Attention Is All You Need

Des RNNs aux Transformers

Jean-Michel Marin

Mars 2026

Université de Montpellier

Modèles de langage neuronaux



Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

31st Conference on Neural Information Processing Systems, Long Beach, 2017

Modèles de type Transformer

NIPS 2017 PAC-Bayes workshop (Some recent advances on Approximate Bayesian Computation techniques)

Modèles de langage neuronaux

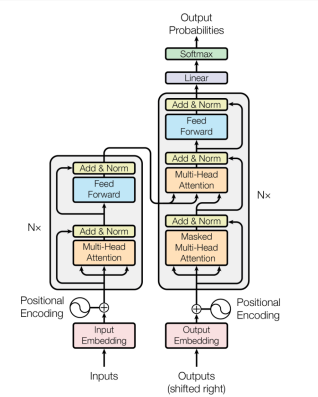
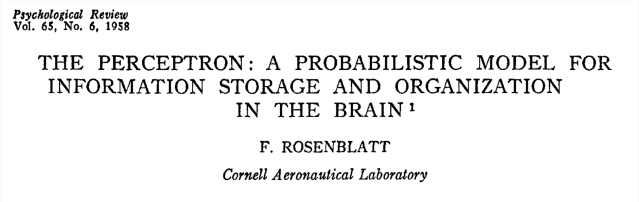


Figure 1: The Transformer - model architecture.

Architecture à mécanisme d'attention (Transformer)

Modèles de langage neuronaux



Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN [1]

F. ROSENBLATT

Cornell Aeronautical Laboratory

Psychological Review, 1958

Réseaux de neurones (Neural Networks)

Modèles de langage neuronaux

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

Nature, 1986

Rétropropagation des erreurs (Back-Propagating Errors)

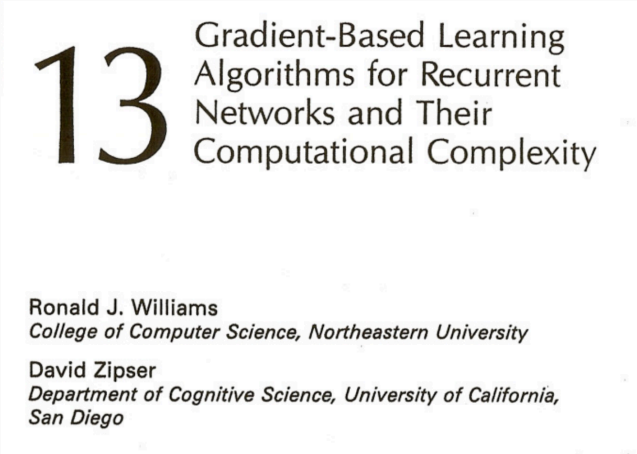
Modèles de langage neuronaux



Cognitive Science, 1990

Réseaux de neurones récurrents (Recurrent Neural Networks)

Modèles de langage neuronaux



13 Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity

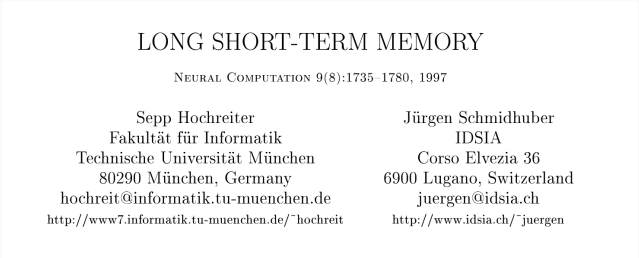
Ronald J. Williams
College of Computer Science, Northeastern University

David Zipser
Department of Cognitive Science, University of California, San Diego

Chapter 13, Backpropagation: theory, architectures, and application, 1995

Rétropropagation à travers le temps (Back-Propagation Through Time)

