Projet 7 - Développez une preuve de concept

Sources du projet : https://github.com/ben74/xlnet_poc/

Table des matières

---

1) Problématique et interprétation 2

2) Plan de travail 2

3) Baseline 3

4) Méthodologie, preprocessing, optimisation 5
- Pré-processing des données 5

5) Limitations du modèle 7

6) Nouvel algorithme : XlNet 8
- A) Tokenizer 9
- B) Attention 9
- C) LTSM vs Transformers 12
- D) Transformers 13
- E) Bert - modèles similaires et associés 15
- F) Transformer XL 15
- G) XLNet 16
- H) Performances comparatives 17

7) Mise en oeuvre 18
- Floating Point 16 18

8) Conclusions 19
- A) Scoring 20
- B) Seuillage 20
- C) Avantages 21

9) Pistes d'amélioration / approfondissement 21

10) Références / Bibliographie 23

---

1) Problématique et interprétation

Pour ce projet il est demandé de développer une preuve de concept, d'appliquer un nouvel algorithme par dessus une baseline.

Le domaine choisi est le NLP (Natural Language processing ou traitement du langage naturel) pour une classification de texte.

La baseline retenue est celle du projet n°5 : catégorisation des posts stack overflow au travers de multiples labels

Le site stackoverflow et ses nombreux satellites par thèmes propose à ses utilisateurs depuis de nombreuses années de poser des questions liées principalement à la programmation à une communauté de personnes qui apportent leurs réponses. Ce dernier a gagné en popularité par son système de votes sur les réponses, ainsi qu'un tagging approprié qui sert énormément aux scores de pertinence pour les moteurs de recherches. Le but à atteindre est de prédire ces tags vis à vis du titre et corps de texte constituant la question d'un utilisateur.

---

2) Plan de travail

Démarré le 16 avril 2020 : retenir une baseline et un modèle "State of the Art" en l'espace de 5 jours.

Pour ce faire j'ai notamment exploré le site https://paperswithcode.com/ notamment sur la partie text classification :

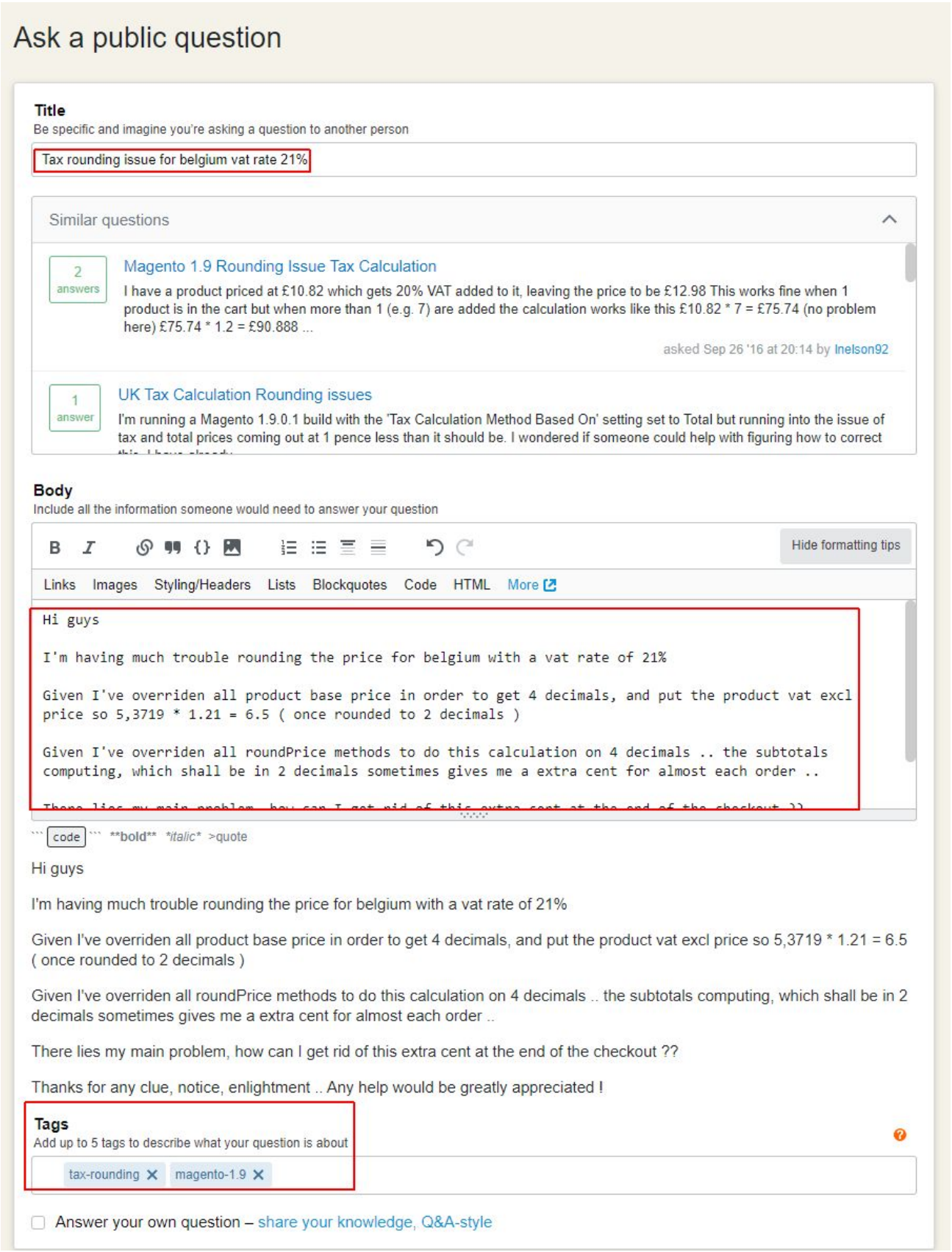
https://paperswithcode.com/task/text-classification / multi label text classification : https://paperswithcode.com/task/multi-label-text-classification

La récurrence de l'apparition de xlnet publié le 19 Juin 2019 semblait évidente. Le modèle est implémenté au travers de 3 frameworks : transformers, tensorflow et pytorch. Plusieurs notebook y faisant référence. Cela constitua mes critères de choix pour l'algorithme et méthodes à retenir. Il ne restait alors plus qu'à tester pour la première fois ce modèle

---

3) Baseline

Prédire les tags associés à une question sur le site stack overflow à partir son contenu. Ce problème a été abordé dans le projet n°5 et résolu via un algorithme Logistic Regression sur lequel est appliqué un classifieur exclusif "OneVsRestClassifier" avec des données d'entrées linéarisées sous une forme Bag Of Words binaire, en matrice disséminée (sparse) afin de cumuler une bonne classification avec de bonnes performances algorithmiques.



Le jeu de données a été récupéré depuis cette adresse :

https://data.stackexchange.com/stackoverflow/query/new

à l'aide d'une requête sql effectuée plusieurs reprises :

```
select top 50000 id,AnswerCount,score,Title,Body,tags from posts where Title<>'' and tags<>'' and AnswerCount>0 and id<500000000 order by id desc
```

puis récupérés sous forme de fichiers csv, agrégés et nettoyés sur un fichier final de 170000 enregistrements randomisé et séparés en un jeu d'entraînement et de test distinct (afin de comparer les performances de ces derniers) disponible à cette adresse : http://1.x24.fr/a/jupyter/stack5/df.tgz

---

4) Méthodologie, preprocessing, optimisation

Pour l'estimation des performances des modèles nous allons utiliser plusieurs indices : l'indice de jaccard ainsi, le score f1 (combinaison de précision : (TP/FP) / rappel (TP/FN)), ainsi que le score roc auc qui présentera moins d'écart afin de comparer ces derniers.

1) Pré-processing des données

Le texte est nettoyé en utilisant des expressions régulières judicieuses (fonction extractionMots) puis converti en bag of words qui prend en compte les 5000 mots les plus récurrents de ce dataframe (une fois nettoyé). Les données cibles à prédire sont la présence parmi les 358 tags les plus récurrents, ce qui en fait un problème de classification multi-labels, ces dernières sont labellisées dans une matrice similaire au bag of words.

