Les transformers

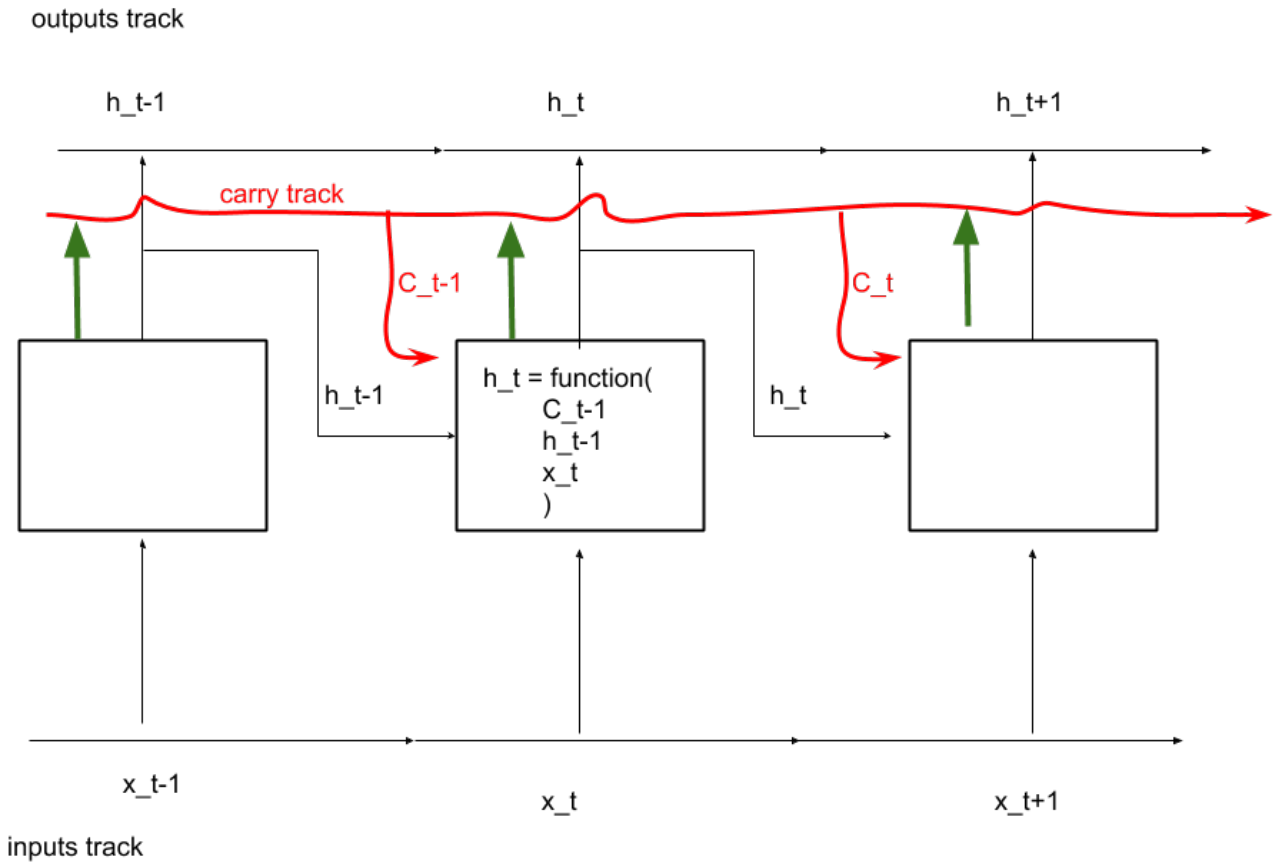
Historique

Traitement du langage naturel = NLP c'est

- la traduction
- chatbot
- la création de titres ou de résumés
- l'analyse des sentiments (ex: détection de commentaires haineux)
- la création de légendes d'images
- La création d'image à partir de légence

Techniques:

1. RNN (Recurent neural Network). 1997. Avantage: longue mémoire



2. RNN + Mécanisme d'attention. 2014. Avantage: Pour produire une sortie, le réseau se focalise sur la ad hoc de l'entrée.
3. Transformer: supprime le RNN, garde seulement l'attention. 2017. Avantage: l'attention est un mécanisme "matriciel" et non pas "séquentiel". Plus facile à paralléliser. Moins de perte d'information.

Gigantisme

Les transformers les plus célèbres sont :

- GPT3 d'openAI qui est utilisé dans ChatGPT
- Bert de Google (devenu maintenant Bart)
- RoBERTa (basé sur Bert) de facebook
- Camenbert (basé sur Bert) dont la particularité est ...

