphique)
h	heure
IDMC	Indtitut des sciences du Digital, Management et Cognition
INRIA	Institut National de Recherche en Informatique et en Automatique
ITEA	*Information Technology for European Advancement*
KiB	*kibibyte* (kilooctet informatioque, 1024 octets)
Lin.	module linéaire
log_n	logarithme en base n
LORIA	Laboratoire Lorrain d'Informatique et ses Applications
LSTM	*Long Short Term Memory* (réseau récurent à mémoire à court et long terme)
MiB	*mebibyte* (mégaoctet informatioque, 1024 KiB)
min	minute
PAPUD	*Profiling and Analysis Platform Using Deep Learning*
RNN	*Recurrent Neural Network* (réseau de neurones récurrents)
Sem.	semaine
SYNALP	*SYmbolic and statistical NAtural Language Processing*
TAL	traitement automatique des langues
UL	Université de Lorraine
UMR	Unité Mixte de Recherche
XML	*eXtensible Markup Language*

1.2 Objectifs du stage

Deux objectifs se sont succédé durant le stage.

L'objectif initial du stage était d'explorer une idée d'architecture innovante de réseau de neurones artificiels, imaginée par Mr. Cerisara, le maître de stage. Il s'agit du projet GMSNN (réseau de neurones récurrents multi-échelles croissant, *Growing Multi-Scale Recurrent Neural Network* en anglais, voir chapitre 4, page 25).

Cependant, à l'issue du deuxième mois du stage, nous avons changé d'objectif.

Le nouvel objectif a été la réalisation d'un réseau de neurones artificiels et des outils nécessaires à son utilisation, en mettant à profit les connaissances acquises durant la première partie du stage. Cette réalisation doit servir de base technique pour une partie du projet PAPUD (*Profiling and Analysis Platform Using Deep Learning*, voir chapitre 10, page 59).

1.3 Plan du rapport

Nous venons de présenter à la fois le contexte, les enjeux, et les objectifs généraux du stage.

Nous poursuivrons par la définition des termes et concepts principaux utilisés dans ce rapport, afin d'en faciliter la compréhension.

Nous aborderons ensuite, en deux parties, le contenu du stage.

La complexité de ce dernier est qu'il est composé de deux projets, le projet GMSNN et le projet PAPUD, l'un étant la continuation de l'autre. En effet, les conclusions tirées du premier projet ont servi de base pour le second. C'est pourquoi, pour simplifier la lecture du présent rapport, chaque projet sera traité suivant le même plan.

Pour chacun d'eux, nous présenterons le contexte, les enjeux, et les entités impliquées dans le projet. Nous décrirons ensuite le modèle réalisé avant de rapporter le travail effectué. Enfin, une conclusion résumera les points majeurs du projet.

Enfin, nous ferons un bilan de l'ensemble du travail réalisé pour en tirer les apports principaux.

2.3.3 Contexte et dépendances

Dans le cadre d'un modèle de la langue, le contexte est l'ensemble des informations disponibles hors du mot ou caractère à prédire.

On parle aussi de dépendances entre d'une part les mots ou caractères et d'autre part l'élément du contexte correspondant.

Parmi les informations contenues dans le contexte d'un mot, on peut trouver aussi bien le sens des mots environnants que la structure syntaxique de la phrase, ou encore des informations plus générales (comme fait que les chats soient des mammifères).

On considère que, plus on a d'informations contextuelles, plus le modèle de la langue est précis. Par exemple, si on nous dit « un chat », il sera plus difficile de prédire la couleur du chat que si on nous dit « un chat de couleur sombre ».

2.4 Performance et mesure

Le dernier concept important présenté dans ce chapitre est celui de performance des modèles produits.

La performance d'un modèle est évaluée par trois composantes :
— la qualité maximale des résultats produits ; dans le cas d'un modèle de la langue, il s'agit de la qualité de la prédiction ;
— le temps d'entraînement nécessaire pour atteindre cette qualité (nombre d'époques, durée, etc.) ;
— la consommation de ressources nécessaire pour atteindre cette qualité (mémoire, puissance de calcul, …).

