TRAITEMENT AUTOMATIQUE DES LANGUES AVANCE

Master Informatique

2ème Année – 1er Semestre

Intervenants CM:

Gaël DIAS (gael.dias@unicaen.fr), Marc SPANIOL, Fabrice MAUREL

Intervenants TP:

Navneet AGARWAL, Kirill MILINTSEVICH

GREYC
Electronics and Computer Science Laboratory

Normandie Université

UNICAEN

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

cnrs

Plan de l'UE

1. **[CM 1]** Représentation sémantique de texte [GD]
2. **[CM 2]** Cohérence textuelle [MS]
3. **[CM 3]** Modélisation thématique [NA]
4. **[CM 4]** Résumé de textes et traduction automatique [MS]
5. **[CM 5]** Génération langagière I [KM]
6. **[CM 6]** Génération langagière II [KM]
7. **[CM 7]** TAL multimodal [NA]
8. **[CM 8]** TAL et web [MS]
9. **[CM 9]** TAL et handicap visuel [FM]
10. **[CM 10]** TAL et psychiatrie [GD]

11. **[TP 1-5]** Génération neuronal de comptes-rendus médicaux [NA - KM]

COURS N°1

Représentation sémantique de texte

GREYC
Electronics and Computer Science Laboratory

Normandie Université

UNICAEN

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

cnrs

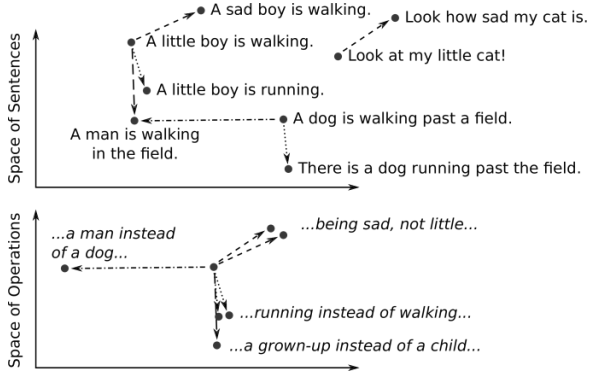
Plan du cours

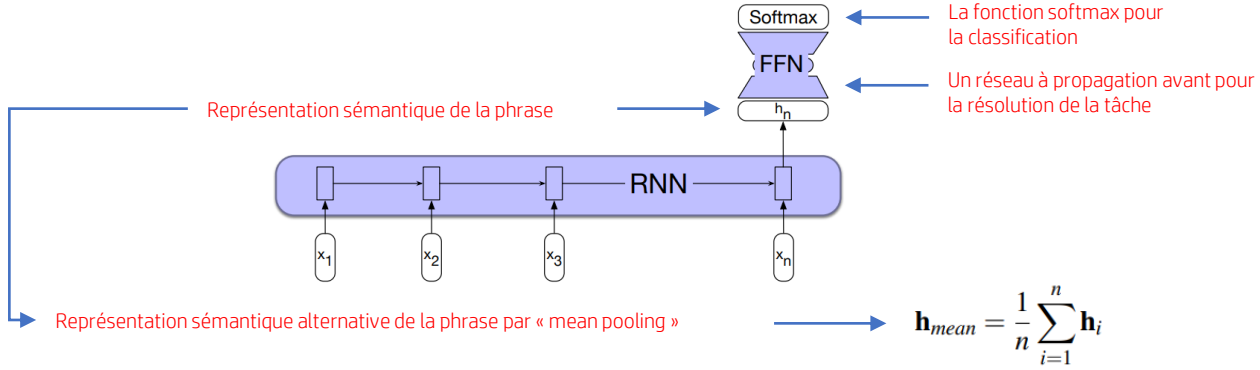
1. Représentations neuronales au niveau de la phrase
2. Les modèles du langage pré-entraînés
3. Amélioration de la représentation phrastique
4. Analyse au niveau du document
5. Les limitations des représentations neuronales
6. Les modèles hiérarchiques
7. Les transformeurs avec récurrence
8. Les transformeurs avec patrons de contenus
9. Les transformeurs avec patrons spécifiques

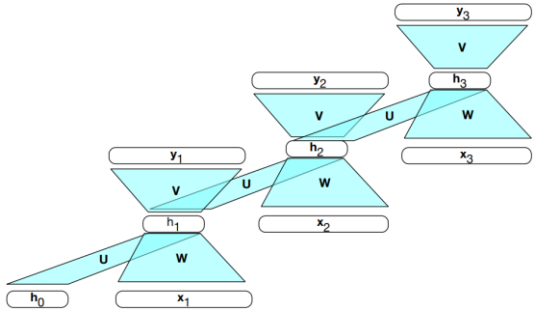
Représentations neuronales au niveau de la phrase

1. Les réseaux récurrents

- Les réseaux de neurones récurrents (RNN) comme les LSTM permettent de traiter des séquences de toute longueur.
- Ils présentent le problème de « vanishing gradient » qui empêche de capturer pleinement la sémantique des phrases.







Représentations neuronales au niveau de la phrase

2. Les transformeurs

- Contrairement aux RNN, les transformeurs ne présentent pas le problème de « vanishing gradient » et sont ainsi capables de représenter la sémantique des phrases plus complètement.
- Par contre, ils ne peuvent pas représenter des phrases de toute longueur puisqu'ils n'encodent pas la récurrence.

Représentation sémantique de la phrase

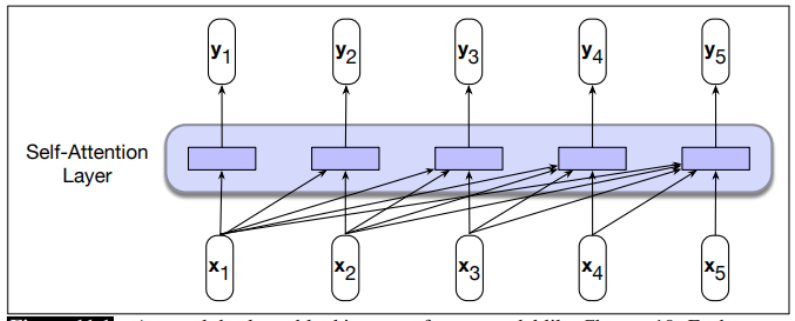
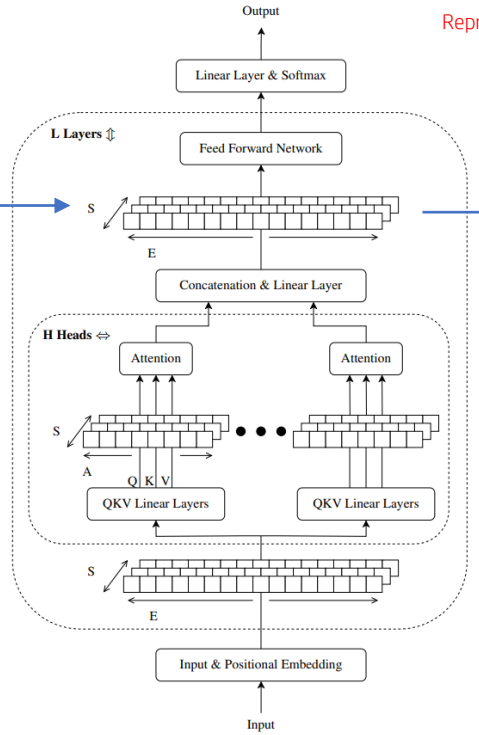


Figure 11.1 A causal, backward looking, transformer model like Chapter 10. Each output is computed independently of the others using only information seen earlier in the context.

Architecture causale : e.g. GPT



Représentation sémantique alternative de la phrase par « mean pooling »

$$\mathbf{h}_{mean} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{h}_i$$

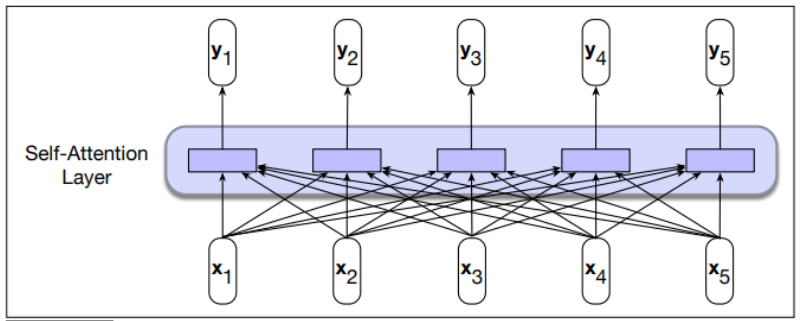


Figure 11.2 Information flow in a bidirectional self-attention model. In processing each element of the sequence, the model attends to all inputs, both before and after the current one.