Passage du texte en minuscules, caractères	\n\n 1.\xa0\xa0\xa0\xa0\xa0\xa0 > the patïents résponsès are âggrëgated. for EXAMPLE,!

unicodes et sauts de lignes	
Conversion des accents	1. > the patïents résponsès are âggrëgated. for example,!
Lemmatization	1. > the patients responses are aggregated. for example,!
Stopswords	1. > the patient responses be aggregated. for example,!
Retrait des tags HTML en conservant leur corps	1. >patient responses aggregated. example ,!
Entités HTML	1. > patient responses aggregated. example ,!
Autres que a-z 0-9 et ponctuation	1. patient responses aggregated. example ,!
Retrait de toute ponctuation	1 patient responses aggregated example

L'ensemble du corpus du jeu d'entraînement est ainsi nettoyé puis splitté afin de compter les occurences de mots les plus récurrents et d'en établir un dictionnaire des 5000 mots les plus récurrents afin de convertir ces extraits en "bag of words".

```
array([[0., 1., 0., 1., 1., 0., 1., 1., 1., 0., 0., 1., 0., 0., 1., 0.,
        1., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 1.,
        1., 0., 0., 0., 0., 1., 0., 0., 1., 0., 1., 0., 0., 0., 1., 1.,
        0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
```

L'utilisation d'une matrice "sparse" disséminée permet en mémoire de ne prendre en compte que l'information sur forme de coordonnées, ce qui permet moins d'utilisation de mémoire, et un entraînement bcp plus rapide des modèles. Le modèle précédemment retenu est une régression logistique imbriquée dans un meta classifieur OnevsRest

Sur ce dernier a été calculé sur un dictionnaire des 5000 mots les plus récurrents au sein d'un post, puis implémenté sur le meilleur seuil (cutoff) de 0.79 calculé sur les données d'entraînement pour un score roc global de 0.82 déterminé à partir des prédictions sur le jeu de test.

Les hyper paramètres retenus par une validation croisée est parmi ces derniers :

Hyper Paramètres	Valeurs possibles	Meilleur choix
- class_weight	balanced ou none	balanced
- solver	['sag','liblinear','lbfgs']	liblinear
- penalty	['l2','l1','elasticnet','none']	l1
- C	[0.01,0.1,0.5,1,5,10,50,100,500]	500

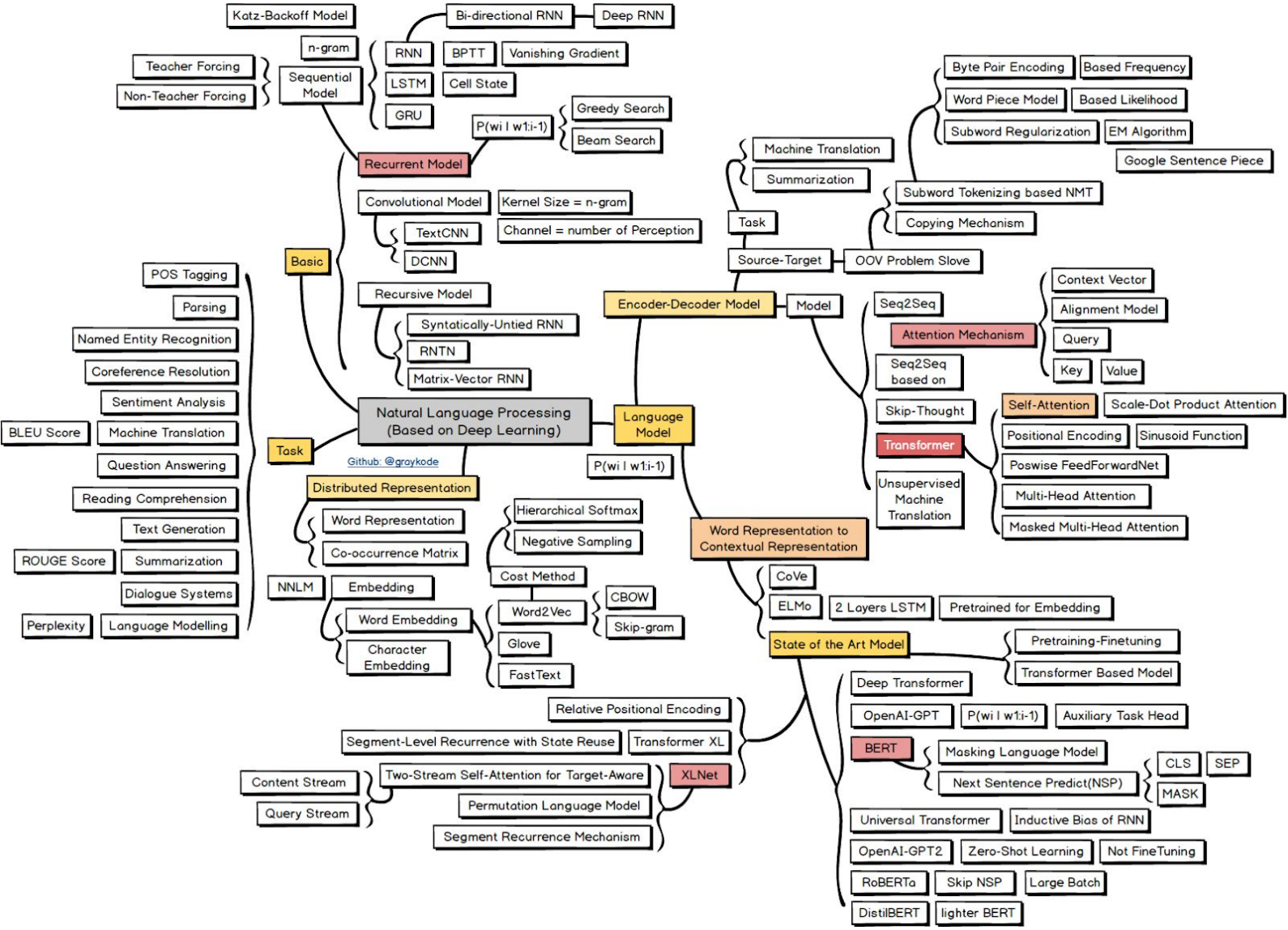
---

5) Limitations du modèle

En dépit d'être rapide et performant, le modèle bag of words / Logistic Regression ne dispose d'aucune sensibilité de compréhension du texte, qu'elle soit grammaticale ou linguistique. Ce dernier se contente de "vectoriser" des mots et d'assigner des probabilités statistiques de classification. Il ne pourra donc en conséquence ne pas comprendre les nuances et attentions d'un texte.

