ATTENTION IS ALL YOUR NEED REVIEW

Astegiano Nathan
Department ITI
Insa ROUEN
nathan.astegiano@insa-rouen.fr

Louis Dispa
Department ITI
Insa ROUEN
louis.dispa@insa-rouen.fr

Lucas Scellos
Department ITI
Insa ROUEN
lucas.scellos@insa-rouen.fr

May 18, 2021

ABSTRACT

Attention is all your need est sorti en **Décembre 2017** et a été écris conjointement par une équipe de 6 chercheurs de Google : Ashish Vaswani, Noam Shazeer, Niki Parmar, Llion Jones, Illia Polosukhin, Łukasz Kaiser, Jakob Uszkoreit ainsi que Aidan N. Gomez et Illia Polosukhin. Il a été présenté a la NIPS en 2017.
La conférence avait pour thème les processus d'information neuronale. D'après CORE, un portail de conférence, NIPS fait partie du top 7 % des conférences, ce qui lui donne un **rang d'A***, le plus haut possible. NIPS (aujourd'hui appelé Advances in Neural Information Processing Systems) est une conférence spécialisée dans le machine learning, l'intelligence artificielle ainsi que dans le traitement d'image.
L'article a ensuite été cité **18 658 fois**.
Nous pouvons donc penser que cet article est **fiable**.
Cet article s'intéresse au problème de **séquences vers séquences** (seq2seq).

Keywords Transformers · Attention · Machine Learning

1 En quoi cet article est-il innovant?

Cet article est particulièrement innovant car il est le fondateur du mécanisme de *multi-head attention*. Les modèles à l'état de l'art utilisent actuellement des Transformers, on peut citer notamment GPT-2 ou BERT.

2 Résumé

2.1 Problème des RNNs *ou pourquoi utiliser des Transformers*

Avant l'apparition des *Transformers*, les problèmes de type seq2seq utilisaient des modèles RNN pour l'encodage et le décodage. Cependant, dûe à leur architecture limitée, lors de la phase de décodage les RNN ne parvenaient pas à garder en mémoire l'**entièreté de la séquence** lors de l'ajout de nouveaux éléments. Ainsi, nous avions une **perte non négligeable** de signification sur cette dernière.

Cette perte peut avoir de lourdes conséquences notamment par exemple dans les problèmes de traduction où le début d'une phrase, la phrase précédente etc peut avoir une influence considérable sur cette dernière.

2.2 Les Transformers

Pour contrer le problème évoqué précédemment, notre article propose l'utilisation de *Transformers*. Ces derniers permettent d'éviter la récurrence et l'oubli des éléments les plus anciens de la séquence grâce à un **mécanisme d'attention** permettant d'établir des dépendances globales entre les entrées/sorties.

L'innovation des *Transformers* provient de l'utilisation unique d'un mécanisme de *self-attention* lors du calcul des représentations des entrées/sorties permettant d'extraire des features de ce dernier.

2.3 Self Attention

Comme énoncé dans la partie précédente, le mécanisme de *self-attention* permet d'intégrer de la compréhension au modèle ainsi que d'indiquer les tokens les plus pertinents vis-à-vis du token traité.

De plus, ce mécanisme permet de traiter la globalité les informations et plus seulement de manière séquentielle comme c'était le cas dans les anciens modèles.

Pour **chaque token** de la séquence, le mécanisme de *self-attention* va calculer l'attention sur **lui-même** ainsi que sur **tous les autres tokens**. Ce traitement va permettre de ressortir **Q : the Query, K : the Key et V : The Value**. Nous reviendrons sur ces 3 vecteurs dans la suite de ce document.

2.4 Architecture

2.4.1 Architectures des Transformers

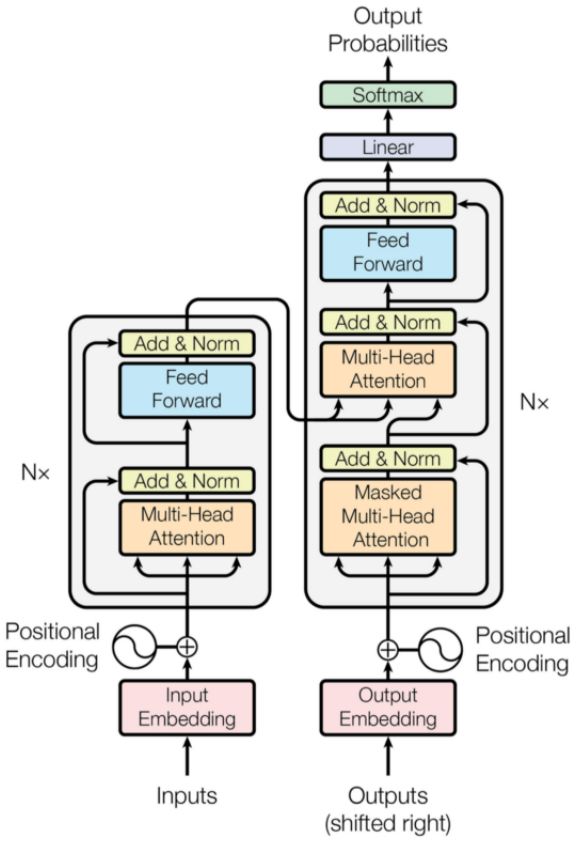

Figure 1: Architecture des Transformers

2.4.2 Encoder et Decoder

L'**encoder** est constitué de N = **6 stacks**. Chaque stack contient 2 couches :

- une couche de *self attention*
- une couche de *FeedForward Neural Network*

Le **decoder** se compose lui aussi de N = **6 stacks**. Cependant, cette fois, chaque stack est constituée de 3 couches :

- une couche de *self attention*
- une couche de *encoder-decoder attention*
- une couche de *Feed Forward Neural Network*

La couche intermédiaire *encoder-decoder attention* permet au decoder de savoir l'**importance** de chaque partie de la séquence.