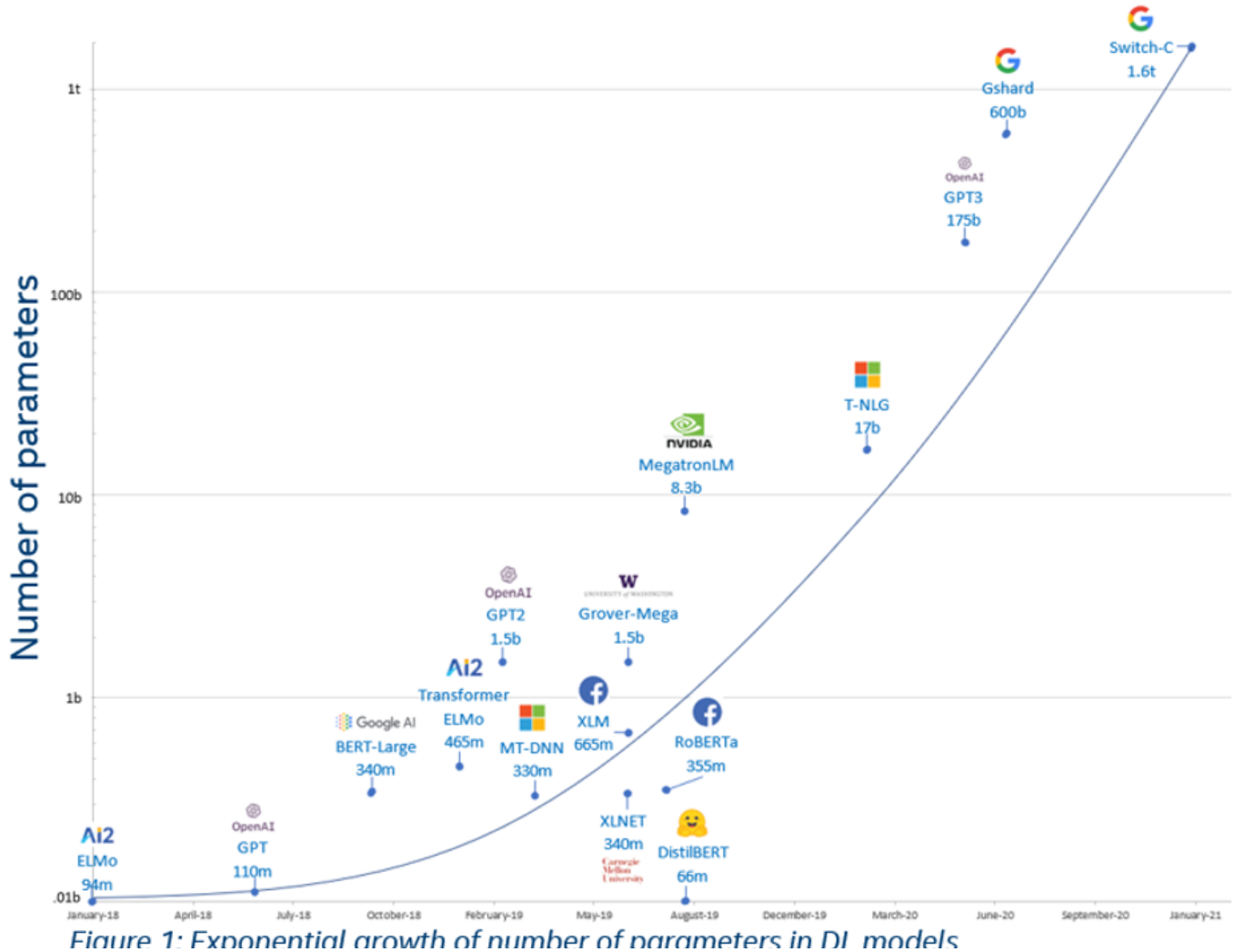


Figure 1: Exponential growth of number of parameters in DL models

Figure 1: Exponential growth of number of parameters in DL models

Embeding Word2Vec

C'est l'étape préliminaire: Les modèles mathématiques ne connaissent que des nombre. Donc chaque mot du dictionnaire doit être transformer en un *vecteur* de grande dimension:

```
a      → [3.5 2.7 8.8 0.5 5.7 9.1 2.6 4.5 2.1 0.8 7.3]
about  → [2.1 2.1 1.3 3.5 4.7 2.1 3.2 3.1 1.8 9.4 3.8]
absurd → [3.2 5.2 6.1 9.1 3.1 2.7 3.0 1.6 3.4 6.6 2.7]
...
```

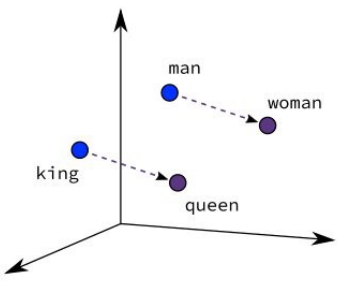
Ces vecteurs sont des paramètres entrainables. Ils vont être modifiés au fur et à mesure de l'entrainement.

Remarque: maintenant on travaille plutôt avec des sous-mot (sorte de syllabes) que des mots.

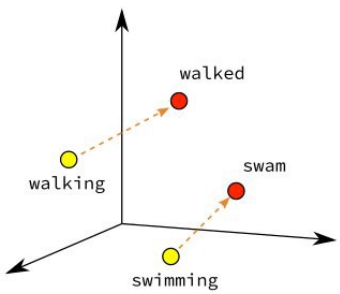
```
mang + er/ez/eons/ères...
viol + ent/ente/emment...
```

Les relations sémentiques entre mot vont progessivement se transformer en des relations vectorielles

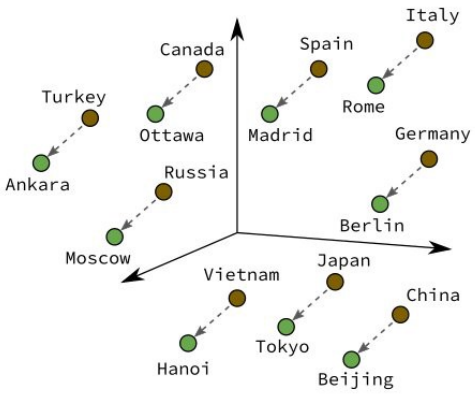
Les relations sémentiques entre mots sont nombreuses et diverses, mais les directions vectorielles, en grande dimension, sont aussi très nombreuses ; le dessin ci-dessous, en dimension 3 (seulement), n'est pas très réaliste.



Male-Female



Verb Tense



Country-Capital

Traducteur

C'est pour cela qu'à été inventé le mécanisme d'attention

La traduction s'effectue en 2 phases:

- **Encodage:** La phrase source (en anglais) est transformée en une séquence qui a du sens pour l'IA. Cette séquence sert de contexte au décodage:
- **Décodage:** La phrase en français est construite mot à mot à l'aide d'un prédicteur qui se nourit des prédictions passées et du contexte encodé.

Chatbot

C'est la moitié d'un traducteur (le décodeur).