6) Nouvel algorithme : XlNet

Xlnet : il s'agit d'un modèle NLP héritant des améliorations de transformers XL, de type Encoder-Decoder avec Attention (non RNN), bidirectionnel.



mindmap des modèles nlp @github-garykode

L'algorithme dans sa version large a été entraîné sur plus de 113 go de données, 16go dans sa version de base

360 millions de paramètres en version "large", 116 millions en version base avec 192 mots pris en compte par séquence

```
Model: "xlnet"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_word_ids (InputLayer)  [(None, 192)]             0
_________________________________________________________________
tfxl_net_model (TFXLNetModel ((None, 192, 1024),)      360268800
_________________________________________________________________
tf_op_layer_strided_slice (T [(None, 1024)]            0
_________________________________________________________________
dense (Dense)                (None, 1)                 1025
=================================================================
Total params: 360,269,825
Trainable params: 360,269,825
```

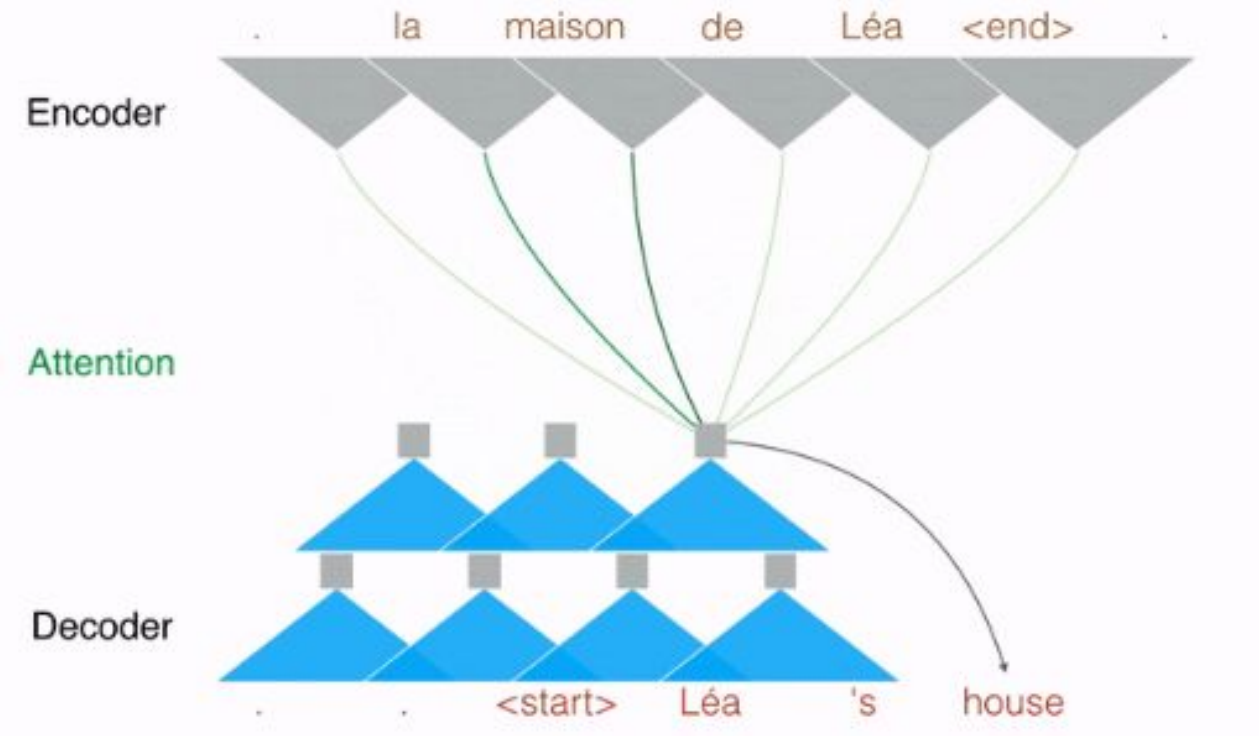
A) Tokenizer

Tout d'abord les données passées en entrée sont issues d'un tokenizer, qui attribue un chiffre à chaque "mot", chaque phrase ainsi est retournée sur la forme d'une matrice de chiffre ayant une longueur maximale pré-établie, si cette longueur n'est pas atteinte, des valeurs 0 seront ajoutées à la fin afin de correspondre à la taille en entrée du réseau de neurones, si cela est plus, le contenu sera alors tronqué.

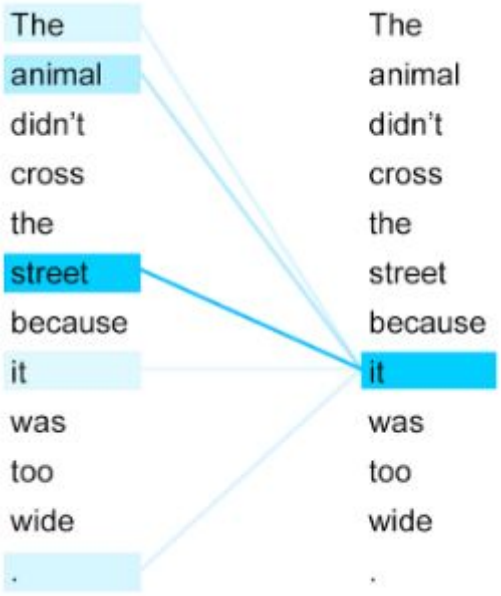
B) Attention

Mécanisme d'attention : permet notamment la permutation grammaticale lors de la traduction d'une langue à une autre, cette valeur est passée à nouveau en entrée de l'encodeur à chaque étape

Séquentiellement

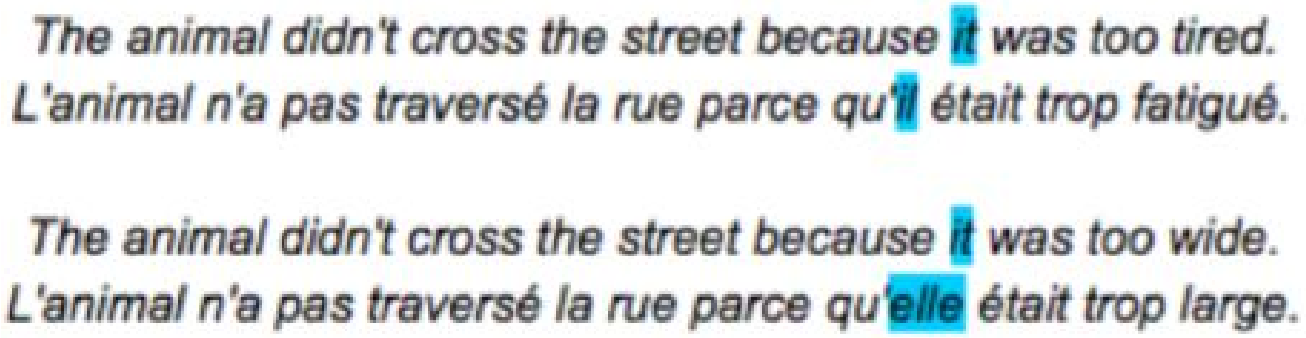


https://medium.com/analytics-vidhya/transformer-vs-rnn-and-cnn-18eeefa3602b



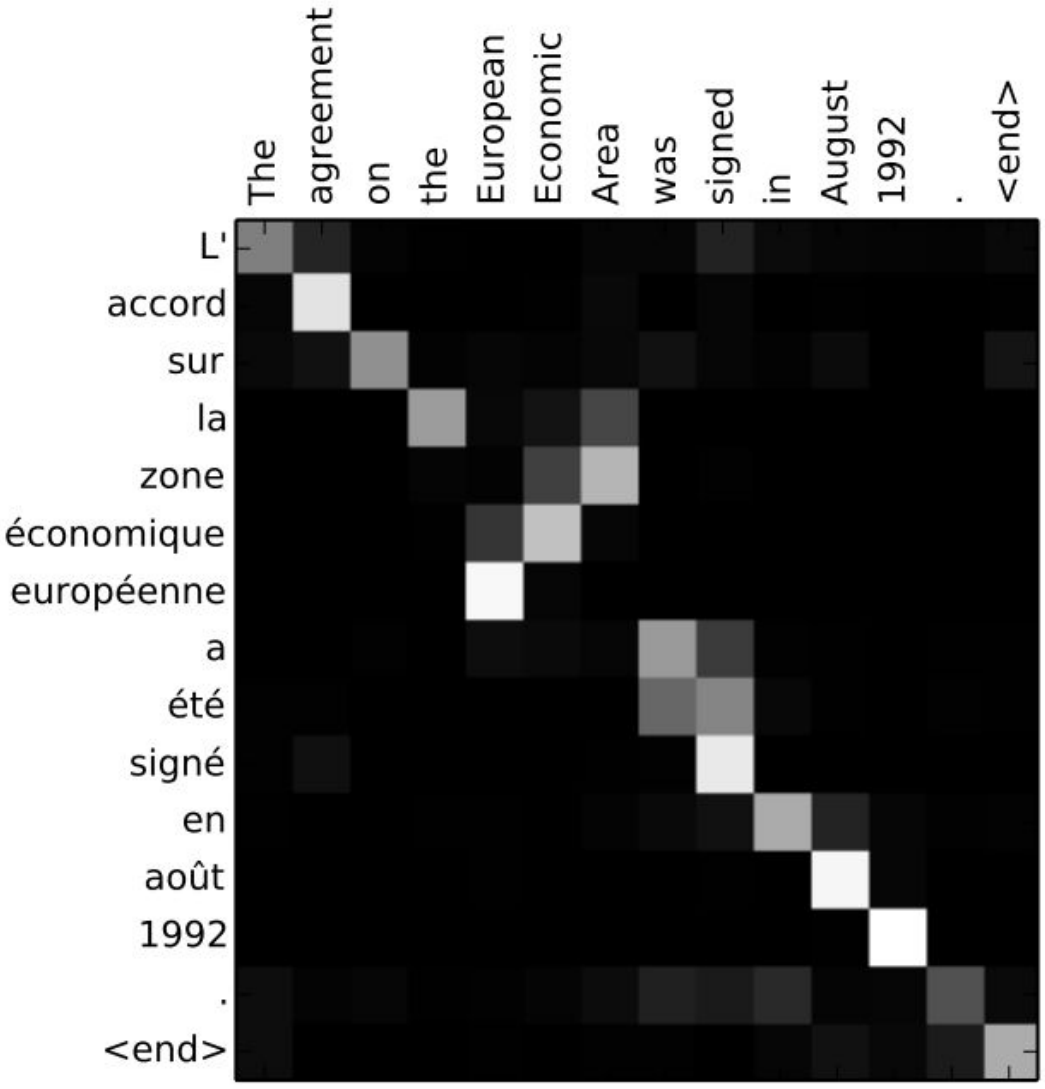
https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/