Modèles de langage neuronaux



LONG SHORT-TERM MEMORY

Neural Computation 9(8):1735–1780, 1997

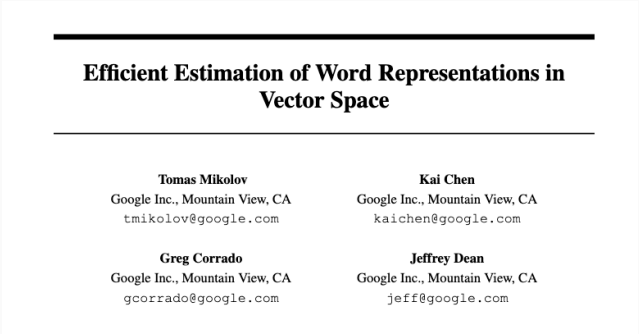
Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

Mémoire à long et court terme (Long Short-Term Memory)

Neural Computation, 1997

Modèles de langage neuronaux



Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

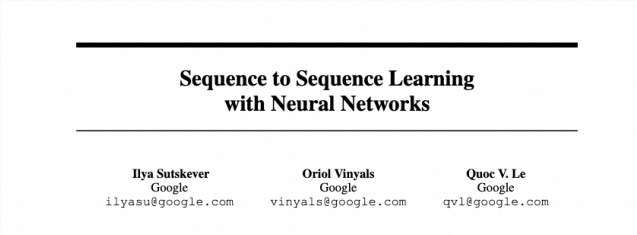
Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

1st International Conference on Learning Representations (ICLR), Scottsdale, 2013
Plongement (embedding)

Et aussi de nombreux travaux de Alex Graves et co-auteurs

Modèles de langage neuronaux



Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

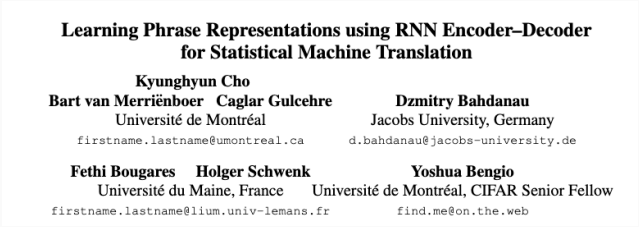
Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

28th Conference on Neural Information Processing Systems, Montréal, 2014

Deux LSTM un encodeur et un décodeur, traduction anglais/français

Modèles de langage neuronaux



Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

Kyunghyun Cho
Bart van Merriënboer **Caglar Gulcehre** **Dzmitry Bahdanau**
Université de Montréal Jacobs University, Germany
firstname.lastname@umontreal.ca d.bahdanau@jacobs-university.de

Fethi Bougares **Holger Schwenk** **Yoshua Bengio**
Université du Maine, France Université de Montréal, CIFAR Senior Fellow
firstname.lastname@lium.univ-lemans.fr find.me@on.the.web

19th Conference on Empirical Methods in Natural Language Processing, Doha, 2014

Deux RNN un encodeur et un décodeur, traduction anglais/français

Modèles de langage neuronaux

Review Article | Published: 27 May 2015

Deep learning

Yann LeCun, Yoshua Bengio & Geoffrey Hinton

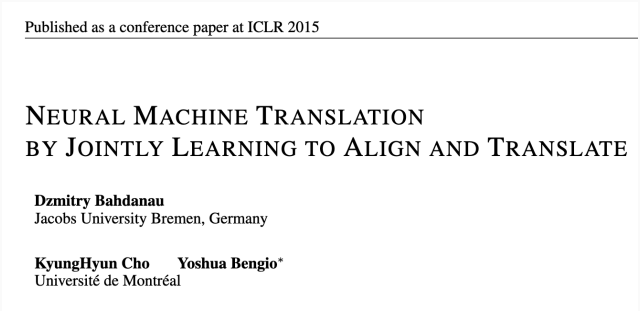
Nature **521**, 436–444 (2015) | Cite this article

1.20m Accesses | **78k** Citations | **1660** Altmetric | Metrics

Nature, 2015

Apprentissage Profond (Deep Learning)

Modèles de langage neuronaux



Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

3rd International Conference on Learning Representations, San Diego, 2015

Mécanisme d'attention (Attention mechanism)