C'est simplement un modèle qui prédit le prochain mot d'un texte. Exemple:

1. un utilisateur humain pose une quesion de 10 mots: $X_0, \ldots, X_9$.
2. Le modèle prédit alors un mot suivant X_{10}.
3. On rentre dans le modèle la nouvelle entrée $X_0, \ldots, X_9, X_{10}$
4. Le modèle prédit le mot suivant X_{11}
5. etc... jusqu'à ce que le modèle prédise un symbole spécial `[END]`

En pratique

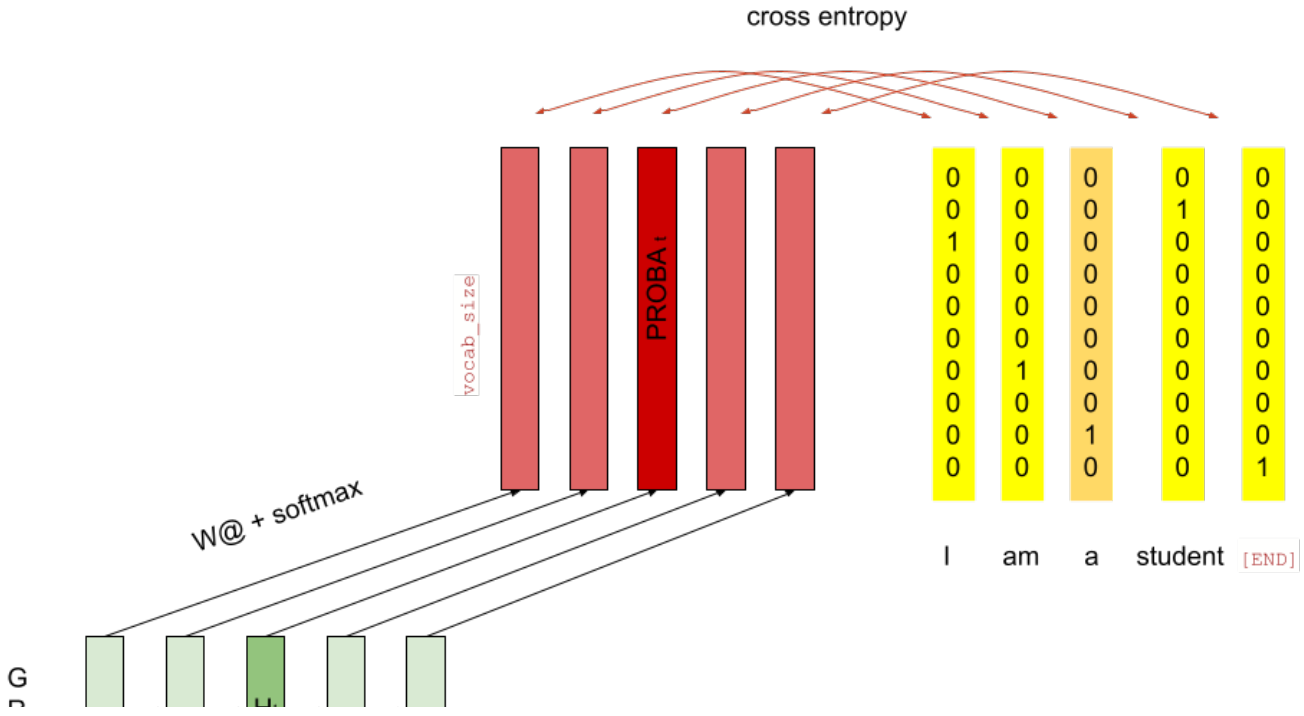
Entrée	X_0	X_1	X_2	X_3	X_4
	[START]	I	am	a	student
Sortie	Y_0	Y_1	Y_2	Y_3	Y_4
	I	am	a	student	[END]

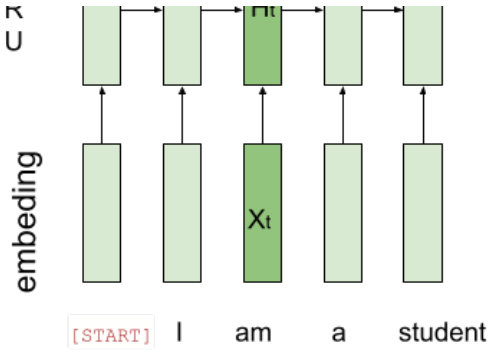
On entraine le modèle à passer directement de la séquence X à la séquence Y (un simple décalage de X).

Il faut absolument mettre une contrainte de non anticipation:

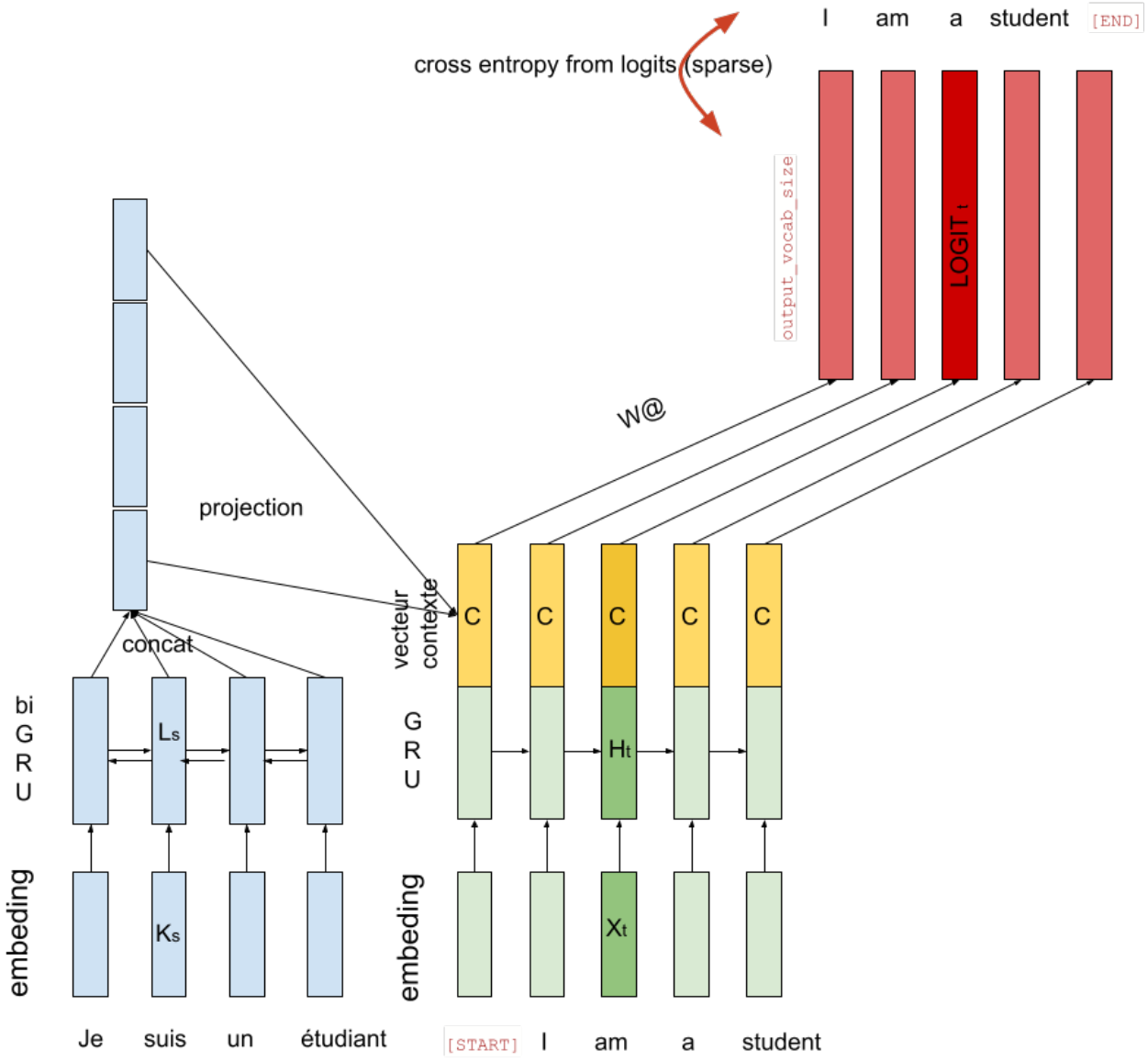
$$Y_t = fonction(X_0, \ldots, X_t)$$

Architecture de chatbot





Architecture de traducteur



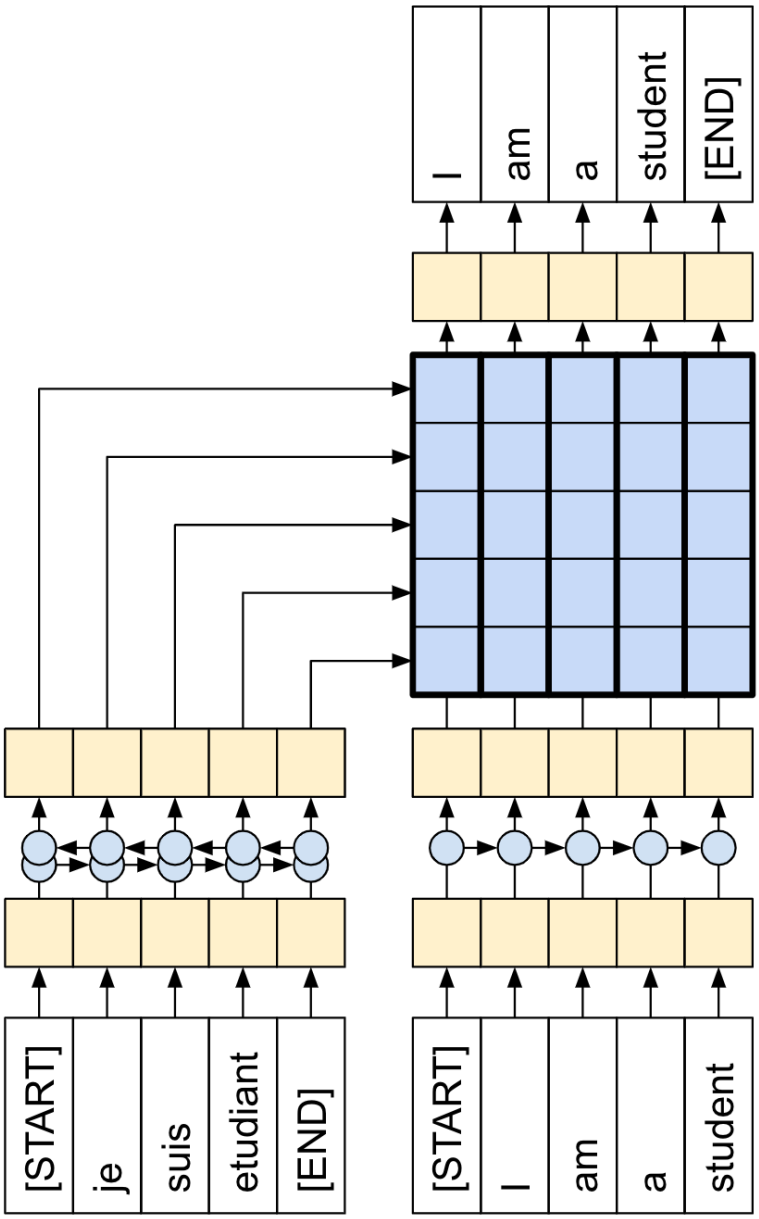
Problème: le contexte est global, il faudrait le particularisé à la prédiction qu'on est entrain de