> Ce qui permet notamment la prise en compte du bon sujet.



https://research.googleblog.com/2017/08/transformer-novel-neural-network.html

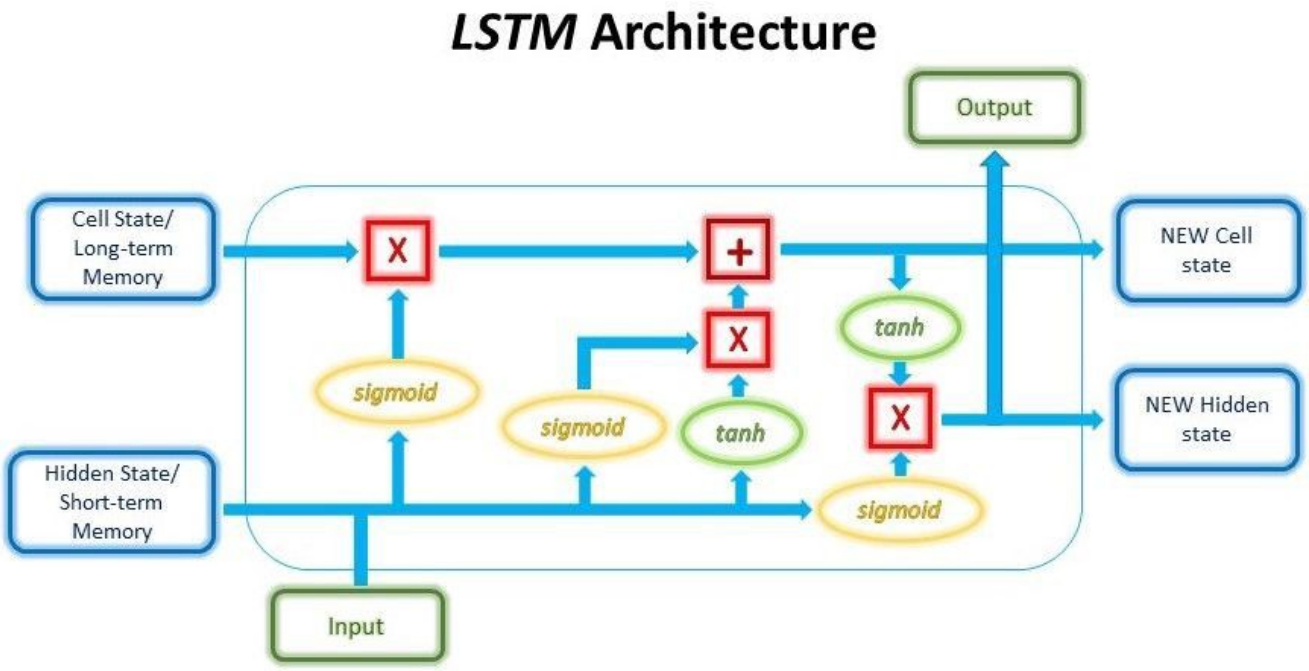
> Puis la permutation de l'ordre de certains mots pour une traduction adéquate.



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

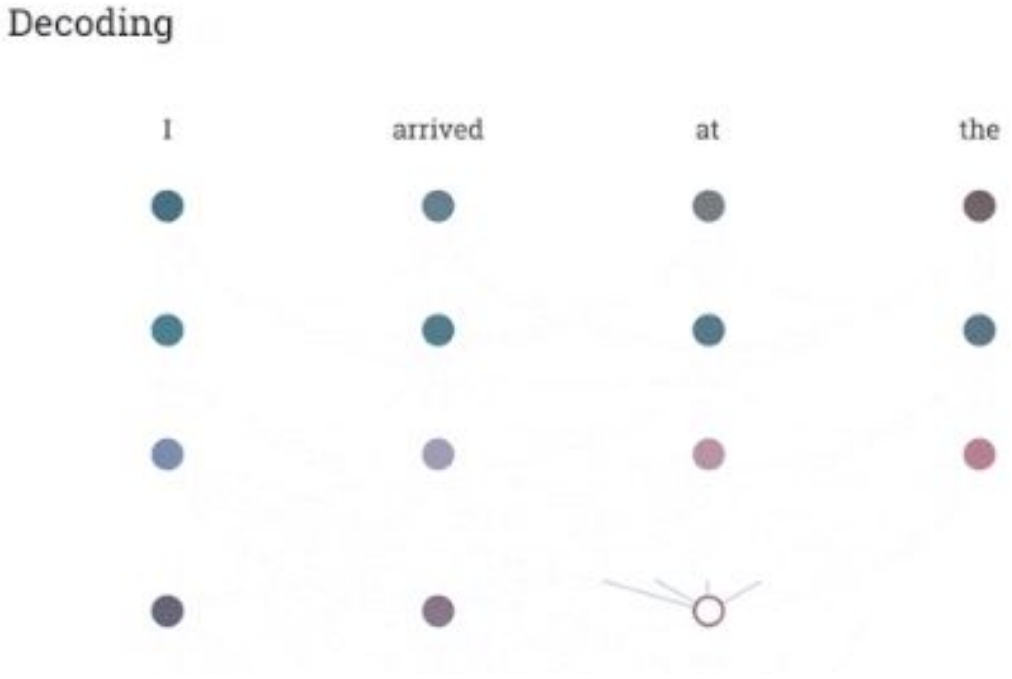
C) LTSM vs Transformers

LTSM : il s'agit de l'acronyme pour "long short term memory", équivalent bascule JK des réseaux de neurones, ayant l'inconvénient d'être séquentielle, non parallélisable, l'état mémorisé de la cellule va servir en input des suivantes, ce qui rend son usage implicitement séquentiel et fournit le socle de base du mécanisme d'attention. Permet de resetter / setter / obtenir une valeur en mémoire (comme les cellules GRU), cela rend également le réseau récurrent, de manière à repasser sur les couches d'entrées à chaque nouveau mot, ce qui ne facilite en aucun cas la parallélisation de ces derniers.



https://medium.com/analytics-vidhya/transformer-vs-rnn-and-cnn-18eeefa3602b

Attention en architecture transformers:



https://medium.com/analytics-vidhya/transformer-vs-rnn-and-cnn-18eeefa3602b

D) Transformers

- Prise en compte du sujet et des accords (coreference), ainsi que de l'ordre (via attention)
- Les attentions entre mots sont toutes associées dans le bloc multi-head attention, ce qui va permettre la parallélisation des calculs.