Architecture bidirectionnelle : e.g. BERT

Les modèles du langage pré-entraînés

1. Idée de base

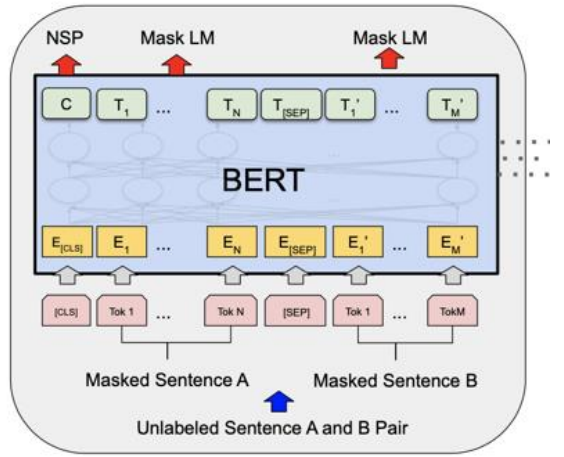
- Plutôt que de travailler avec des modèles dont les poids sont initialisés aléatoirement et de ne se fonder que sur les plongements lexicaux comme base de connaissances, il est souvent préférable de pré-calculer les poids du réseau.
- Ainsi, les modèles pré-entraînés sont des modèles où des connaissances sur la langue ont déjà été apprises de façon non supervisée (autoencodeurs).
- Ces modèles pré-entraînés sont donc des modèles du langage.
- Suivant les modèles pré-entraînés, différentes techniques peuvent être utilisées.

2. Le modèle du langage masqué (« Masked Language Modeling »)

- L'idée est de prédire des tokens d'une phrase qui sont masqués aléatoirement, appelée « cloze task ».

Please turn your homework ____ . Please turn ____ homework in.

- Cette technique est utilisée dans le cadre du modèle BERT (Delvin et al., 2019) basé sur des transformeurs bidirectionnels. En particulier, BERT est composé de 12 blocs chacun avec 12 têtes d'attention et la taille des couches cachées est de 768. Les phrases sont décomposées en tokens qui représentent des sous-mots (30000 tokens pour l'anglais).

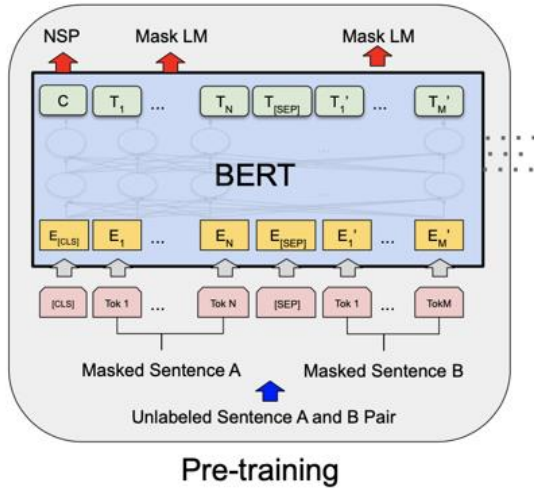


Pre-training

Les modèles du langage pré-entraînés

2. Le modèle du langage masqué (« Masked Language Modeling ») (suite)

- En plus des tokens masqués, certains mots sont remplacés par des mots non pertinents de façon aléatoire. Cela a pour objectif d'améliorer le regroupement sémantique des tokens (idée proche du « contrastive learning »).



Pre-training

3. Interprétation du modèle du langage masqué

- Etant donnée une séquence de tokens, les vecteurs de sortie de chaque token correspondent à des plongements contextualisés c'est-à-dire prenant en compte le contexte de la phrase.
- Ceci est particulièrement important pour les tokens polysémiques (e.g. jaguar) dont les représentations seront différentes selon le contexte de la phrase dans lesquels ils sont inclus.
- Ainsi, le vecteur yi du dernier bloc est le plongement du token xi.
- Il est possible de faire la moyenne des vecteurs yi des derniers blocs du transformeur pour atteindre une meilleure représentation.

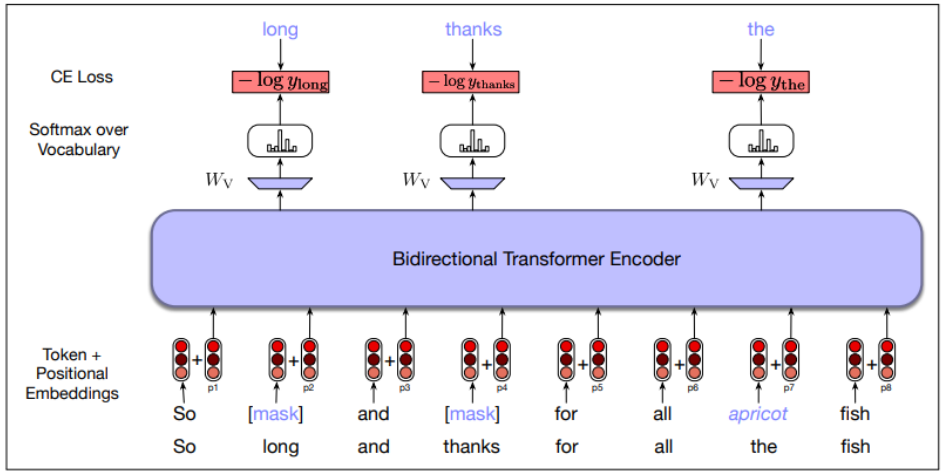


Figure 11.5 Masked language model training. In this example, three of the input tokens are selected, two of which are masked and the third is replaced with an unrelated word. The probabilities assigned by the model to these three items are used as the training loss. (In this and subsequent figures we display the input as words rather than subword tokens; the reader should keep in mind that BERT and similar models actually use subword tokens instead.)

Les modèles du langage pré-entraînés

4. La prédiction de la phrase suivante

- Dans le cadre de BERT, une autre tâche est apprise qui consiste à prédire si une phrase en précède une autre ou non.
- En effet, ceci peut aider à résoudre des tâches comme la détection de paraphrases, l'implication de phrases (« textual entailment ») ou la cohérence textuelle.
- Pour se faire, un token de segment est créé qui facilite la différenciation des deux phrases: [SEP].
- Notez que la prédiction n'est faite que sur le vecteur du token [CLS] qui correspond à la représentation de la phrase dans sa globalité.
- Chaque token est représenté par un plongement lexical, un plongement positionnel et un plongement de segment.

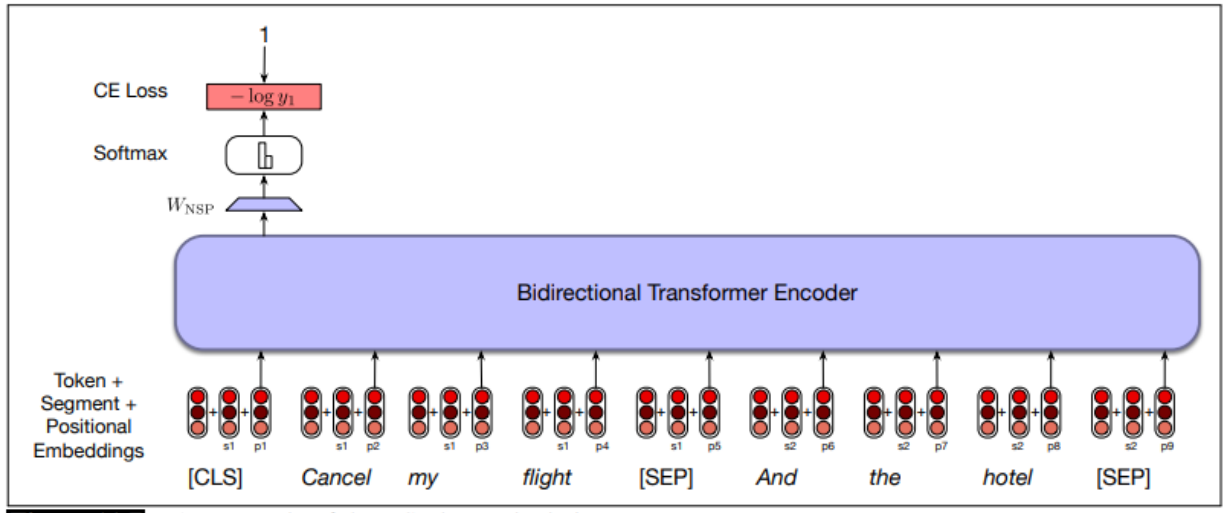
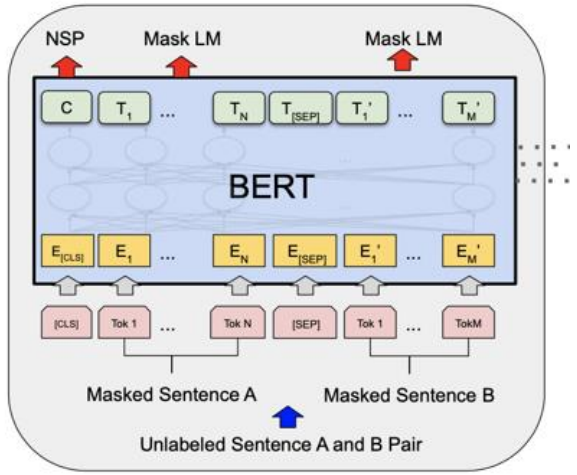