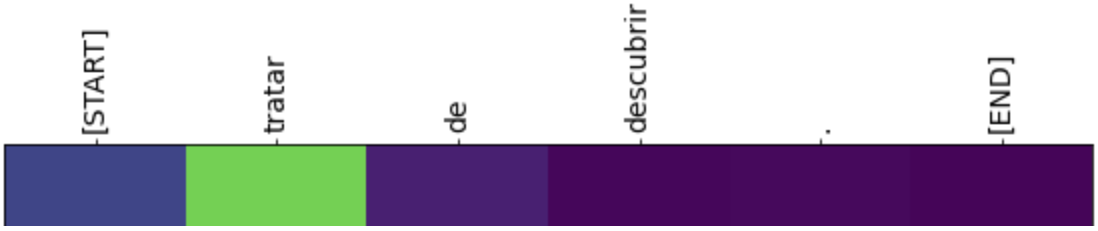
faire

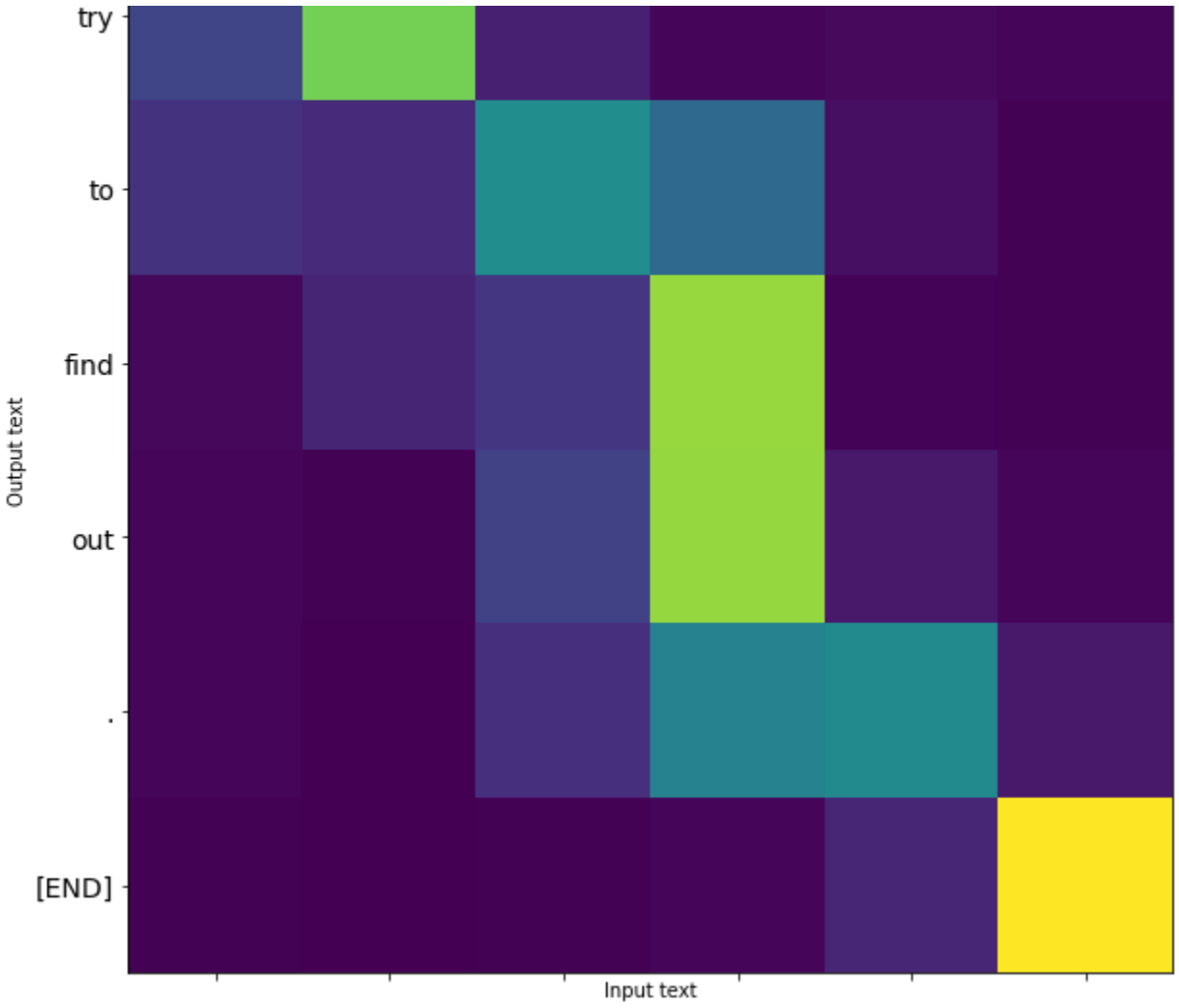
Arrive l'attention

C'est une matrice qui va pondérer le contexte

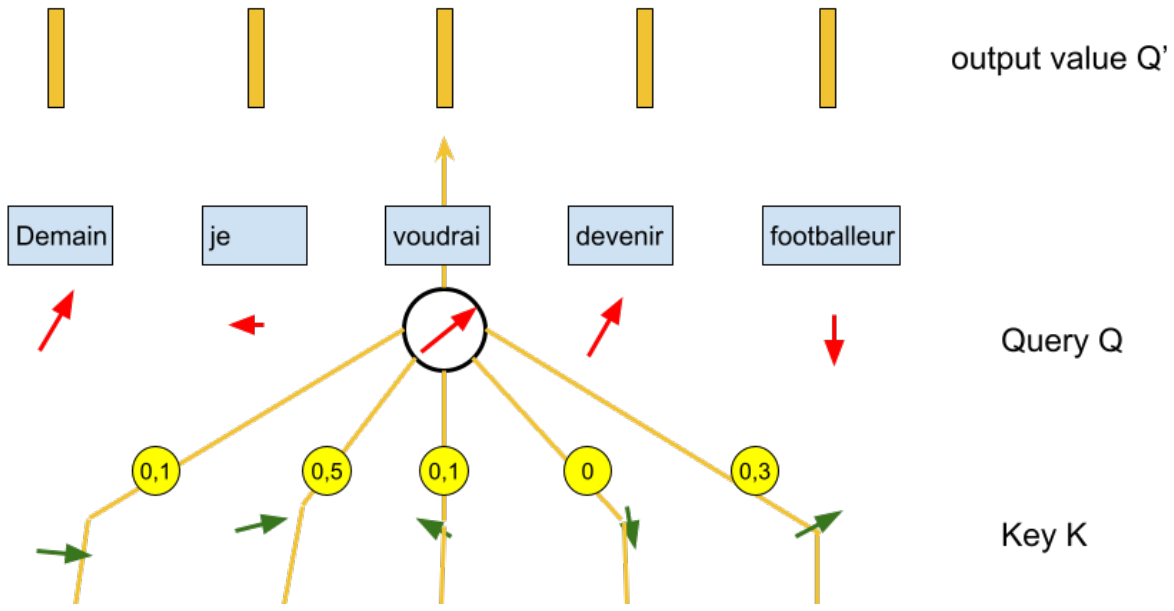
The [RNN+Attention model]