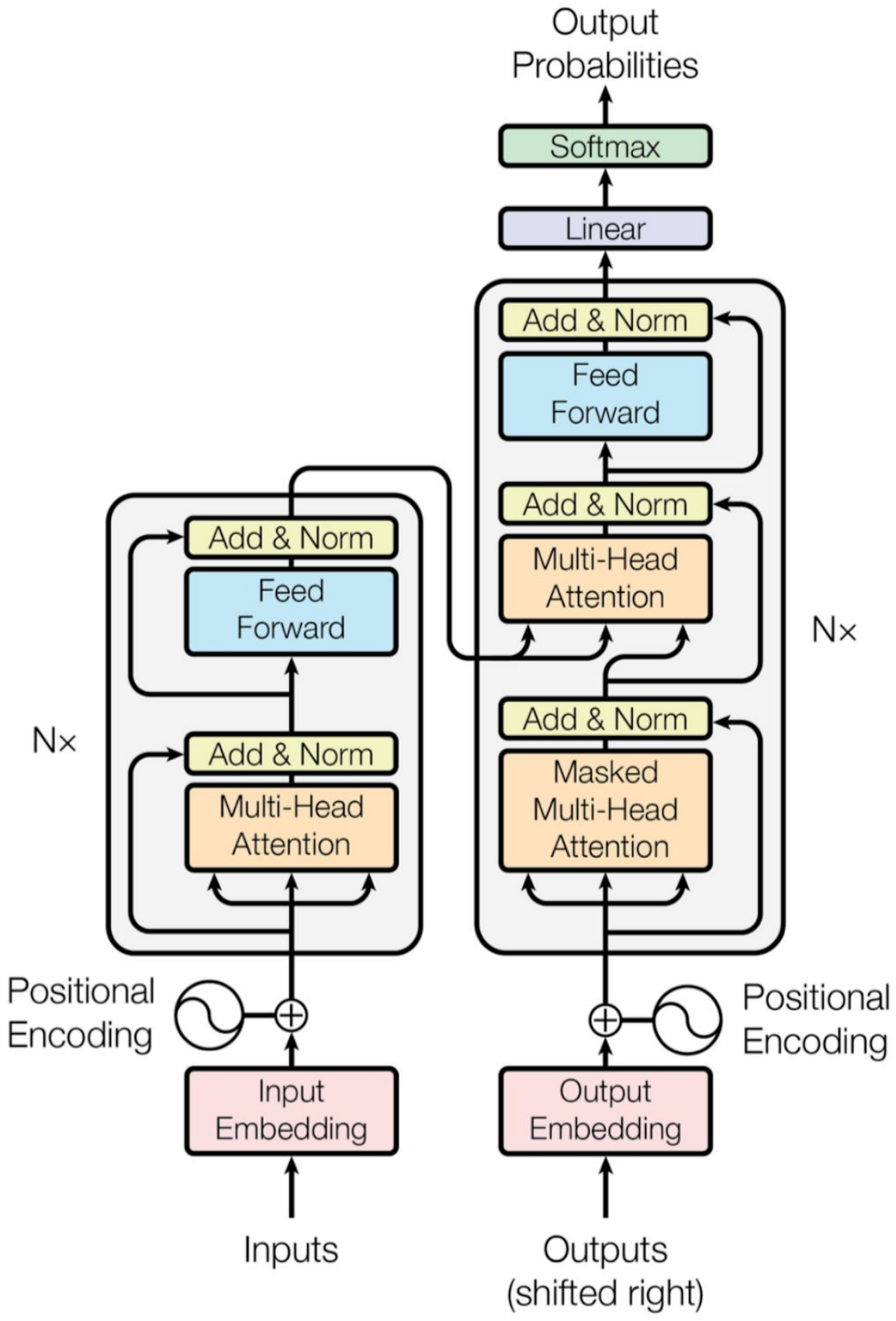


Figure 1: The Transformer - model architecture.

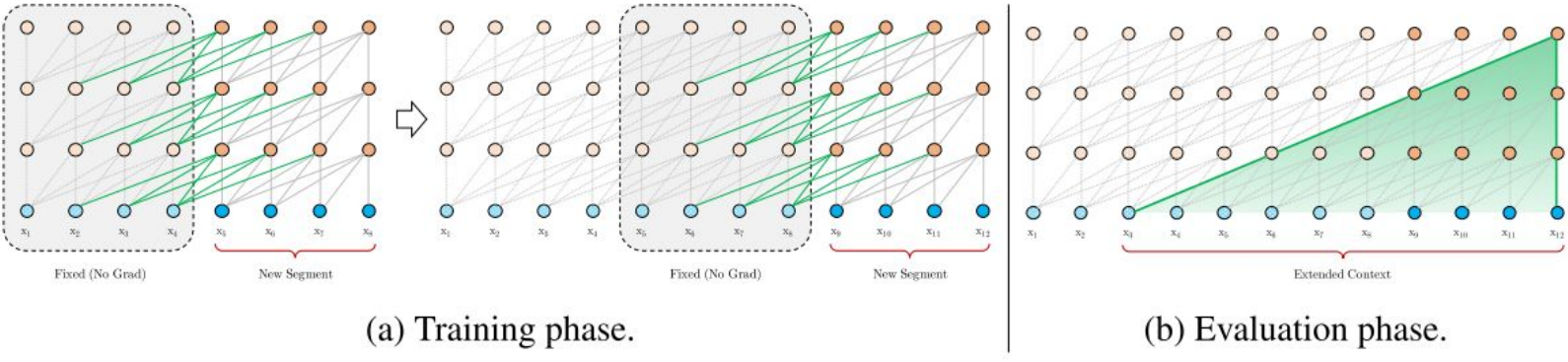
https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04

E) Bert - modèles similaires et associés

	BERT	RoBERT	DistilBERT	XLNet
Size (millions)	**Base**: 110 **Large**: 340	**Base**: 110 **Large**: 340	**Base**: 66	**Base**: ~110 **Large**: ~340
Training Time	**Base**: 8 x V100 x 12 days* **Large**: 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	**Large**: 1024 x V100 x 1 day; 4-5 times more than BERT.	**Base**: 8 x V100 x 3.5 days; 4 times less than BERT.	**Large**: 512 TPU Chips x 2.5 days; 5 times more than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	5% degradation from BERT	2-15% improvement over BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.	**Base**: 16 GB BERT data **Large**: 113 GB (16 GB BERT data + 97 GB additional). 33 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation	Bidirectional Transformer with Permutation based modeling

Ces modèles sont autorégressifs : peuvent être utilisés afin de prédire la suite d'une phrase (ou mots masqués) attention vers la suite