Figure 11.7 An example of the NSP loss calculation.



Pre-training

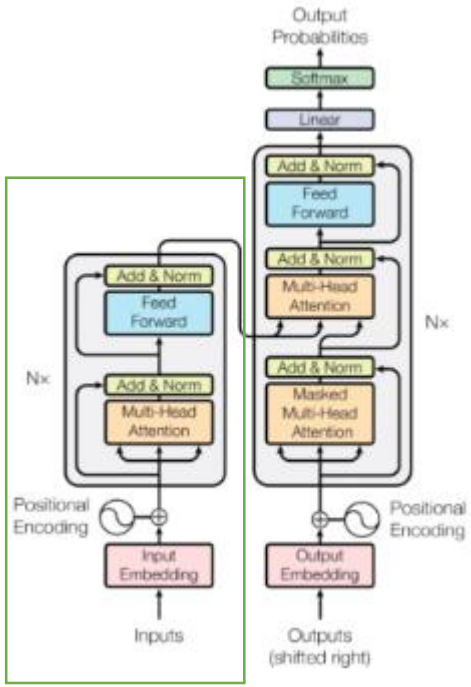
Les modèles du langage pré-entraînés

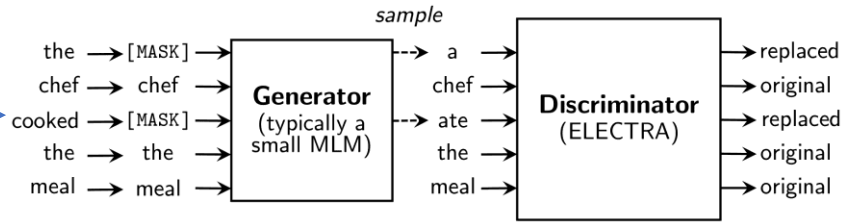
5. Types de modèles pré-entraînés

- Il existe différents types de modèles du langage pré-entraînés, selon qu'ils ont été entraînés sur des tâches de compréhension de la langue (« natural language understanding ») ou des tâches de génération (« natural language generation ») ou les deux à la fois.

6. Modèles de type Encodeur (« natural language understanding »)

- BERT (Delvin et al., 2019)
- DistillBERT (Sanh et al., 2020) – un modèle compact de BERT par distillation
- RoBERTa (Liu et al., 2019) – plus de données, pas de NSP et des phrases longues
- XLM (Lample et Conneau, 2019) – version multilingue avec une nouvelle fonction d'erreur
- ALBERT (Lan et al., 2019) – prédiction de l'ordre des phrases
- ELECTRA (Clark et al., 2020) – architecture de type *Generative Adversarial Network* (GAN)
- DeBERTa (He et al., 2021) – deux vecteurs de représentation des tokens

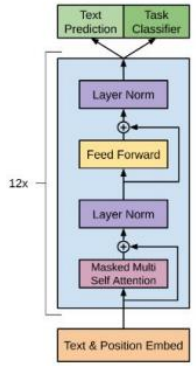


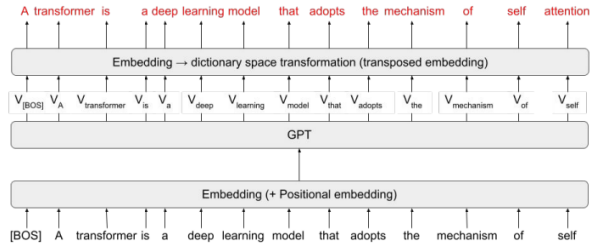


Les modèles du langage pré-entraînés

7. Modèles de type Décodeur (« natural language generation »)

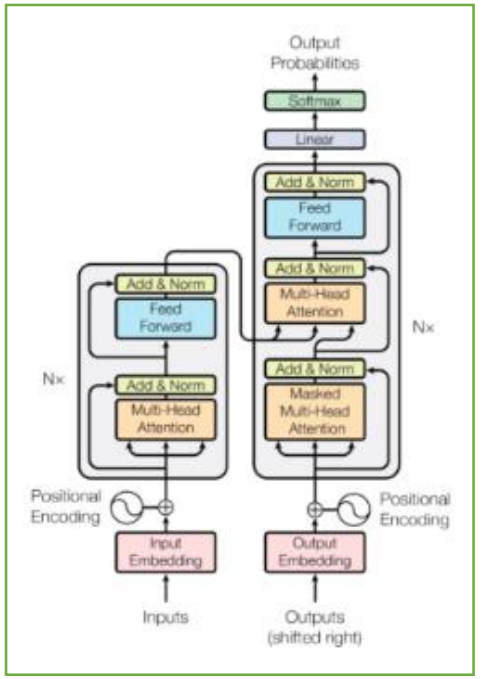
- GPT (Radford et al., 2018) – modèle causal
- CTRL (Keshar et al., 2019) – contrôle du style de génération
- GPT-2 (Solaiman et al., 2019) – peut produire des textes cohérents
- GPT-3 (Brown et al, 2020) – version avec beaucoup plus de paramètres





8. Modèles de type Encodeur - Décodeur (NLU + NLG)

- T5 (Raffel et al., 2020)
- BART (Lewis et al., 2019) – BERT + GPT
- M2M-100 (Fan et al., 2020) – modèle multilingue
- BigBIRD (Zaheer et al, 2020) – optimisé en terme de mémoire utilisée



Le « fine tuning » ou ajustement

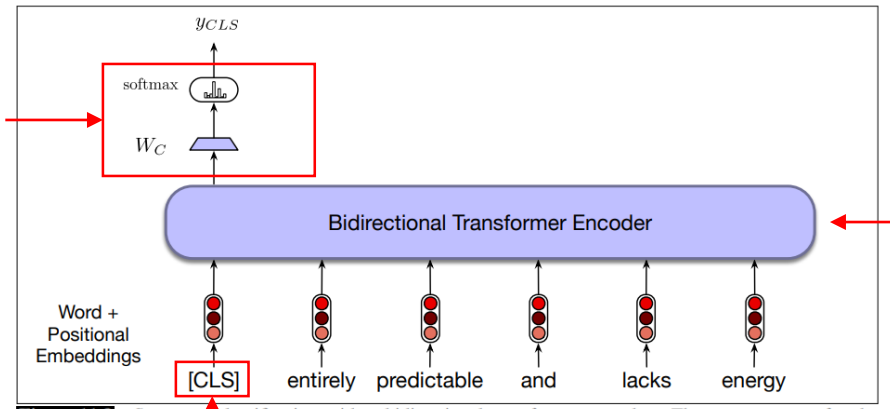
1. Idée générale

- Le « fine tuning » consiste à utiliser les modèles du langage pré-entraînés comme base de connaissance initiale pour finalement ajuster les poids du réseau pour une application donnée, e.g. analyse de sentiments, l'annotation des entités nommées etc.
- Intellectuellement, ceci revient à concevoir un réseau de neurones comme un cerveau qui contient déjà de l'information à sa naissance. Il hérite donc d'une connaissance du langage acquise au long des temps, c'est-à-dire à partir d'une quantité astronomique des textes. Ceci correspond à l'idée initiale de Chomsky mais dans un cadre statistique/probabiliste plutôt que logique.