Modèles de langage neuronaux



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

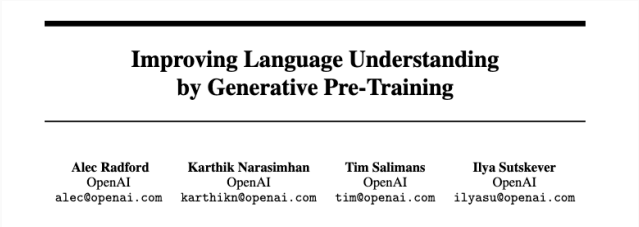
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

31st Conference on Neural Information Processing Systems, Long Beach, 2017

Modèles de type Transformer

Modèles de langage neuronaux



Improving Language Understanding by Generative Pre-Training

Alec Radford OpenAI alec@openai.com

Karthik Narasimhan OpenAI karthikn@openai.com

Tim Salimans OpenAI tim@openai.com

Ilya Sutskever OpenAI ilyasu@openai.com

Preprint OpenAI, 2018

ChatGPT-1

Modèles de langage neuronaux



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

North American Chapter of the Association for Computational Linguistics (NAACL)
Human Language Technology: Conference of the NAACL (HLT-NAACL), Minneapolis, 2019
Bidirectional Encoder Representations from Transformers

Modèles de langage neuronaux

Frise simplifiée

RNN $\Longrightarrow$ LSTM $\Longrightarrow$ Encoder-Decoder $\Longrightarrow$ Attention $\Longrightarrow$ **Transformers** $\Longrightarrow$ LLM ...

Question centrale

Qu'est-ce qui a changé après 2017 avec le papier ***Attention Is All You Need*** *?*

Apprentissage automatique - Machine Learning

L'apprentissage automatique (Machine Learning) est un domaine de l'intelligence artificielle qui consiste à apprendre des modèles à partir de données afin de

- prédire (classification, régression)
- décider (contrôle, stratégie)
- découvrir des structures (clustering, représentation)
- générer de nouvelles données (texte, images, sons...)

Apprentissage automatique - Machine Learning

Certains modèles ne se contentent pas de prédire, ils apprennent une distribution de probabilité des données, puis peuvent échantillonner de nouvelles observations

C'est le cas de modèles dits génératifs

- modèles autoregressifs (modèles de langage comme ChatGPT)
- modèles à densité explicite : VAE, normalizing flows
- modèles implicites : GAN
- modèles de diffusion (état de l'art actuel)

Exemples d'apprentissage automatique

Classification **prédire des classes discrètes** : filtrage de spam, détection d'objets (par ex. visage), prévision météo (pluie, neige, soleil), scoring de crédit...

Régression **prédire des valeurs numériques** : marché boursier, prévision météo (par ex. température)...

Classement (ranking) **comparer des éléments** : recherche web (mots-clés), étant donnée une image, trouver des images similaires...

Exemples d'apprentissage automatique

Clustering : regrouper des points, des images, regrouper des résultats de recherche web...

Embedding **encoder les données** : embeddings d'images, de mots, de tokens...

Prédiction structurée **données séquentielles** : reconnaissance de la parole/de l'image, traduction, traitement du langage naturel (NLP), LLM, ...

La science de la décision, **combinant apprentissage automatique et contrôle optimal** : AlphaGo, conduite autonome, robotique, ...

Quelques types d'apprentissage automatique

- Apprentissage supervisé
- Apprentissage non supervisé
- Apprentissage par renforcement
- ...