Afin d'évaluer la performance, des mesures ont été définies :
— pour mesurer la qualité du résultat, on utilise l'écart entre le résultat produit et le résultat attendu ; une mesure définie dans la littérature et utilisée dans les annexes est le BPC [1] ;
— pour mesurer le temps d'entraînement, on utilise le nombre d'époques et le temps nécessaire par époque, la durée totale en heures, etc. ;
— pour mesurer la consommation de ressources, on évalue l'espace mémoire occupé (en MiB ou GiB), la puissance de calcul utilisée (pourcentage de la puissance disponible), etc.

Un modèle optimal serait un modèle qui atteint d'excellents résultats (l'écart entre ce qui est attendu et ce qui est produit le plus faible possible), le plus rapidement possible, en consommant le moins de ressources possible.

1. Le BPC (*Bits Per Character* en anglais) [3] est originalement une mesure de la qualité de compression de texte, mais plusieurs publications ont détourné cette mesure en s'en servant d'estimation de la qualité d'un modèle de la langue au niveau du caractère. Dans cet usage, le BPC est assimilable à une mesure de la précision des résultats du modèle de la langue. Plus le BPC est proche de 0 plus la qualité du modèle de la langue est élevé.

Estimation de la consommation normale du modèle

La première étape, qui est détaillée dans l'annexe A.2 (page 88), a été d'estimer l'usage normal de la mémoire (sans fuite), et d'isoler les paramètres qui ont le plus d'impact sur la consommation de mémoire. Ceci a confirmé que l'explosion de la consommation n'était pas due à l'architecture en elle-même et qu'il s'agissait bien d'une anomalie dans le fonctionnement du programme.

Résolution des fuites

L'analyse et la résolution des fuites de mémoire s'est révélée ardue. Si quelques fuites mineures ont été simples à détecter et réparer, la principale fuite était due à une spécificité non documentée de PyTorch.

En effet, PyTorch utilise la différentiation automatique [7] pour mettre à jour les paramètres du réseau de neurones artificiels. Pour cela, PyTorch a besoin de connaître la suite d'opérations et l'implication des différents paramètres du modèle et se base sur un « graphe de computation ». C'est le mode de gestion de ce graphe, couplé aux spécificités de l'architecture GMSNN, qui est la cause de la principale fuite mémoire.

Le rapport dans l'annexe A.4 (page 97) présente un extrait de la résolution du problème.

Conclusion

Le problème de la fuite de mémoire a été résolu et, avec lui, celui de la lenteur de l'entraînement. On peut déplorer de ne pas avoir analysé plus en profondeur cet étrange lien entre la mémoire et le temps d'entraînement. Cependant, résoudre les problèmes de fuites de mémoire et de lenteur de l'entraînement était l'objectif principal de cette étape, et l'optimisation du module GMSNN a pu reprendre.

On notera l'ampleur de l'optimisation par rapport à la version initiale :
- le temps d'entraînement a été réduit par un facteur 5 000 (de plus de 400 h à environ 5 min pour une époque) ;
- la consommation de mémoire est passée d'une utilisation en constante augmentation, dépassant les 6 GiB par époque, à une consommation constante inférieure à 200 MiB.

7. La différentiation automatique est un algorithme clé qui, en mettant en œuvre le principe de rétro-propagation du gradient, permet d'entraîner les réseaux de neurones. En effet, la rétro-propagation du gradient permet de déterminer l'implication de chaque paramètres dans les résultats obtenus. Ainsi, on peut déterminer de quelle façon mettre à jour les paramètres pour améliorer les résultats.

Pour ce qui est du modèle simple, il a été décidé de ne pas utiliser de RNN, bien trop lent pour la quantité de données à traiter. On y préférera un réseau de neurones artificiels basique.

On peut noter que les conclusions du projet GMSNN ont été appliquées, autant pour le déroulement du projet que pour le type de modèle à utiliser.

10.3 Projet PAPUD

La tâche qui nous a été confiée est la réalisation du modèle simple.

Plus exactement, étant donné qu'il était évident que la durée restante du stage serait insuffisante pour réaliser le modèle simple et l'optimiser au mieux, nos objectifs étaient la réalisation d'un prototype du modèle, et la mise en place des outils nécessaires à l'entraînement. Ceux-ci sont principalement les outils de gestion et de prétraitement des données, les outils d'évaluation des performances du modèle, et l'algorithme d'entraînement du modèle.