2. Classification par « fine tuning »

- A l'entraînement, les poids du modèle pré-entraînés sont ajustés pour une petite quantité de données spécifiques à la tâche.

Couche de classification

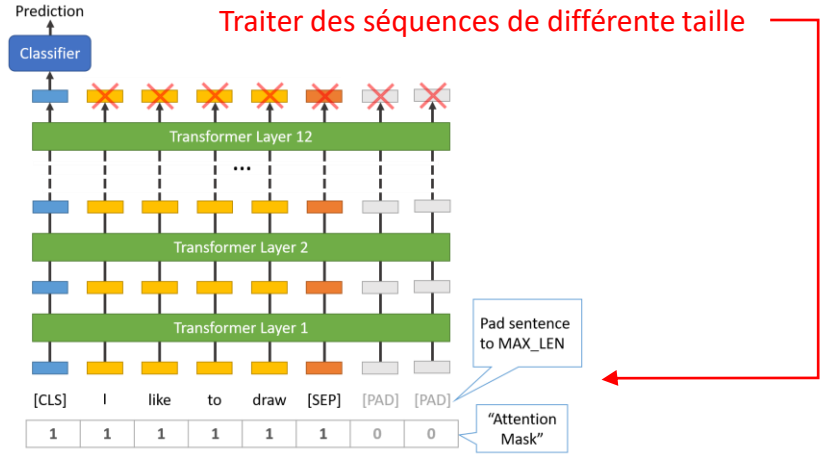


BERT

Token [CLS] sensé représenter sémantiquement la phrase

Figure 11.8 Sequence classification with a bidirectional transformer encoder. The output vector for the [CLS] token serves as input to a simple classifier.

Traiter des séquences de différente taille



Le « fine tuning » ou ajustement

3. Annotation par « fine tuning »

- Dans le cadre d'une annotation au niveau des mots ou des séquences selon le modèle BIO, le « fine tuning » est facile à modéliser avec une particularité en ce qui concerne la tokenization.

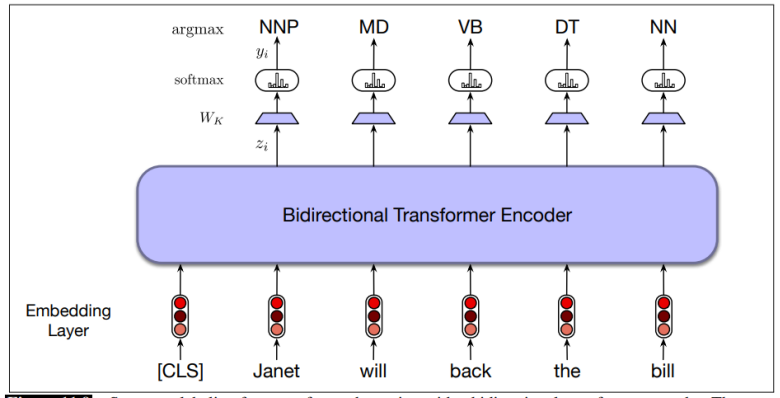


Figure 11.9 Sequence labeling for part-of-speech tagging with a bidirectional transformer encoder. The output vector for each input token is passed to a simple k-way classifier.

[LOC **Mt. Sanitas**] is in [LOC **Sunshine Canyon**]

Mt. Sanitas is in Sunshine Canyon .
B-LOC I-LOC O O B-LOC I-LOC O

BERT se repose sur une tokenization au niveau des sous-mots, donc la tâche d'annotation n'est pas directe.

```
'Mt', '.', 'San', '##itas', 'is', 'in', 'Sunshine', 'Canyon' '.'
```

A l'entraînement, chaque sous-mot reçoit l'étiquette du mot complet.

A l'inférence, le mot reçoit le label le plus significatif de son premier sous-mot. Il existe des techniques plus sophistiquées dont le but est de regarder la distribution des étiquettes de tous les sous-mots pour en déduire l'étiquette du mot.

Le « fine tuning » ou ajustement

4. Classification de séquences par « fine tuning »

- Cette tâche consiste à classer des séquences de tokens d'intérêt, comme par exemple les unités polylexicales, des unités étiquetées par des experts (e.g. mémoire épisodique, annotations de psychiatres, etc.). Cette tâche se rapproche des méthodes d'annotation BIO.

Une séquence est représentée par la concaténation de son plongement initial, de son plongement final et d'un plongement calculé par un mécanisme d'auto-attention sur l'ensemble de la séquence.

$$\text{spanRep}_{ij} = [h_i; h_j; g_{i,j}]$$

$$\mathbf{g}_{ij} = \text{SelfAttention}(\mathbf{h}_{i:j})$$

Ainsi, le plongement intermédiaire entre le plongement initial et le plongement final est une moyenne (« average pooling ») de tous les plongements de la séquence pondérés par le mécanisme d'auto-attention.

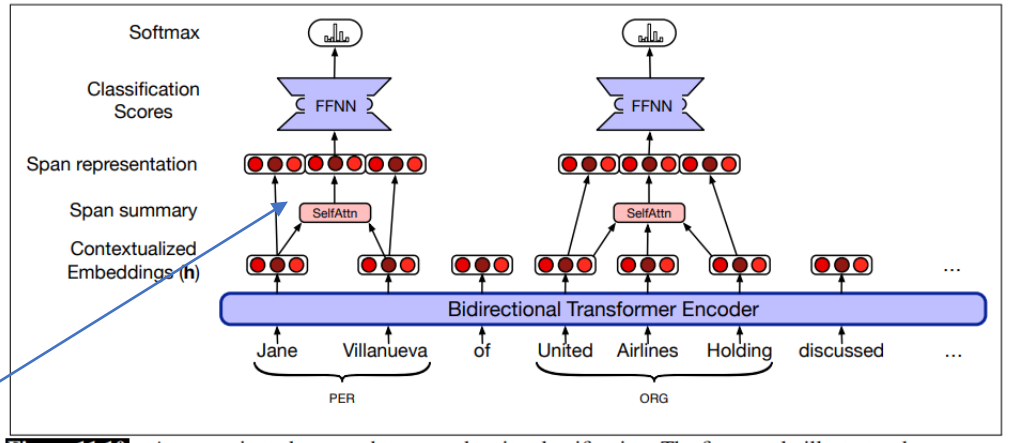
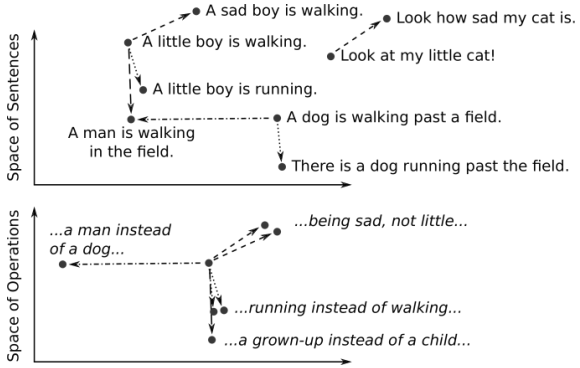


Figure 11.10 A span-oriented approach to named entity classification. The figure only illustrates the computation for 2 spans corresponding to ground truth named entities. In reality, the network scores all of the $\frac{T(T-1)}{2}$ spans in the text. That is, all the unigrams, bigrams, trigrams, etc. up to the length limit.

Toutes les sous-séquences jusqu'à une taille maximale sont classifiées

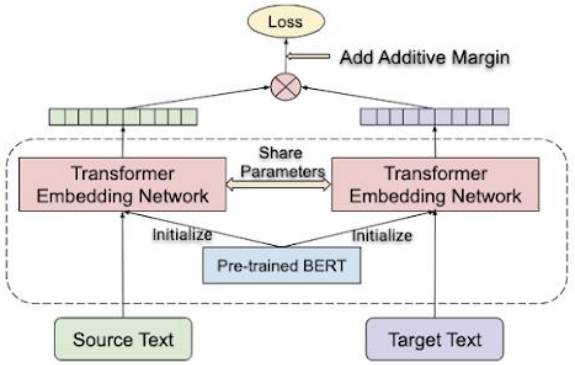
Amélioration de la représentation phrastique

1. Importance de la représentation sémantique
   - Les modèles du langage ont comme principale tâche de correctement représenter les phrases dans un espace sémantique cohérent. Plus cet espace sera juste et plus les tâches pourront être apprises simplement et plus les performances seront élevées.
2. Limites des modèles du langage
   - Les modèles du langage ont été appris sur la base de tâches bien définies. Pour BERT, le « masked language modelling » et pour GPT le « next word prediction ».
   - Or ces tâches ne peuvent garantir la construction d'un espace sémantique cohérent même si elles s'en rapprochent.
   - Des travaux s'attachent à construire des espaces de représentation sémantique, notamment basés sur l'idée de « contrastive learning » (Reimers and Gurevych 2019, Chen et al. 2020)



Please turn ____ homework in.