F) Transformer XL

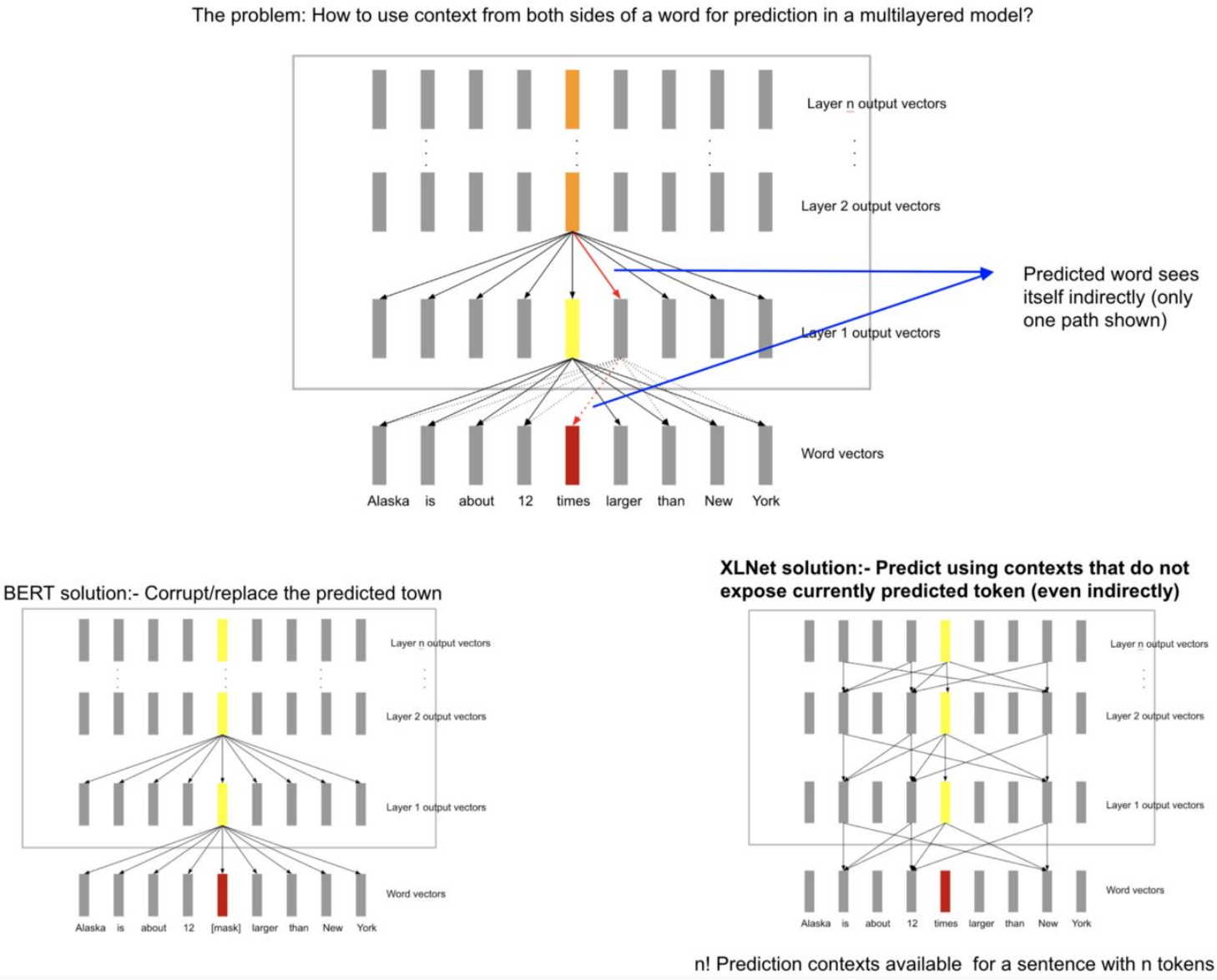


https://arxiv.org/abs/1901.02860

Est la version extra-large de transformers permettant d'étendre le contexte des mots, donc de la matrice d'attention, afin de pouvoir traiter des phrases plus longues, dont le sujet évoqué au départ, serait notamment évoqué en fin de

phrase.

G) Permutations language Modeling

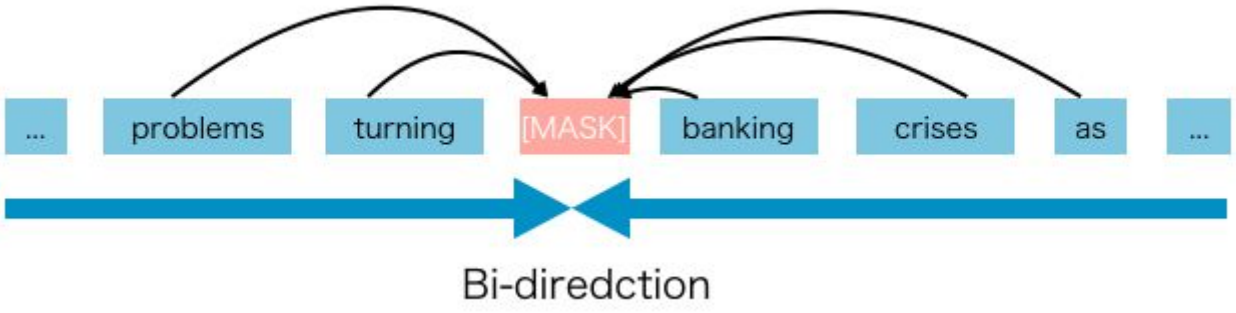


https://towardsdatascience.com/xlnet-a-clever-language-modeling-solution-ab41e87798b0

H) XLNet

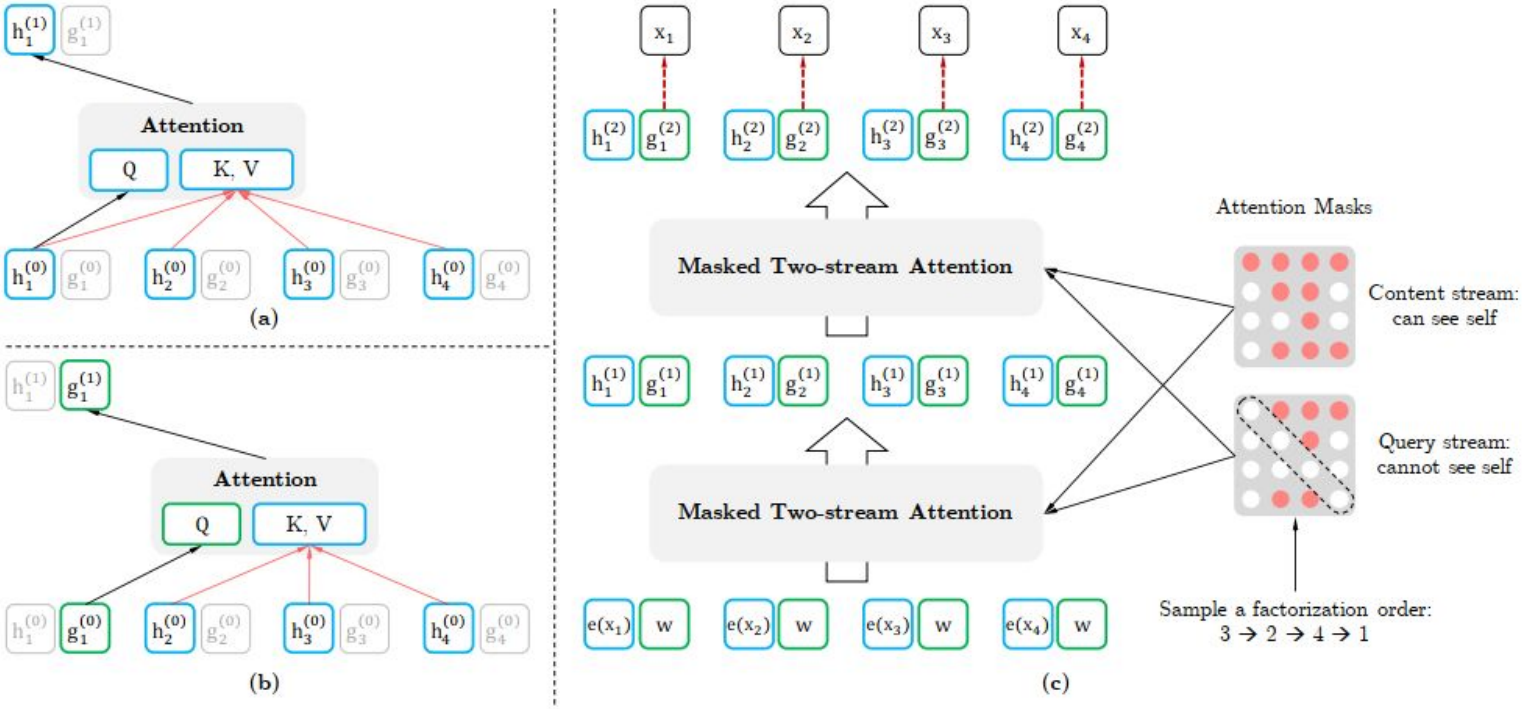
XLNet : Il s'agit d'un modèle Transformer XL Encoder-Decoder avec Attention (non RNN, donc non récurrent),

héritant de transformers xl, bidirectionnel : fonctionne dans les deux directions, peut prédire ce qui précède, attention dans les deux sens. (Masked language Modeling : 15% de mots sont masqués lors de l'entraînement afin que les matrices d'attention puissent "apprendre" les caractéristiques grammaticales d'une langue)



https://towardsdatascience.com/what-is-xlnet-and-why-it-outperforms-bert-8d8fce710335

2.3 Architecture: Two-Stream Self-Attention for Target-Aware Representations



https://arxiv.org/pdf/1906.08237.pdf

Sous le capot, deux modèles sont entraînés en parallèle, le premier voit tous les mots, le second a certains mots masqués et positions shufflées (d'où bidirectionnel), et permet la validation de l'ensemble.