La description des caractéristiques du modèle est disponible dans le chapitre 12 (page 65).

Par la suite, nous désignerons ce projet par « projet PAPUD ».

10.4 Organisation du travail

Contrairement au projet GMSNN, ce projet s'est déroulé en équipe (voir section 9.1, page 55). Cependant, étant, avec le maître de stage, les seuls habitués à manipuler des réseaux de neurones, nous avons travaillé individuellement sur la définition de l'architecture et la réalisation du prototype.

Le projet GMSNN s'étant bien déroulé, une organisation similaire a été mise en place afin de tenir ces autres membres de l'équipe de projet informés de l'avancement et des conclusions de notre travail. Plus exactement, nous avons continué de rédiger des rapports d'avancement et nous avons présenté la progression du travail au cours de réunions hebdomadaires.

Les rapports de ce projet sont également disponibles en annexe (voir l'annexe B, page 159). Le rapport d'une des réunions est disponible pour l'exemple à l'annexe B.6 (page 172).

Glossaire

Terme	Description	Pages
Apprentissage automatique	Défini sous-section 2.1.1, page 15.	15, 16
Apprentissage profond	Défini sous-section 2.2.1, page 16.	13, 16, 28, 31, 51, 56, 62, 69, 73, 74
Architecture	Défini sous-section 2.2.3, page 16.	16, 17
Batch	Défini sous-section 7.5.4, page 44.	42, 44, 68, 71
Bogue	Défaut d'un logiciel entraînant des anomalies de fonctionnement.	35
Centres de calcul	Défini sous-section 9.4.2, page 57.	57
Cloud	Défini sous-section 9.4.2, page 57.	57
Corpus	Défini sous-section 2.1.4, page 16.	16, 30, 67
Différentiation automatique	Défini note 7, page 43.	43
Données	Défini sous-section 2.1.4, page 16.	13, 15, 16, 25, 26, 32, 40, 59, 60, 61, 67, 68, 70
Embedding	Défini sous-section 7.2.2, page 36.	36, 38, 65, 66
Entraînement	Défini sous-section 2.1.3, page 15.	15, 16, 67, 68, 69
Époque	Défini sous-section 2.1.4, page 16.	16, 18
État de l'art	Défini note 2, page 27.	26, 35, 37, 47, 51
Exemple	Défini sous-section 2.1.4, page 16.	16, 25, 40
Fichier de journalisation	Défini chapitre 11, page 63.	59, 63
Matrice	Objet mathématique similaire à un tableau de valeurs.	16, 28
Modèle	Défini sous-section 2.1.2, page 15.	14, 15, 16, 18, 26, 28, 30, 31, 35, 36, 37, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 52, 67, 68, 69, 73
Modèle de la langue	Défini sous-section 2.3.2, page 17.	13, 17, 18, 23, 26, 31, 35, 47, 51, 59, 65, 66
Module	Défini sous-section 2.2.3, page 16.	16
Paramètre	Défini sous-section 2.2.4, page 17.	16, 42, 43, 45, 46, 69
Prétraitement des données	Défini sous-section 2.1.4, page 16.	16, 29, 51, 60, 61, 68, 70
Processeur graphique (GPU)	Défini note 3, page 28.	28

Terme	Description	Pages
Réseau de neurones récurrents multi-échelles croissant (GMSNN)	Défini chapitre 6, page 31.	26, 31, 32, 35, 37, 42, 43, 44, 45, 46, 51, 52, 65
Réseau récurrent à mémoire à court et long terme (LSTM)	Défini sous-section 2.2.5, page 17.	17, 25, 36, 37, 42
Réseau de neurones récurrents (RNN)	Défini sous-section 2.2.5, page 17.	17, 25, 31, 32, 36, 37, 38, 39, 42, 45, 51, 52, 59, 66, 73
Réseau de neurones	Défini sous-section 2.2.2, page 16.	13, 14, 16, 17, 25, 28, 31, 36, 39, 43, 44, 51, 59, 65, 74
Rétro-propagation du gradient	Défini note 7, page 43.	43
Tenseur	Défini sous-section 2.2.2, page 16.	16, 28, 36, 38, 40, 65
Traitement automatique des langues (TAL)	Défini sous-section 2.3.1, page 17.	13, 17, 23, 25, 74