Please turn your homework ____ .



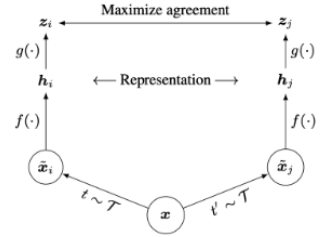


Figure 2. A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\boldsymbol{h}$ for downstream tasks.

Amélioration de la représentation phrastique

1. SentenceBERT (Reimers & Gurevych, 2019)
   - SentenceBERT propose d'ajuster un modèle du langage de type BERT en se basant sur une architecture siamoise.
   - Pour se faire, l'architecture est entraînée sur les jeux de données SNLI (https://nlp.stanford.edu/projects/snli/) et MultiNLI (https://cims.nyu.edu/~sbowman/multinli/).
   - Cette architecture est ensuite testée sans nouvel apprentissage (i.e. « linear probing ») sur la série de jeux de données STS qui évaluent les similarités entre phrases .

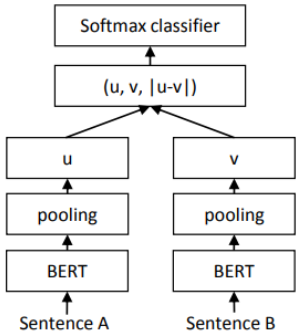


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

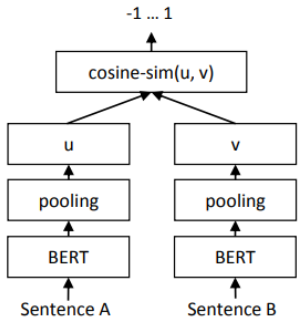


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

Model	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
Avg. GloVe embeddings	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
Avg. BERT embeddings	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT CLS-vector	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
InferSent - Glove	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
Universal Sentence Encoder	64.49	67.80	64.61	76.83	73.18	74.92	**76.69**	71.22
SBERT-NLI-base	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
SBERT-NLI-large	72.27	**78.46**	**74.90**	80.99	76.25	**79.23**	73.75	76.55
SRoBERTa-NLI-base	71.54	72.49	70.80	78.74	73.69	77.77	74.46	74.21
SRoBERTa-NLI-large	**74.53**	77.00	73.18	**81.85**	**76.82**	79.10	74.29	**76.68**

Table 1: Spearman rank correlation ρ between the cosine similarity of sentence representations and the gold labels for various Textual Similarity (STS) tasks. Performance is reported by convention as $\rho \times 100$. STS12-STS16: SemEval 2012-2016, STSb: STSbenchmark, SICK-R: SICK relatedness dataset.

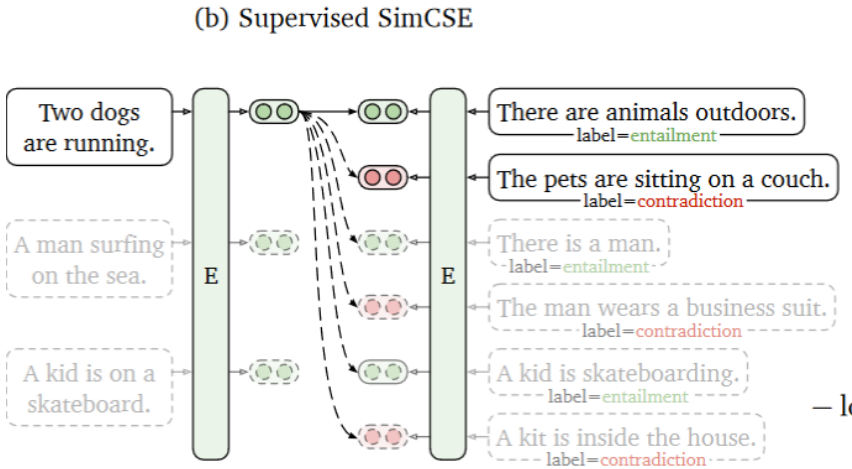
	NLI	STSb
Pooling Strategy		
MEAN	**80.78**	**87.44**
MAX	79.07	69.92
CLS	79.80	86.62
Concatenation		
(u, v)	66.04	-
$(\|u - v\|)$	69.78	-
$(u * v)$	70.54	-
$(\|u - v\|, u * v)$	78.37	-
$(u, v, u * v)$	77.44	-
$(u, v, \|u - v\|)$	**80.78**	-
$(u, v, \|u - v\|, u * v)$	80.44	-

Table 6: SBERT trained on NLI data with the classification objective function, on the STS benchmark (STSb) with the regression objective function. Configurations are evaluated on the development set of the STSb using cosine-similarity and Spearman's rank correlation. For the concatenation methods, we only report scores with MEAN pooling strategy.

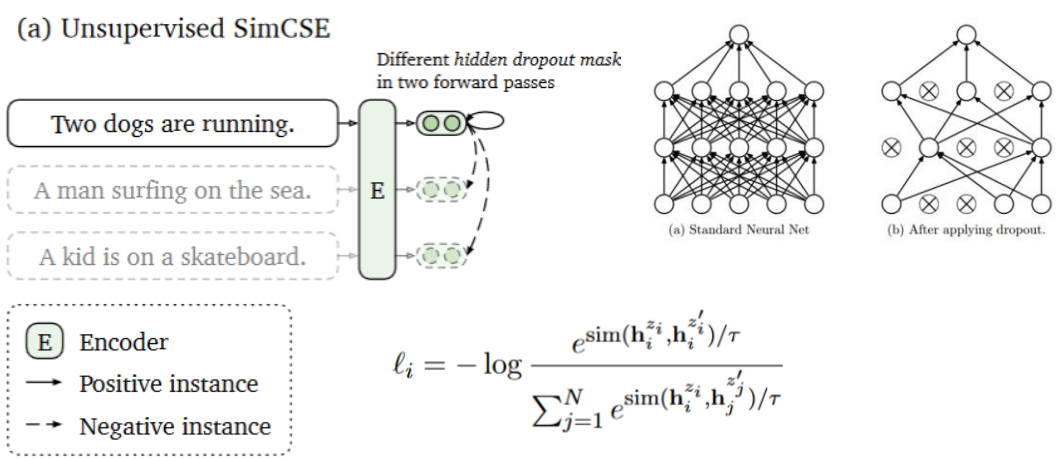
Amélioration de la représentation phrastique

1. SimSCE (Gao et al., 2022)

- SimSCE propose d'ajuster un modèle du langage de type BERT ou RoBERTa en se basant sur deux idées différentes: une architecture non supervisée et une architecture supervisée dans le cadre d'un apprentissage contrastif.
- L'architecture non supervisée reçoit une même phrase deux fois en entrée dans le même batch mais avec deux masques de « droupout » différents. L'objectif d'apprentissage est donc de rapprocher ses deux représentations.
- L'architecture supervisée est entraînée sur les jeux de données SNLI (https://nlp.stanford.edu/projects/snli/) et MultiNLI (https://cims.nyu.edu/~sbowman/multinli/) pour donner les meilleurs résultats.
- Dans l'architecture supervisée, des exemples positifs (« entailment ») et des exemples négatifs (« contradiction ») partagent le même « batch » et les exemples positifs doivent être rapprochés alors que les négatifs éloignés.



$$-\log \frac{e^{\mathrm{sim}(\mathbf{h}_i,\mathbf{h}_i^+)/\tau}}{\sum_{j=1}^{N}\left(e^{\mathrm{sim}(\mathbf{h}_i,\mathbf{h}_j^+)/\tau}+e^{\mathrm{sim}(\mathbf{h}_i,\mathbf{h}_j^-)/\tau}\right)}$$



$$\ell_i = -\log \frac{e^{\mathrm{sim}(\mathbf{h}_i^{z_i},\mathbf{h}_i^{z_i'})/\tau}}{\sum_{j=1}^{N} e^{\mathrm{sim}(\mathbf{h}_i^{z_i},\mathbf{h}_j^{z_j'})/\tau}}$$

Amélioration de la représentation phrastique

1. SimSCE (Gao et al., 20220) (suite)

Dataset	sample	full
Unsup. SimCSE (1m)	-	82.5
QQP (134k)	81.8	81.8
Flickr30k (318k)	81.5	81.4
ParaNMT (5m)	79.7	78.7
SNLI+MNLI		
entailment (314k)	**84.1**	**84.9**
neutral (314k)[8]	82.6	82.9
contradiction (314k)	77.5	77.6
all (942k)	81.7	81.9
SNLI+MNLI		
entailment + hard neg.	-	**86.2**
+ ANLI (52k)	-	85.0

Table 4: Comparisons of different supervised datasets as positive pairs. Results are Spearman's correlations on the STS-B development set using $\text{BERT}_{\text{base}}$ (we use the same hyperparameters as the final SimCSE model). Numbers in brackets denote the # of pairs. *Sample*: subsampling 134k positive pairs for a fair comparison among datasets; *full*: using the full dataset. In the last block, we use entailment pairs as positives and contradiction pairs as hard negatives (our final model).