Apprentissage supervisé

Données étiquetées

$$(x_1, y_1), \ldots, (x_N, y_N)$$

Apprendre la relation entre y et x afin de prédire y à partir de x

$$\widehat{y} = \hat{f}(x)$$

Hypothèse clé, une relation stable existe entre les entrées et les sorties

Apprentissage supervisé

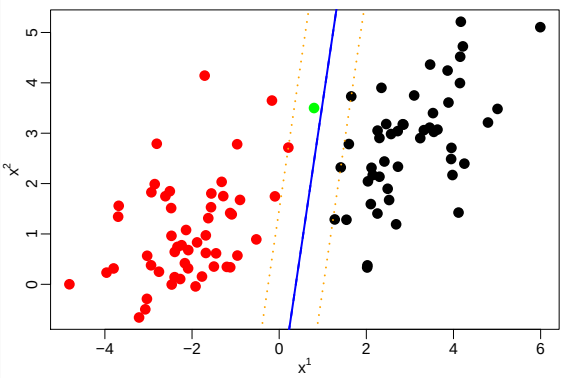


Figure 1: Un classifieur SVM

Apprentissage supervisé

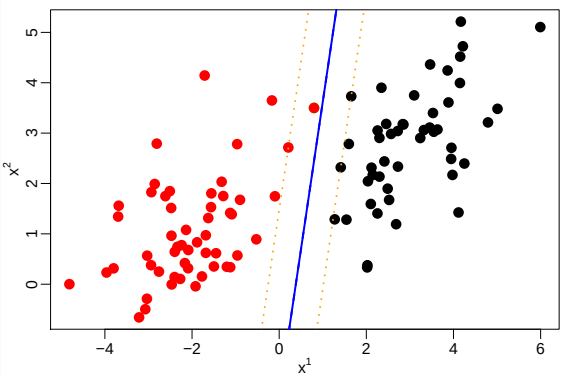


Figure 2: Le point vert est affecté à la classe rouge

Apprentissage non supervisé

Données non étiquetées

$$x_1, \ldots, x_N$$

Apprendre une représentation ou une structure des données

$$\hat{z} = g(x)$$

où z décrit une organisation des données (clusters, variables latentes, représentation)

Hypothèse clé, les données possèdent une organisation intrinsèque exploitable

Apprentissage non supervisé

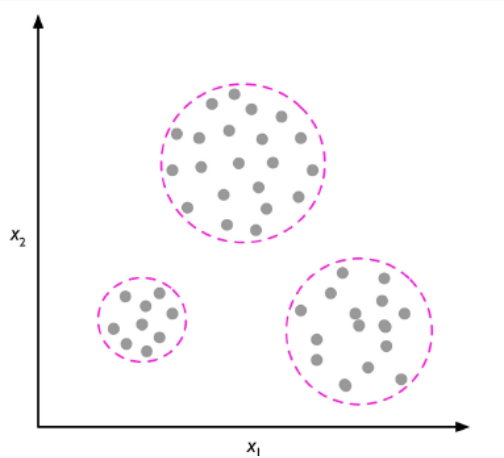


Figure 3: Clustering

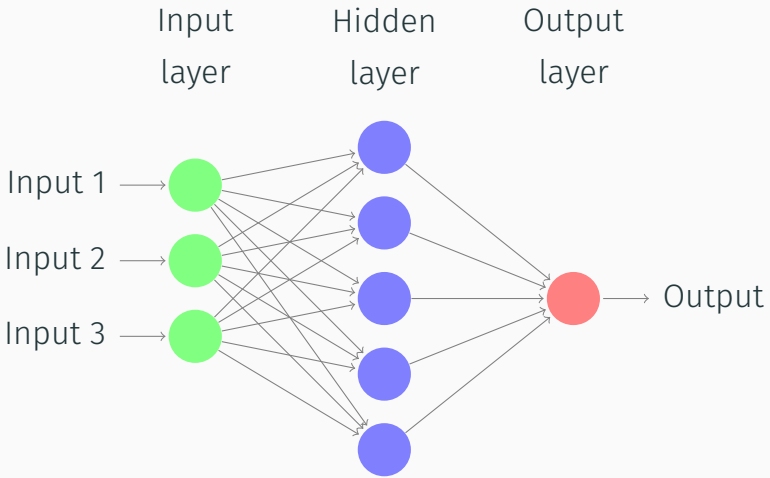
Réseaux de neurones

Combinent de nombreuses fonctions élémentaires pour former des fonctions complexes

Apprennent les relations entre ces fonctions simples à l'aide de l'échantillon d'apprentissage