2.4.3 Multi Head Attention

Cet article a pris le parti-pris d'utiliser **plusieurs layers d'attentions** s'exécutant en parallèle au lieu d'un seul de plus grande dimension. En effet, le fait d'utiliser plusieurs têtes, spécialisées chacune dans un type d'informations différent, permet de récupérer des informations d'attentions sur **plusieurs tokens**. Ces différentes têtes effectuent leurs calculs sur des bouts de mots et non des mots entiers. Cette séparation permet de donner un plus grand pouvoir de discrimination à l'ordre des bouts de mots.

Une fois ces différentes informations d'attentions récupérées, ces dernières sont concaténées et projetées une dernière fois.

The Query, The Value and The Key : Chaque vecteur d'entrée est utilisé de trois manières différentes dans le mécanisme de self attention :

- la requête *Q*
- la clé *K*
- la valeur *V*

Pour chaque rôle *Q*, *K* et *V*, une **matrice de poids** lui est associée. Ces 3 couches de poids sont ainsi appliquées sur le même vecteur d'entrée ce qui permet l'application du mécanisme d'attention sur le **vecteur d'entrée avec lui-même**.

Scaled Dot-ProductAttention : La fonction *Scaled Dot-Product Attention* retourne un score. Ce score reflète le **degré d'importance** à accorder à certains endroits ou à certains mots de la séquence d'entrée.

Le principe de cette fonction est le suivant :

- Étape 1 : calcul du produit scalaire entre *Q* et *K*, c'est dire le produit scalaire entre le vecteur de requête et le vecteur de clé du token respectif en cours de traitement (exemple : pour la position 1 nous aurions $\mathrm{q}_1.k_1, q_1.k_2, q_1.k_3 etc$).

- Étape 2 : mise à l'échelle afin d'obtenir des valeurs de gradient plus stable
- Étape 3 : application de la fonction `softmax`.
- Étape 4 : multiplication de ces résultats avec la matrice de valeurs *V*. Cette multiplication permet d'accentuer ou minimiser le focus sur les mots importants, respectivement inutiles, car la valeur associée à ces mots est importante, respectivement très faible.

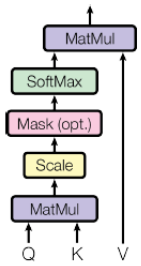

Figure 2: SDPA

Nous obtenons ainsi une fonction de la forme suivante :

$$Attention(Q, K, V) = softmax(\frac{Q^K}{d_k})V$$

2.4.4 Positional encoding

Un problème du mécanisme de *self attention* est qu'il ne prends pas en compte le positionnement des mots dans une phrase. Ainsi, deux phrases avec les mêmes mots mais dans un sens différent obtiendrait les mêmes résultats d'attention. Pour palier à ce problème, l'article propose l'utilisation d'un *positional encoding*. Ce dernier permet de **mapper** la position du mot à l'intérieur d'un vecteur. Le modèle apprendra ensuite à utiliser cette information et la rendre pertinente.

2.5 Résultat

2.6 Forces

Les Transformers peuvent être entraînés **beaucoup plus rapidement** que les architectures de réseaux récurrents ou des réseaux à convolution. De plus, ils résolvent des problèmes des LSTM ou RNN en permettant de donner plus de contexte au modèle, grâce aux mécanismes de *self-attention*.

Enfin, les modèles sont très facilement **ré-entraînables**.

2.7 Faiblesses

Plusieurs problèmes et désavantages sont cités dans l'article de recherche *Universal Transformers*. Le premier est que les *Transformers* sont limités sur les **tâches séquentielles**. En effet, l'absence de calculs en récursion est un problème pour la performance. De plus, le nombre de transformations possible est encadré par la **profondeur du modèle** ce qui a pour conséquence d'avoir, en général, des modèles de taille très importantes.

Ensuite, certains benchmarks pointent le fait que les *Transformers* de base ont une complexité en mémoire et en calcul en **0(n2)**. Il faut donc des machines de calculs importantes avec notamment l'utilisation de GPUs.

Enfin, pour certaines tâches, les *Transformers* sous-performent en comparaison des RNN, on peut notamment citer les benchmarks de l'article *The Importance of Being Recurrent for Modeling Hierarchical Structure*.

Pour finir, on retrouve de nombreux biais en fonction du training set.

References

Universal Transformers, MostafaDehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser, 2019

Language Models are Unsupervised Multitask Learners, Alec Radford _ 1 Jeffrey Wu_ 1 Rewon Child 1 David Luan 1 Dario Amodei ** 1 Ilya Sutskever

BERT: Pre-training of DeepBidirectional Transformers for Language Understanding, Ke Tran, Arianna Bisazza, Christof Monz, 2018

The Importance of BeingRecurrent for Modeling Hierarchical Structure,Ke Tran, Arianna Bisazza, Christof Monz

Review lors du NIPS sur l'article.