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Unsupervised models								
GloVe embeddings (avg.)♣	55.14	70.66	59.73	68.25	63.66	58.02	53.76	61.32
$\text{BERT}_{\text{base}}$ (first-last avg.)	39.70	59.38	49.67	66.03	66.19	53.87	62.06	56.70
$\text{BERT}_{\text{base}}$-flow	58.40	67.10	60.85	75.16	71.22	68.66	64.47	66.55
$\text{BERT}_{\text{base}}$-whitening	57.83	66.90	60.90	75.08	71.31	68.24	63.73	66.28
IS-$\text{BERT}_{\text{base}}$♡	56.77	69.24	61.21	75.23	70.16	69.21	64.25	66.58
CT-$\text{BERT}_{\text{base}}$	61.63	76.80	68.47	77.50	76.48	74.31	69.19	72.05
∗ SimCSE-$\text{BERT}_{\text{base}}$	**68.40**	**82.41**	**74.38**	**80.91**	**78.56**	**76.85**	**72.23**	**76.25**
$\text{RoBERTa}_{\text{base}}$ (first-last avg.)	40.88	58.74	49.07	65.63	61.48	58.55	61.63	56.57
$\text{RoBERTa}_{\text{base}}$-whitening	46.99	63.24	57.23	71.36	68.99	61.36	62.91	61.73
DeCLUTR-$\text{RoBERTa}_{\text{base}}$	52.41	75.19	65.52	77.12	78.63	72.41	**68.62**	69.99
∗ SimCSE-$\text{RoBERTa}_{\text{base}}$	**70.16**	**81.77**	**73.24**	**81.36**	**80.65**	**80.22**	68.56	**76.57**
∗ SimCSE-$\text{RoBERTa}_{\text{large}}$	**72.86**	**83.99**	**75.62**	**84.77**	**81.80**	**81.98**	**71.26**	**78.90**
Supervised models								
InferSent-GloVe♣	52.86	66.75	62.15	72.77	66.87	68.03	65.65	65.01
Universal Sentence Encoder♣	64.49	67.80	64.61	76.83	73.18	74.92	76.69	71.22
$\text{SBERT}_{\text{base}}$♣	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
$\text{SBERT}_{\text{base}}$-flow	69.78	77.27	74.35	82.01	77.46	79.12	76.21	76.60
$\text{SBERT}_{\text{base}}$-whitening	69.65	77.57	74.66	82.27	78.39	79.52	76.91	77.00
CT-$\text{SBERT}_{\text{base}}$	74.84	83.20	78.07	83.84	77.93	81.46	76.42	79.39
∗ SimCSE-$\text{BERT}_{\text{base}}$	**75.30**	**84.67**	**80.19**	**85.40**	**80.82**	**84.25**	**80.39**	**81.57**
$\text{SRoBERTa}_{\text{base}}$♣	71.54	72.49	70.80	78.74	73.69	77.77	74.46	74.21
$\text{SRoBERTa}_{\text{base}}$-whitening	70.46	77.07	74.46	81.64	76.43	79.49	76.65	76.60
∗ SimCSE-$\text{RoBERTa}_{\text{base}}$	**76.53**	**85.21**	**80.95**	**86.03**	**82.57**	**85.83**	**80.50**	**82.52**
∗ SimCSE-$\text{RoBERTa}_{\text{large}}$	**77.46**	**87.27**	**82.36**	**86.66**	**83.93**	**86.70**	**81.95**	**83.76**

Table 5: Sentence embedding performance on STS tasks (Spearman's correlation, "all" setting). We highlight the highest numbers among models with the same pre-trained encoder. ♣: results from Reimers and Gurevych (2019); ♡: results from Zhang et al. (2020); all other results are reproduced or reevaluated by ourselves. For BERT-flow (Li et al., 2020) and whitening (Su et al., 2021), we only report the "NLI" setting (see Table C.1).

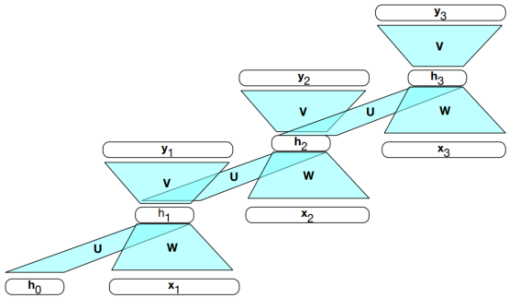
Analyse au niveau du document

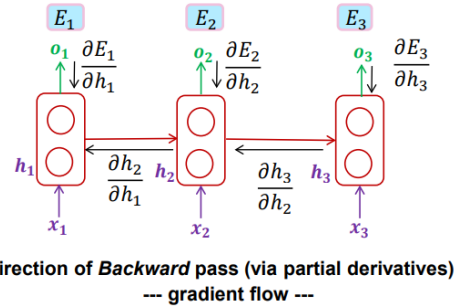
1. Différentes applications
   - Au semestre 1 du Master 1, nous avons étudié le langage au niveau du mot (analyse lexicale).
   - Au semestre 2 du Master 1, nous avons étudié le langage au niveau de la phrase (analyse phrastique)
   - Or, de nombreuses applications du langage naturel traitent des textes qui sont des séquences longues de phrases ou de paragraphes.
2. Applications intra-document
   - Cohérence textuelle : analyse des coréférences
   - Structure du discours : chaînes lexicales, segmentation thématique
   - Structure argumentative : « argument mining »
3. Applications de classification
   - Analyse de sentiments, d'émotions, de fake news, de diagnostic médical, de profilage etc …
4. Applications de génération
   - Traduction automatique
   - Résumé de textes
   - Dialogues

Les limitations des représentations neuronales

1. Représentations des textes par RNN

- Les RNN ont une capacité illimitée de représentation du fait de leur récurrence. Tout texte peut être représenté par un RNN. Par contre, leur capacité à représenter les séquences longues est limitée par la notion de « vanishing gradient ».





Direction of *Backward* pass (via partial derivatives)
--- gradient flow ---

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W} + \frac{\partial E_3}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W} + \frac{\partial E_3}{\partial h_3}\frac{\partial h_3}{\partial h_3}\frac{\partial h_3}{\partial W}$$

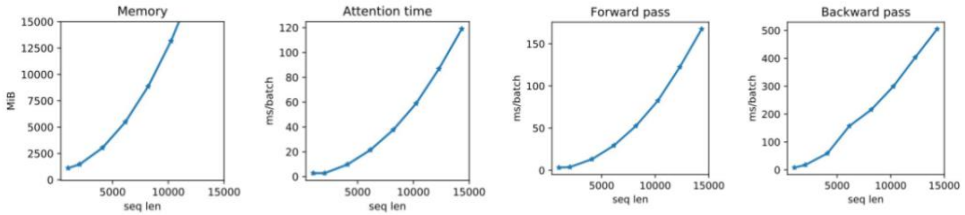
$$= \lll 1 \quad + \quad \ll 1 \quad + \quad < 1$$

Les gradients des dépendances longues sont dominés par les dépendances courtes.