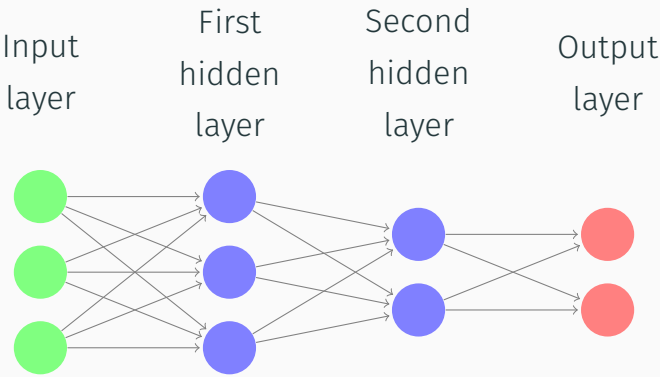
Réseaux de neurones

Exemple de réseau de neurones entièrement connecté avec une couche cachée



Réseaux de neurones

Exemple de réseau de neurones entièrement connecté avec deux couches cachées



Pourquoi la séquence est essentielle en langage

Le langage n'est **pas** un simple ensemble de mots

- **L'ordre** des mots compte
- **Le contexte** compte
- un même mot peut changer de sens selon les mots qui l'entourent

Le chien poursuit le chat
Le chat poursuit le chien
Chien le chat le poursuit

Pourquoi la séquence est essentielle en langage

Pour qu'une machine comprenne le langage, elle doit traiter les mots comme une **suite ordonnée**, et non comme des éléments indépendants

C'est pour cela qu'on a d'abord utilisé des **modèles séquentiels**

Les réseaux de neurones récurrents (RNN)

Un RNN lit une phrase **mot après mot**
À chaque nouveau mot, il met à jour une mémoire interne

J'aime l'apprentissage automatique

Le traitement se fait comme ceci

```
J'aime -> l'apprentissage -> automatique
```

À chaque étape, le modèle garde un état caché h_t qui résume ce qu'il a déjà lu

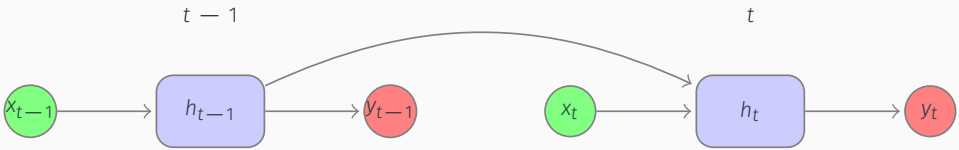
Les réseaux de neurones récurrents (RNN)

Architecture des RNN

- Les RNN ont une **boucle** : l'information du passé revient dans le calcul du présent
- Ils sont adaptés aux données séquentielles
  - langage
  - séries temporelles
  - parole

Ils ont permis les premiers progrès importants en traduction automatique, génération de texte, reconnaissance de la parole

Les réseaux de neurones récurrents (RNN)



$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

$$y_t = g(W_{hy}h_t + b_y)$$

Les réseaux de neurones récurrents (RNN)

- x_t représente le mot courant
- h_{t-1} contient un résumé du passé
- h_t combine les deux

Le modèle essaie de comprendre le mot courant en tenant compte de ce qui précède

Les réseaux de neurones récurrents (RNN)

Deux difficultés majeures apparaissent

- vanishing gradient : l'information ancienne s'efface
- exploding gradient : les calculs deviennent instables

Les réseaux de neurones récurrents (RNN)

Considérons la phrase

Le livre que j'ai acheté la semaine dernière dans une petite librairie du centre-ville est passionnant

Pour comprendre que *est passionnant* se rapporte à *le livre*, il faut se souvenir d'un mot apparu très tôt

Un RNN classique a souvent du mal à conserver cette information sur une longue distance

Les réseaux de neurones récurrents (RNN)

Les RNN comprennent assez bien les dépendances courtes, mais beaucoup moins bien les dépendances longues