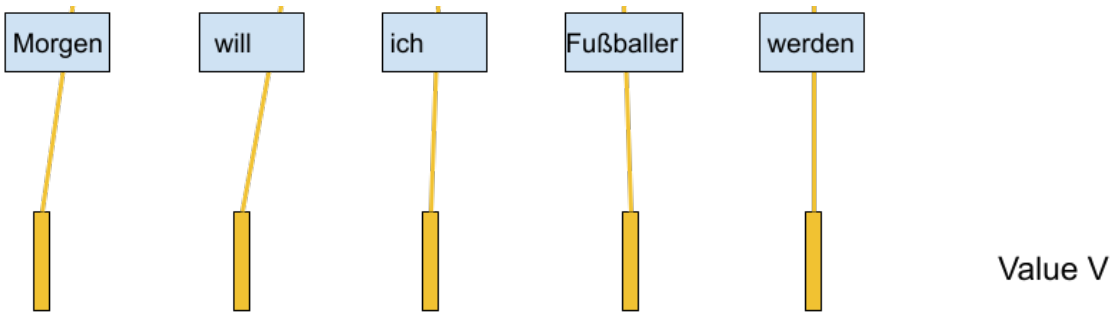






L'attention produit scalaire

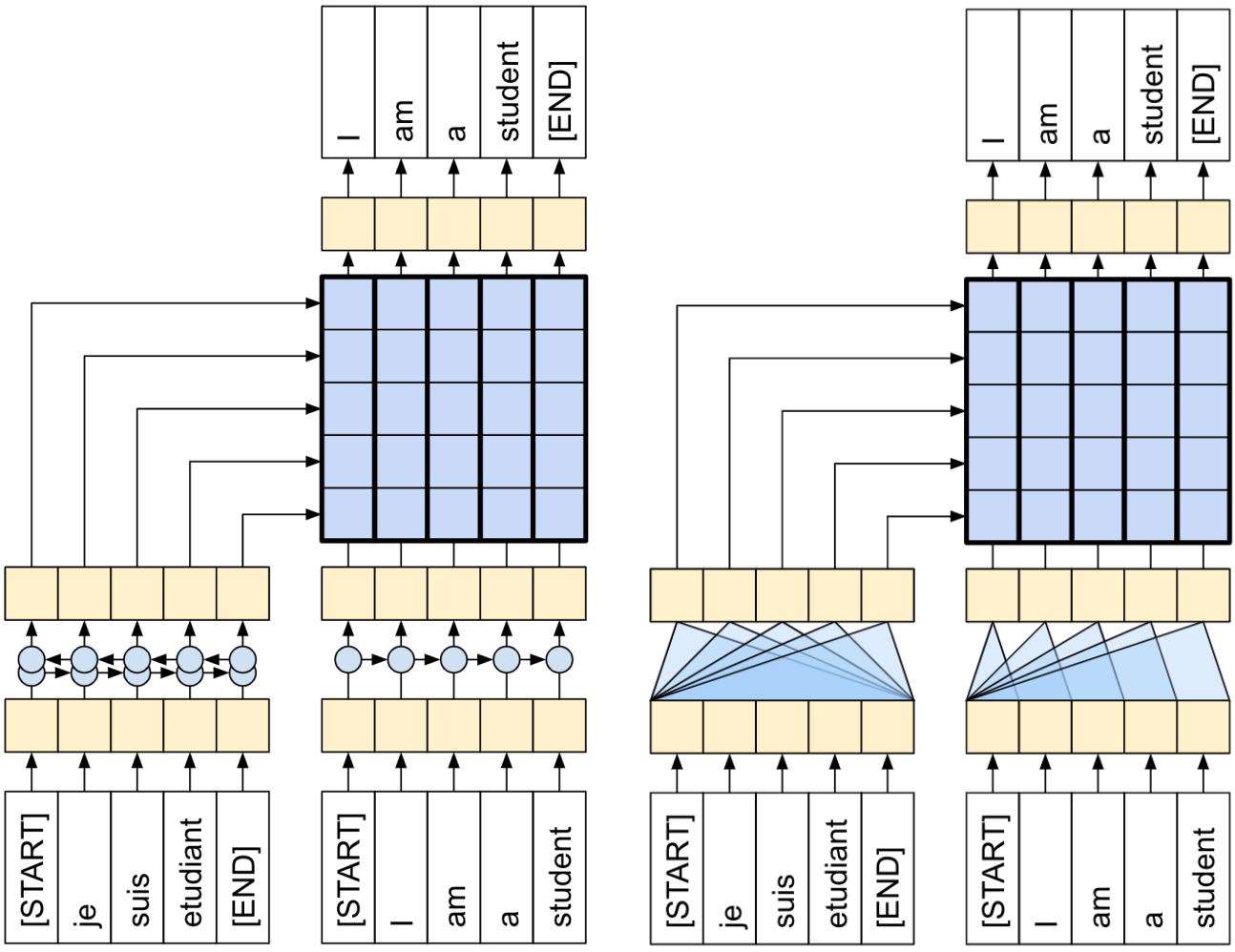




Transformers

On utilise aussi l'attention pour l'analyse des phrases dans les langues source et cible.

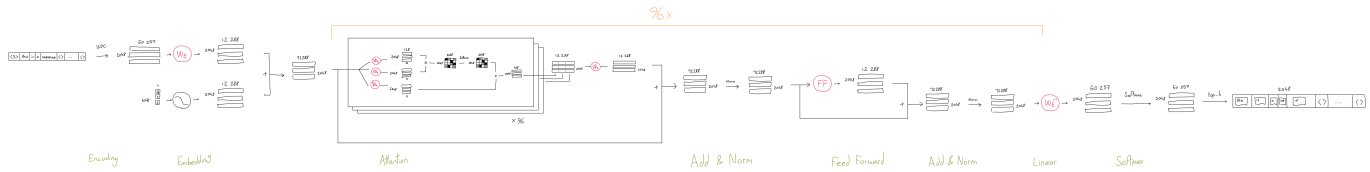
The RNN+Attention model **A 1-layer transformer**



Comme d'hab, un transformer-chatbot, c'est seulement la partie à droite (le décodeur)

Chat GPT de la première à la dernière couche

Ci-dessous un schéma complet du modèle GPT. J'ai fait confiance au tuto ici qui m'a charmé par ses dessins très clairs.