2. Représentation des textes par transformeurs

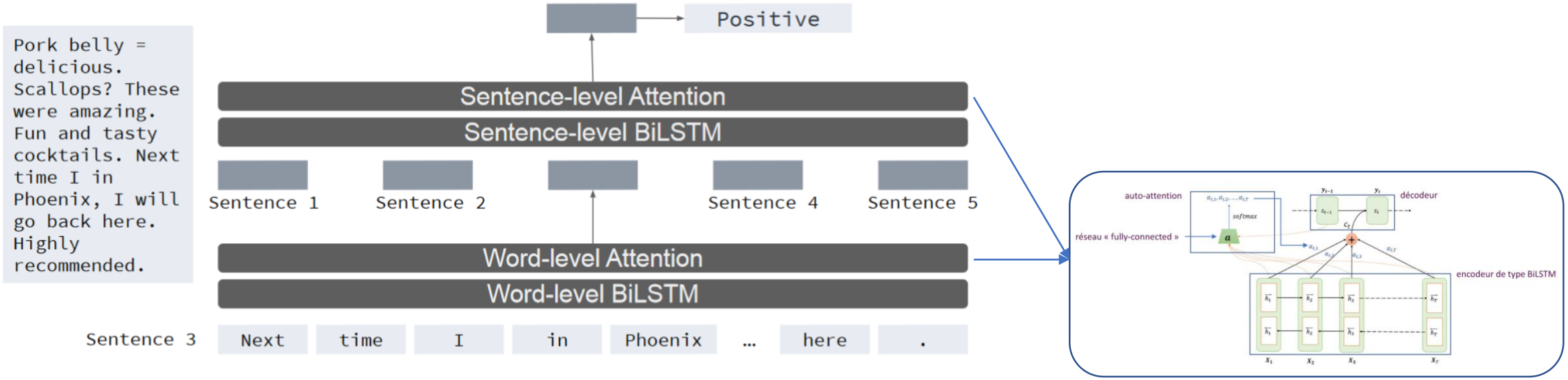
- Les transformeurs ont une capacité limitée de représentation du fait de leur non récurrence. Par exemple, pour BERT le nombre de tokens en entrée est limité à 512.
- Cette limitation est aussi due à la complexité quadratique en coût de calcul: $O(N^2)$ où N est la taille de la séquence.

Pour un transformeur à une tête et un sel bloc sur une carte graphique RTX8000 GPU.



Les modèles hiérarchiques

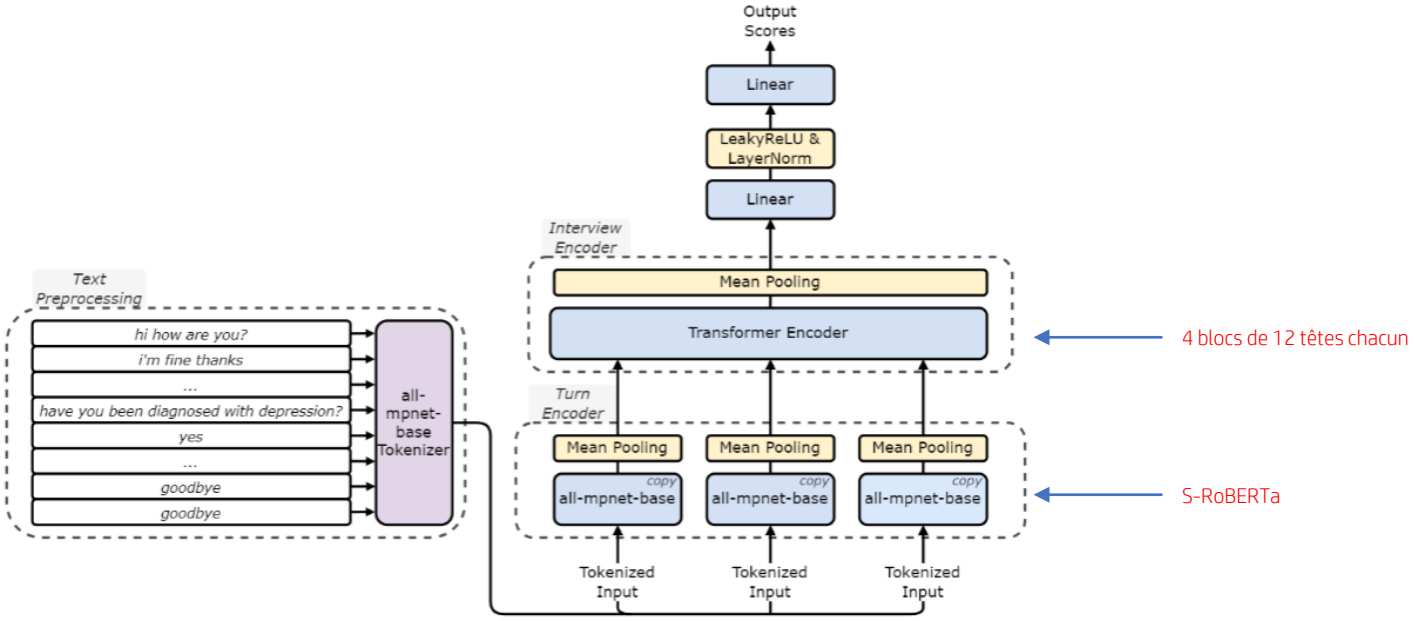
1. Utiliser la nature hiérarchique des textes
   - Les textes sont composés hiérarchiquement de caractères, de mots, de phrases et de paragraphes.
   - Les modèles de représentation prennent en compte cette organisation pour modéliser de longs textes.
2. Une représentation à partir de BiLSTM (Yang et al., 2016)
   - Le texte est d'abord divisé en phrases qui sont elles-mêmes divisées en mots. Chaque phrase est représentée par la dernière couche du BiLSTM initialisé à partir de plongements lexicaux. Toutes les phrases sont ensuite agglomérées dans un BiLSTM dont la dernière couche représente le texte dans son entièreté.



Les modèles hiérarchiques

3. Une représentation à partir de modèles pré-entraînés (Milintsevich et al., 2023)

- Afin de prendre en compte les modèles pré-entraînés de type BERT comme représentation des phrases et les transformeurs comme agrégateurs de sémantique, des architectures hiérarchiques plus performantes peuvent être élaborées.
- Dans ce cas, les phrases sont représentées grâce à des modèles de type SentenceBERT, ici S-RoBERTa qui sont ensuite agglomérées grâce à un transformeur.



Les modèles hiérarchiques

4. Des modèles hiérarchiques pré-entraînés au niveau du document

- (Zhang et al. 2020) proposent de pré-entraîner un modèle hiérarchique à partir de la notion de « masked language modeling » mais au niveau de la phrase.
- L'idée est de construire un modèle capable de représenter sémantiquement un texte de manière auto-supervisée, c'est-à-dire en reconstruisant des phrases masquées et en prédisant la phrase suivante.
- Le modèle HiBERT a été entraîné à partir du corpus GIGA-CM comportant 6.3 millions de documents et 2.8 milliards de tokens.

Pré-entraîné sur GIGA-CM

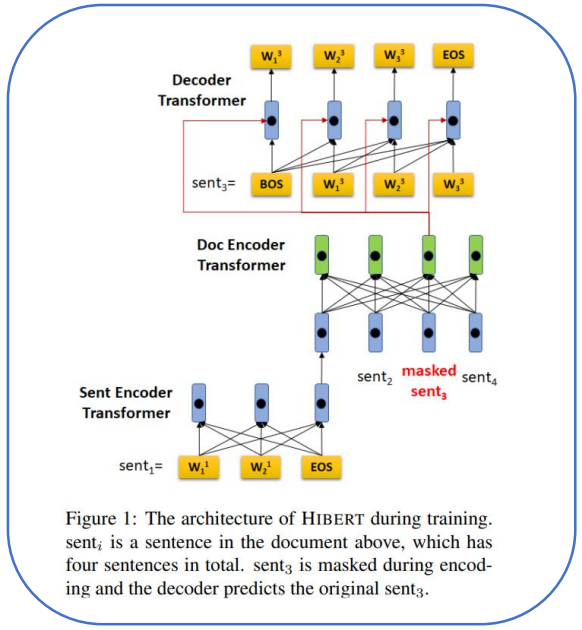


Figure 1: The architecture of HIBERT during training. $sent_i$ is a sentence in the document above, which has four sentences in total. $sent_3$ is masked during encoding and the decoder predicts the original $sent_3$.