Il fallait donc un modèle capable de

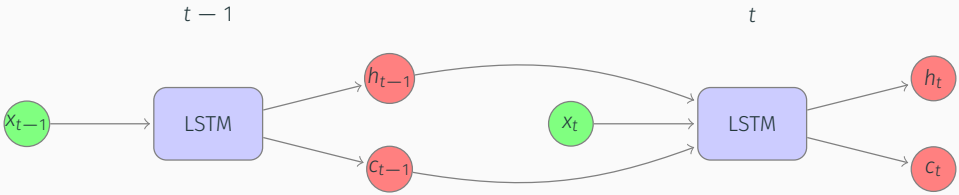
- mémoriser plus longtemps
- oublier sélectivement ce qui n'est pas important

Les modèles LSTM (Long Short-Term Memory)

Les **LSTM** ont été introduits en 1997 pour corriger les limites des RNN

Leur innovation centrale est l'utilisation de **portes (gates)**, qui contrôlent l'information

Les modèles LSTM (Long Short-Term Memory)



- **Forget gate** : décide quelle partie de la mémoire passée c_{t-1} est conservée ou oubliée
- **Input gate** : sélectionne l'information issue de x_t et h_{t-1} à intégrer dans la mémoire
- **Output gate** : filtre la mémoire interne pour produire l'état caché h_t transmis à la suite

Les modèles LSTM (Long Short-Term Memory)

Exemple

Le film était un peu long, mais la fin était excellente

Pour prédire le sentiment global, il est utile de retenir *excellente* plutôt que de donner trop de poids à *long*

Le LSTM peut apprendre à

- diminuer l'importance de certaines informations
- conserver celles qui seront utiles plus tard

Les modèles LSTM (Long Short-Term Memory)

Avantages

- mémoire sélective
- meilleure gestion des longues dépendances
- bons résultats en
  - reconnaissance vocale
  - traduction
  - analyse de sentiment
  - séries temporelles

Les modèles LSTM (Long Short-Term Memory)

Limites

- architecture plus complexe
- entraînement plus lourd
- toujours séquentiel
- donc difficile à paralléliser sur GPU

Les GRU (Gated Recurrent Units)

Les GRU sont une version simplifiée des LSTM

- moins de portes
- moins de paramètres
- entraînement plus rapide

Les GRU (Gated Recurrent Units)

LSTM : 3 portes, plus expressif, plus lourd

GRU : 2 portes, plus simple, souvent presque aussi performant

Avantages des GRU

- fusion des mécanismes d'oubli et d'entrée
- moins de paramètres
- convergence souvent plus rapide
- très utiles quand on veut un bon compromis entre performance et coût

Les GRU (Gated Recurrent Units)

Mais le problème fondamental reste le même

Comme les LSTM, les GRU traitent la séquence **un mot après l'autre**

- peu de parallélisme
- dépendances longues encore difficiles
- vitesse limitée sur très grands corpus

Le système Encoder–Decoder

Objectif : transformer une séquence en une autre

Exemples

- traduction automatique
- résumé de texte
- génération de réponse

Le système Encoder–Decoder

Principe

- **Encoder** : lit la séquence d'entrée
- **Decoder** : génère la séquence de sortie

$$(x_1, \ldots, x_T) \longrightarrow (y_1, \ldots, y_{T'})$$

Le système Encoder–Decoder

Idée clé

- L'encodeur transforme la séquence en une représentation interne
- Le décodeur utilise cette représentation pour produire la sortie

Une même architecture permet de traiter des tâches très différentes

Le système Encoder–Decoder

Étape 1 — Encodage

On transforme la séquence d'entrée en une suite de représentations :

$$h_1, h_2, \ldots, h_T$$

où chaque h_t contient de l'information sur le préfixe $(x_1, \ldots, x_t)$

L'ensemble des états $\{h_t\}$ constitue une représentation de la séquence

Encoder - RNN

Le système Encoder–Decoder

Étape 2 — Décodage

Le décodeur génère la sortie mot par mot.