La différence notoire avec Bert réside dans cette matrice d'attention bidirectionnelle qui va permettre de prédire en terme statistiques "New york is a city" au lieu de "York is a city"

Xlnet : log P(New | is a city) + log P(York | New, is a city) - given New *Mask* is a city

Bert : log P(New | is a city) + log P(York | is a city)

I) Performances comparatives

Model	IMDB	Yelp-2	Yelp-5	DBpedia	AG	Amazon-2	Amazon-5
CNN [15]	-	2.90	32.39	0.84	6.57	3.79	36.24
DPCNN [15]	-	2.64	30.58	0.88	6.87	3.32	34.81
Mixed VAT [31, 23]	4.32	-	-	0.70	4.95	-	-
ULMFiT [14]	4.6	2.16	29.98	0.80	5.01	-	-
BERT [35]	4.51	1.89	29.32	0.64	-	2.63	34.17
XLNet	**3.20**	**1.37**	**27.05**	**0.60**	**4.45**	**2.11**	**31.67**

Table 4: Comparison with state-of-the-art error rates on the test sets of several text classification datasets. All BERT and XLNet results are obtained with a 24-layer architecture with similar model sizes (aka BERT-Large).

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	WNLI
Single-task single models on dev									
BERT [2]	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-
RoBERTa [21]	90.2/90.2	94.7	92.2	**86.6**	96.4	**90.9**	68.0	92.4	-
XLNet	**90.8/90.8**	**94.9**	**92.3**	85.9	**97.0**	90.8	**69.0**	**92.5**	-
Multi-task ensembles on test (from leaderboard as of Oct 28, 2019)									
MT-DNN* [20]	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0
RoBERTa* [21]	90.8/90.2	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0
XLNet*	**90.9/90.9**†	**99.0**†	**90.4**†	**88.5**	**97.1**†	**92.9**	**70.2**	**93.0**	**92.5**

Table 5: Results on GLUE. $*$ indicates using ensembles, and † denotes single-task results in a multi-task row.

https://arxiv.org/pdf/1906.08237.pdf

Cependant de nouveaux modèles / benchmarks semblent détroner xlnet sur certaines tâches / datasets

---

7) Mise en oeuvre

Cette dernière est disponible sur le notebook en ligne à cette adresse : https://github.com/ben74/xlnet_poc/blob/master/IML_Projet_7_Poc_Stack_Overflow.ipynb

La première mise en oeuvre de Xlnet a été effectuée via pytorch & transformers (

https://towardsdatascience.com/multi-label-text-classification-with-xlnet-b5f5755302df), dont le modèle est nommé Xlnet.bin pour les comparaisons à suivre (nom de l'export du modèle entrainté)

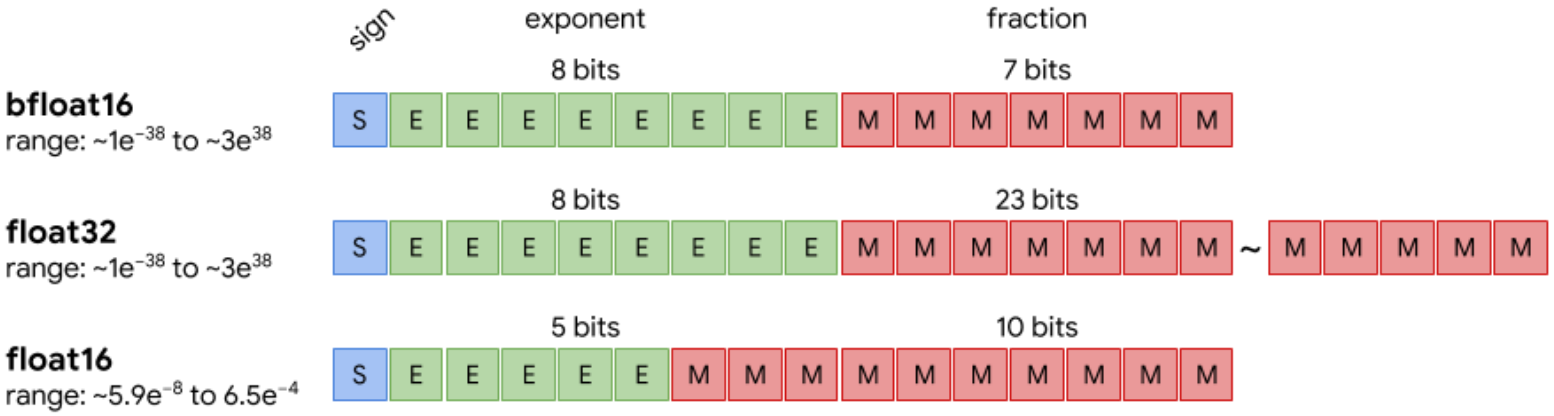
```
from transformers import AdamW, XLNetTokenizer, XLNetModel, XLNetLMHeadModel, XLNetConfig
XLNetModel.from_pretrained('xlnet-base-cased')
```

Puis une implémentation plus légère : Xlnet via simpletransformers (https://towardsdatascience.com/multi-label-classification-using-bert-roberta-xlnet-xlm-and-distilbert-with-simple-transformers-b3e0cda12ce5)

A) Floating Point 16

Il existe une option relative à l'entraînement de modèles sur un support GPU : l'utilisation de chiffre en mode floating point 16 bits, nécessitant l'installation de la librairie "nvidia APEX", réduit consécutivement l'utilisation de la mémoire en accélérant les temps de rendu, en particulier lorsque le modèle à entraîner est "très gourmand"

Ce format permet de représenter 63487 valeurs discrètes, si l'on considère la normalisation des données en entrée (via tokenization propre à xlnet, attribue des chiffres aux mots en entrée), ce qui est amplement suffisant