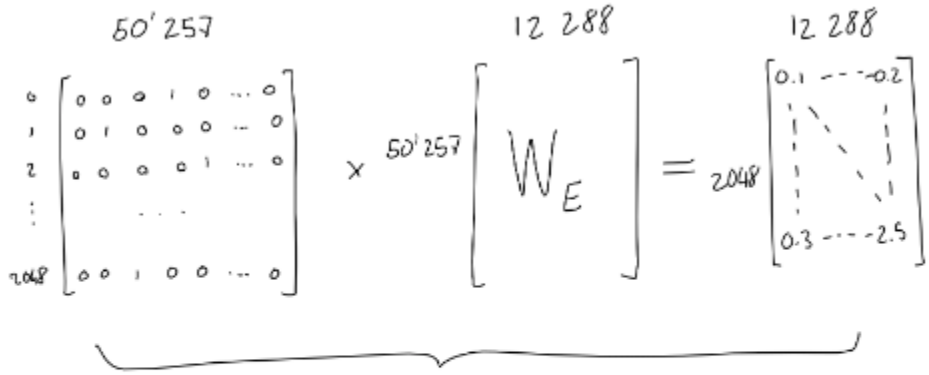


Plongement

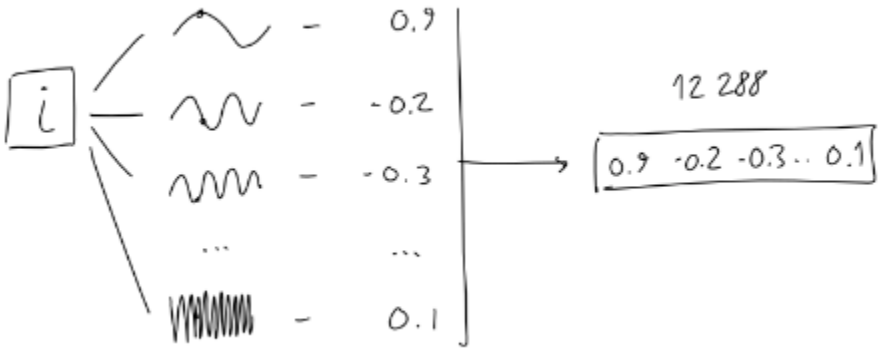
Les phrases ont une longueur de 2048 mots.

Le vocabualire est de 50257 mots.

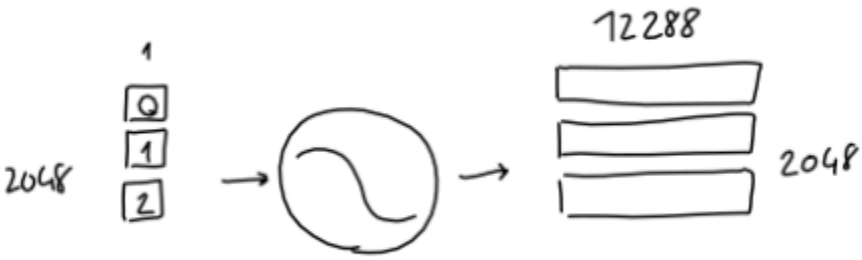
Les mots sont encodé en des vecteurs de taille 12288.



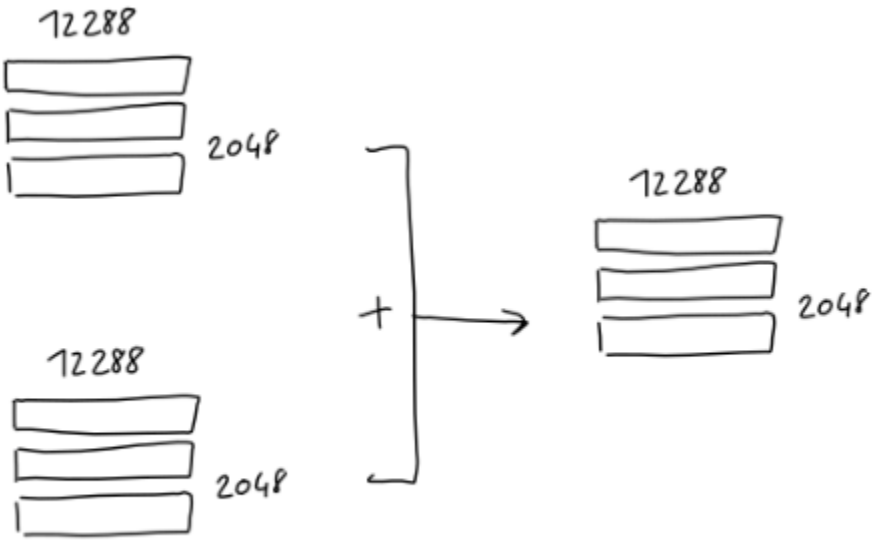
Encodage positionnel



Chacun indice de position est passé dans 12288 sinusoides.

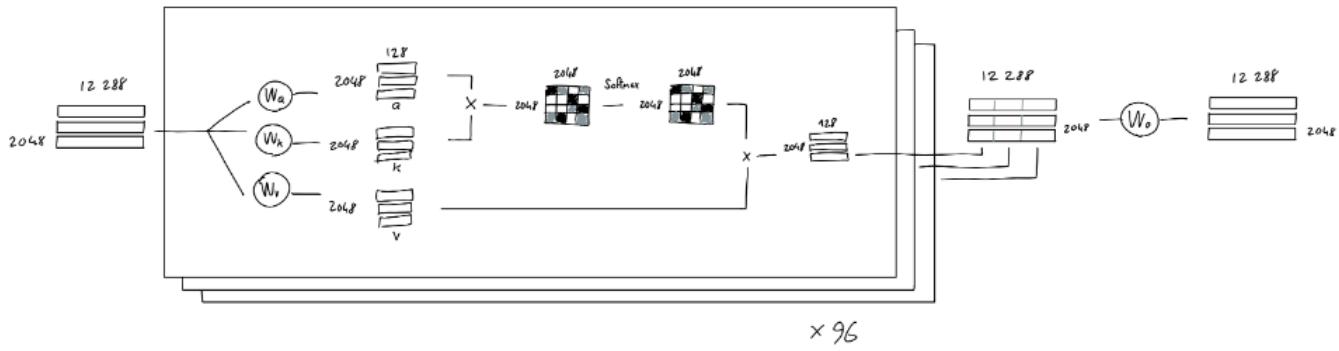


Ainsi la suite de toutes les position: le vecteur 0,1,...2047 est encodé en une matrice 12288*2048