À chaque étape, il s'appuie sur :

- ce qu'il a déjà produit
- son état interne courant
- l'information fournie par l'encodeur

Il prédit ainsi le prochain mot de la séquence

Le système Encoder–Decoder

Limite

Toute l'information est compressée dans un seul vecteur h_T

difficile pour les longues séquences

C'est ce problème qui motive l'introduction de la **self-attention**

Le tournant de 2017 : Attention Is All You Need

Le papier *Attention Is All You Need* change profondément la manière de traiter le langage

Le changement de paradigme

Les **RNN / LSTM / GRU** lisent les mots séquentiellement

Les **Transformers** analysent tous les mots simultanément

C'est là la rupture centrale

Le tournant de 2017 : Attention Is All You Need

L'idée clé : la self-attention

Quand un mot apparaît dans une phrase, de quels autres mots faut-il tenir compte pour le comprendre ?

La réponse des Transformers : chaque mot peut regarder tous les autres mots, et apprendre lui-même lesquels sont importants

C'est le mécanisme de **self-attention**

Comment fonctionne la self-attention

Le chat mange la souris

Pour comprendre *mange*, il faut regarder

- *chat* (qui agit)
- *souris* (sur quoi)

Comment fonctionne la self-attention

Mécanisme

Chaque mot fait trois choses :

- **Query (Q)** : Quelles informations me sont utiles ?
- **Key (K)** : Quel type d'information je propose aux autres ?
- **Value (V)** : Voici l'information que je transporte

Comment fonctionne la self-attention

Ce qui se passe

- chaque mot compare sa **Query** avec les **Keys** des autres mots
- cela donne des **poids d'importance**
- il combine ensuite les **Values** correspondantes

Un mot devient une **moyenne pondérée des autres mots**

Comment fonctionne la self-attention

Les scores sont obtenus par :

```
scores = (Q @ K.T) / sqrt(d_k)
attention = softmax(scores)
output = attention @ V
```

Sens des étapes

- Produit QK^T mesure à quel point un mot doit regarder un autre mot
- Division par $\sqrt{d_k}$ stabilise les valeurs numériques
- Softmax transforme les scores en poids positifs qui somment à 1
- Moyenne pondérée des (V) produit une nouvelle représentation du mot

Comment fonctionne la self-attention

Avant pour relier deux mots éloignés, il fallait transporter l'information de proche en proche

Après, avec le mécanisme de self-attention

- tout mot peut directement regarder n'importe quel autre mot
- les dépendances longues sont beaucoup plus faciles à capter
- l'entraînement peut être **parallélisé**

C'est le vrai point de rupture

Multi-head attention : plusieurs regards en parallèle

Les Transformers n'utilisent pas une seule attention, mais plusieurs en parallèle, on parle de multi-head attention

Intuition

Chaque tête peut apprendre un type de relation différent :

- une tête pour les relations syntaxiques
- une pour les dépendances sujet-verbe
- une pour les références pronominales
- une pour des relations sémantiques globales

Encodage de position

Un Transformer traite tous les mots en parallèle

Il faut donc lui donner une information sur l'ordre des mots

Sinon, les phrases :

- *Le chien mord le facteur*
- *Le facteur mord le chien*

auraient les mêmes mots, donc presque la même représentation brute

On ajoute à chaque mot un encodage de position.

Pourquoi les Transformers ont pris le dessus

- Ils permettent un parallélisme massif
- Ils passent très bien à l'échelle, ils supportent :
  - des jeux de données gigantesques
  - des modèles de millions ou milliards de paramètres
- Ils donnent de meilleurs résultats, en particulier pour :
  - longues séquences
  - compréhension contextuelle
  - génération de texte

Impact concret dans le monde réel

Traduction la qualité de la traduction automatique a fait un bond

Assistants conversationnels

- ChatGPT
- Claude
- Gemini ...

Recherche d'information

- Meilleure compréhension sémantique des requêtes

Impact concret dans le monde réel

Code

- GitHub Copilot
- auto-complétion intelligente
- génération et explication de code

Multimodalité

- texte + image
- texte + audio
- texte + vidéo

Ce qu'il faut retenir

- RNN : première vraie approche séquentielle, mais mémoire limitée
- LSTM : mémoire sélective grâce aux portes
- GRU : simplification efficace du LSTM
- Transformer : attention globale, parallélisme, changement d'échelle