https://cloud.google.com/tpu/docs/bfloat16

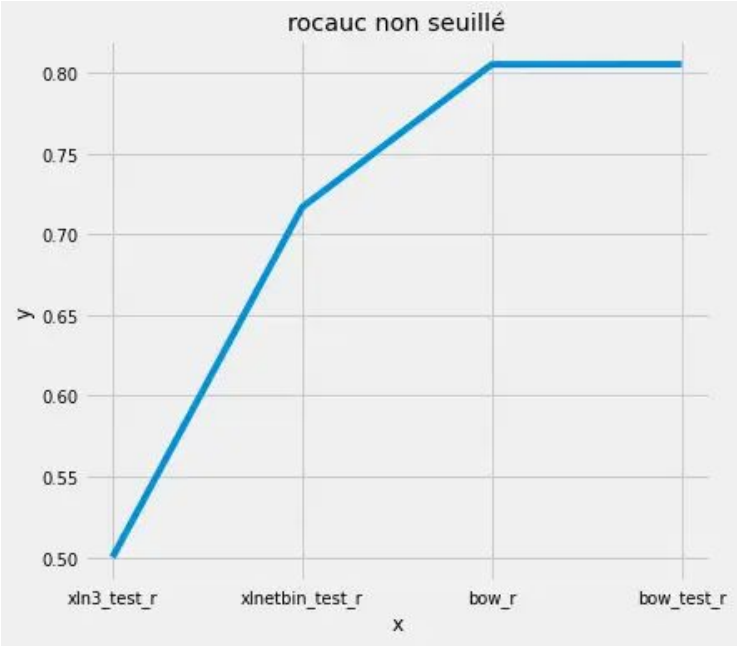
8) Conclusions

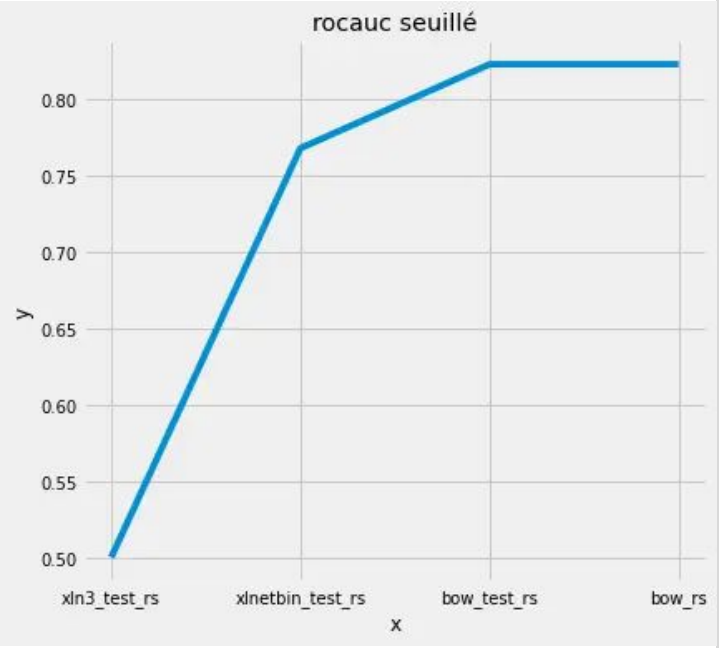
Pour une classification qui ne nécessite pas d'exploiter ni la grammaire ni les nuances d'une phrase afin d'en extraire le sentiment global positif ou négatif, l'emploi d'un mécanisme relevant de l'attention ne semble pas donner de résultats supérieurs à une méthode classique bag of words associé à une régression logistique (sous réserve de ne pas avoir déniché les bon hyperparamètres)

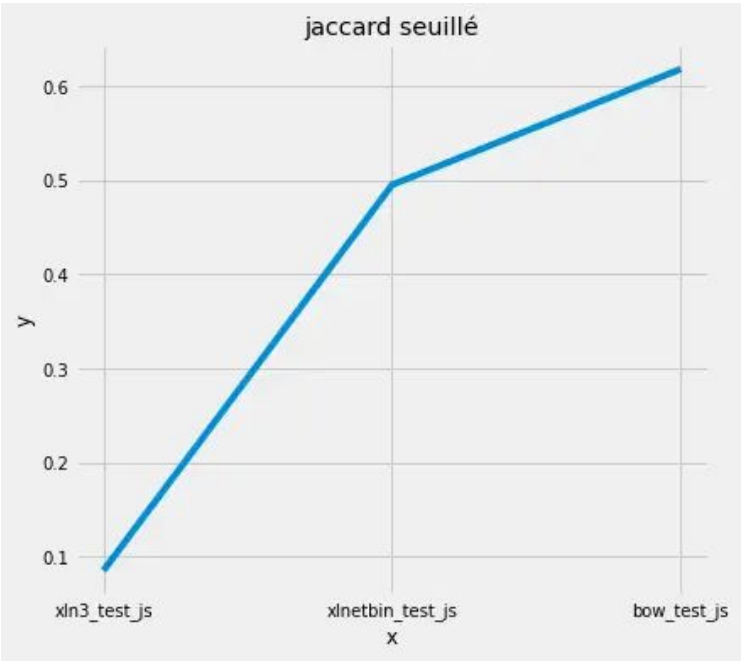
L'emploi de xlnet ne dépasse hélas pas la baseline sur ce sujet là ..

L'inverse se vérifie parfaitement sur le dataset de la compétition kaggle jigsaw (score roc auc de **0.7669** contre **0.96** pour xlnet sur le même jeu de données en prédisant une seule classe, présenté sur le prochain projet)

A) Scoring







B) Seuillage

On constate une amélioration moyenne entre 1 à 3 points de score roc auc lorsque l'on applique le seuillage.

C) Avantages

Cela m'a également permis d'explorer les modèles et leurs implémentations via plusieurs frameworks : pytorch, tensorflow, transformers et simpletransformers

Il n'existe pas de méthode ou algorithme répondant à tous les besoin, mieux que les autres, chaque méthode, algorithme peut mieux répondre à un besoin qu'un autre lorsqu'il est proprement maîtrisé

---

9) Pistes d'amélioration / approfondissement

Je me suis initialement intéressé à "Stanza", anciennement connu sous le nom de Stanford NLP, mais me suis aperçu que la librairie était orienté analyse grammaticale des phrases, mais n'ait pas trouvé de snippet de code afin de l'utiliser. Je me suis ensuite intéressé à Magnet, mais ait renoncé suite au repo github toujours vide en l'espace de 2 mois depuis sa publication initiale, puis RMDL dont mes tests d'implémentation ne retournent que des bugs listés dans le repo mais marqué comme "closed" en dépit de toute résolution.

Afin d'améliorer les résultats des prédictions il serait également possible d'effectuer un seuillage classe par classe

A la lumière de certains commentaires sur github (https://github.com/huggingface/transformers/pull/1405) il serait possible de fine-tuner le modèle xlnet afin d'obtenir de meilleur résultats

---

10) Références / Bibliographie

https://arxiv.org/pdf/1906.08237#publication originelle Xlnet

https://paperswithcode.com/task/text-classification -
https://paperswithcode.com/task/multi-label-text-classification
https://paperswithcode.com/paper/xlnet-generalized-autoregressive-pretraining
https://paperswithcode.com/paper/rmdl-random-multimodel-deep-learning-for#rmdl

https://github.com/monk1337/MAGnet#magnet
https://mccormickml.com/2019/07/22/BERT-fine-tuning/
https://colab.research.google.com/drive/1pTuQhug6Dhl9XalKB0zUGf4FIdYFlpcX#with notebook

https://towardsdatascience.com/xlnet-explained-in-simple-terms-255b9fb2c97c#xlnet simple
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/#attention weights
https://arxiv.org/abs/1706.03762#attention paper
https://www.kdnuggets.com/2019/07/xlnet-outperforms-bert-several-nlp-tasks.html
https://www.kdnuggets.com/2019/09/bert-roberta-distilbert-xlnet-one-use.html
https://towardsdatascience.com/understanding-the-difference-of-gpt-bert-and-xlnet-in-2-min-8aa917330ad1
https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04
https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129
https://medium.com/analytics-vidhya/transformer-vs-rnn-and-cnn-18eeefa3602b
https://towardsdatascience.com/what-is-xlnet-and-why-it-outperforms-bert-8d8fce710335#schémas,Gifs Attention

https://towardsdatascience.com/multi-label-classification-using-bert-roberta-xlnet-xlm-and-distilbert-with-simple-transformers-b3e0cda12ce5#xlnet simpletransformers
https://towardsdatascience.com/multi-label-text-classification-with-xlnet-b5f5755302df#xlnet pytorch
https://towardsdatascience.com/xlnet-a-clever-language-modeling-solution-ab41e87798b0
https://en.wikipedia.org/wiki/Half-precision_floating-point_format -
https://fr.wikipedia.org/wiki/IEEE_754
https://stackoverflow.com/questions/7744016/how-many-distinct-values-can-be-stored-in-floating-point-formats