Cette matrice est ensuite additionnée avec la matrice représentant une phrase

Attention multi-tête



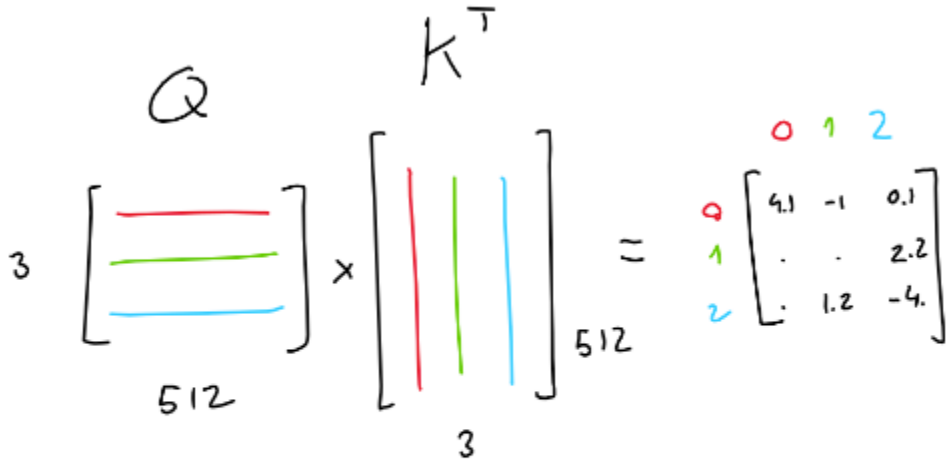
Le nombre de tête est de 96.

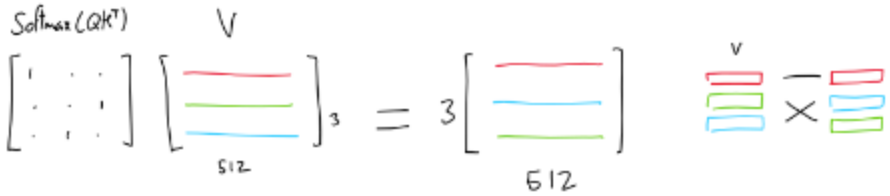
Pour chacune des têtes, la head_depth est de 128.

Les 96 résutats des têtes sont concaténées entre elle en une matrice de taille 2028*12288, puis mélangées par une multiplication matricielle qui garde la même shape.

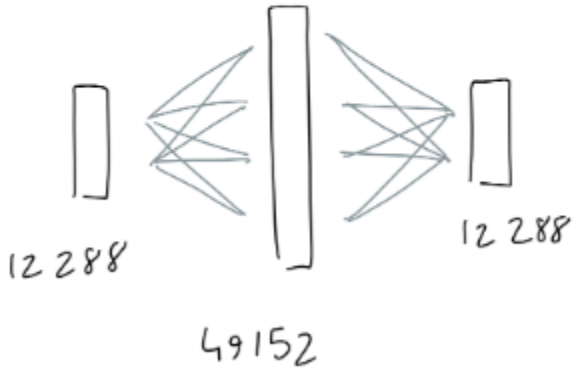
Juste l'attention

Le dessin que fait l'auteur de l'attention est très sympa aussi. Mais attention, là les tailles ne sont pas les vraies longeur séquence seulement 3, et depth seulement 512.





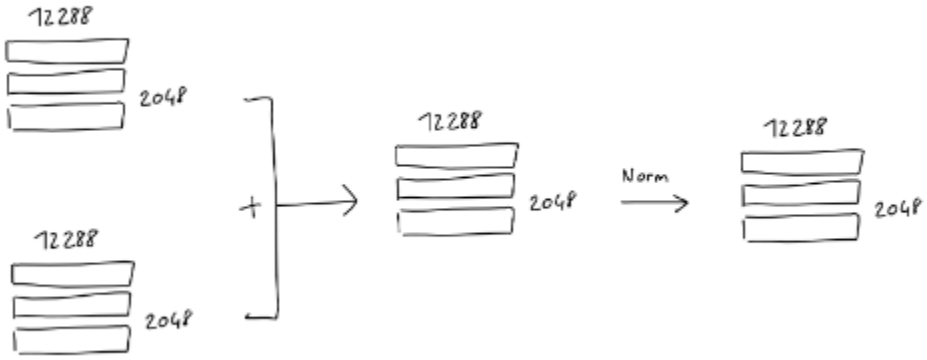
Feed wordward



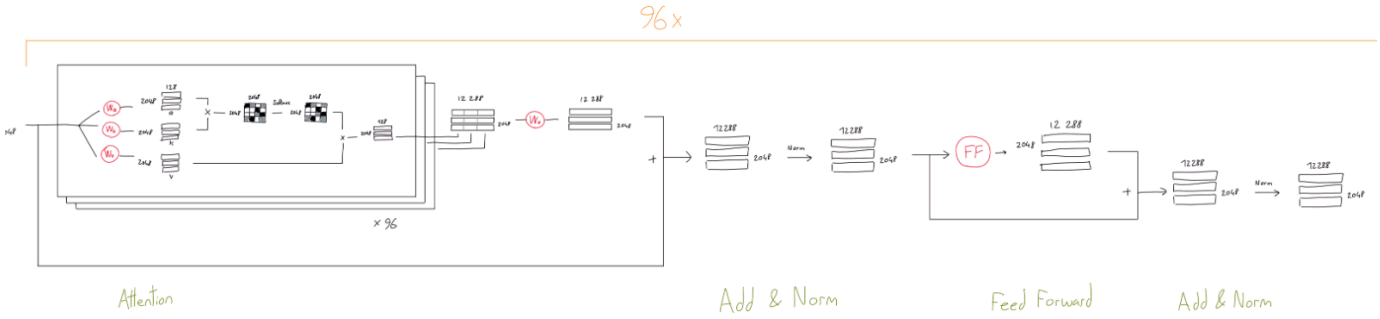
C'est un réseau à une couche caché, dont la taille est de 49152. Les fonctions d'activation `relu`.

On ajoute bien des biais même si ce n'est pas dessiné.

Add et norm



Empilement des blocks



Le block est répéter 96 fois.

Décodage final

La sortie du dernier block est transformé une suite de logits par une multiplication matricielle.

Au total 175 Milliards de paramètres pour GPT3.

Pour GPT4 on ne sait pas. Des rumeurs disent 100 000 Milliards. Mais c'est sans doute plutôt de l'odre de 1000 Milliards.