Model	R-1	R-2	R-L
Pointer+Coverage	39.53	17.28	36.38
Abstract-ML+RL	39.87	15.82	36.90
DCA	41.69	19.47	37.92
SentRewrite	40.88	17.80	38.54
InconsisLoss	40.68	17.97	37.13
Bottom-Up	41.22	18.68	38.34
Lead3	40.34	17.70	36.57
SummaRuNNer	39.60	16.20	35.30
NeuSum	40.11	17.52	36.39
Refresh	40.00	18.20	36.60
NeuSum-MMR	41.59	19.01	37.98
BanditSum	41.50	18.70	37.60
JECS	41.70	18.50	37.90
LatentSum	41.05	18.77	37.54
HierTransformer	41.11	18.69	37.53
BERT	41.82	19.48	38.30
HIBERT$_S$ (in-domain)	42.10	19.70	38.53
HIBERT$_S$	**42.31**	**19.87**	**38.78**
HIBERT$_M$	**42.37**	**19.95**	**38.83**

Table 1: Results of various models on the CNNDM test set using full-length F1 ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L).

Les transformeurs avec récurrence

1. Introduire de la récurrence dans les transformeurs

- (Dai et al. 2019) proposent de sérialiser le traitement des séquences en gardant en mémoire les valeurs d'activation (valeurs d'attention et couches cachées) du segment précédent.
- Ce modèle appelé Transformer-XL introduit aussi la notion de plongement de position relative.

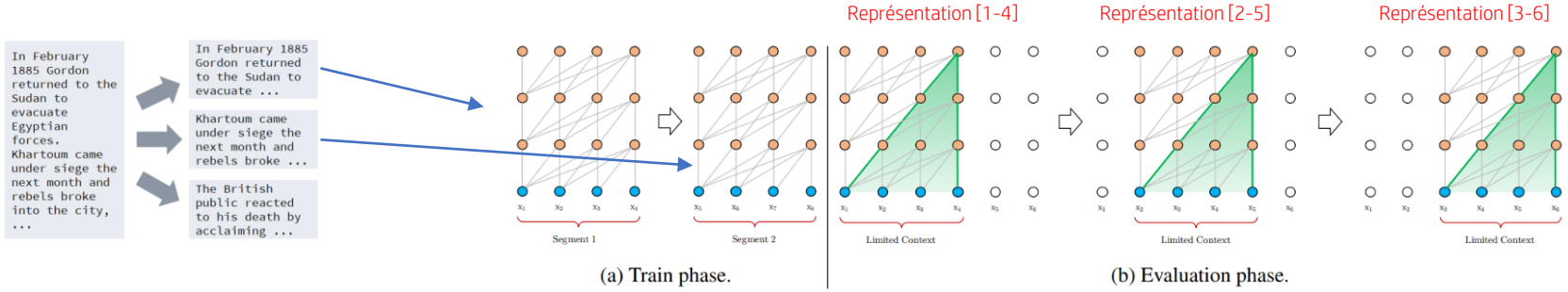


Figure 1: Illustration of the vanilla model with a segment length 4.

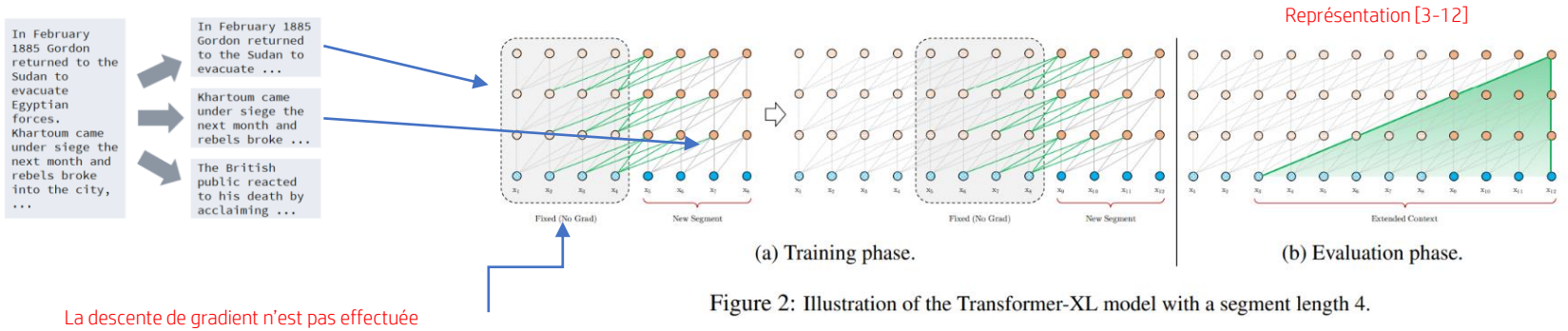


Figure 2: Illustration of the Transformer-XL model with a segment length 4.

La descente de gradient n'est pas effectuée

Model	#Param	PPL
Grave et al. (2016b) - LSTM	-	48.7
Bai et al. (2018) - TCN	-	45.2
Dauphin et al. (2016) - GCNN-8	-	44.9
Grave et al. (2016b) - LSTM + Neural cache	-	40.8
Dauphin et al. (2016) - GCNN-14	-	37.2
Merity et al. (2018) - QRNN	151M	33.0
Rae et al. (2018) - Hebbian + Cache	-	29.9
Ours - Transformer-XL Standard	151M	**24.0**
Baevski and Auli (2018) - Adaptive Input◇	247M	20.5
Ours - Transformer-XL Large	257M	**18.3**

Table 1: Comparison with state-of-the-art results on WikiText-103. ◇ indicates contemporary work.

Performance pour la tâche de « language modeling » (prédire le mot suivant) sur le jeu de données standard WikiText-103.

Les transformeurs avec récurrence

1. Introduire de la récurrence plus lointaine dans les transformeurs

- (Rae et al. 2019) proposent d'étendre la méthodologie de (Dai et al., 2019) afin de prendre en compte un historique plus lointain et ainsi de permettre une récurrence plus longue.
- Pour se faire, deux types de mémoire sont utilisés: une mémoire de court terme et une mémoire compressée.
- Ce modèle appelé « Compressive Transformer » introduit plusieurs fonctions de compression dont l'objectif est de minimiser la différence entre les attentions compressées et les attentions réelles de la mémoire. Cet objectif est optimisé dans un second temps car il est contre-productif avec l'apprentissage du modèle de langage.

Séquence précédente de la précédente comprimée mais accessible pour le calcul de la séquence actuelle

Séquence précédente comme dans (Dai et al., 2019)

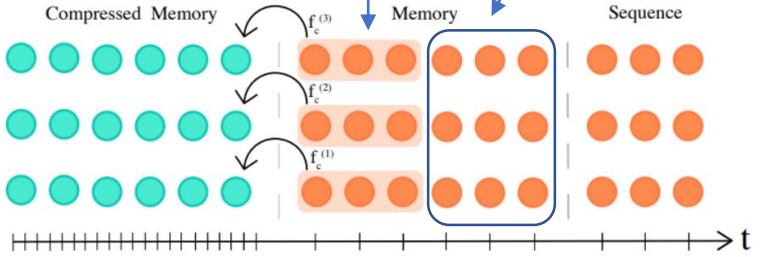


Figure 1: The Compressive Transformer keeps a fine-grained memory of past activations, which are then compressed into coarser *compressed* memories. The above model has three layers, a sequence length $n_s = 3$, memory size $n_m = 6$, compressed memory size $n_{cm} = 6$. The highlighted memories are compacted, with a compression function f_c per layer, to a single compressed memory — instead of being discarded at the next sequence. In this example, the rate of compression $c = 3$.

Table 6: Validation and test perplexities on WikiText-103.

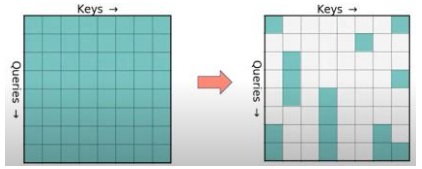
	Valid.	Test
LSTM (Graves et al., 2014)	-	48.7
Temporal CNN (Bai et al., 2018a)	-	45.2
GCNN-14 (Dauphin et al., 2016)	-	37.2
Quasi-RNN Bradbury et al. (2016)	32	33
RMC (Santoro et al., 2018)	30.8	31.9
LSTM+Hebb. (Rae et al., 2018)	29.0	29.2
Transformer (Baevski and Auli, 2019)	-	18.7
18L TransformerXL, M=384 (Dai et al., 2019)	-	18.3
18L TransformerXL, M=1024 (ours)	-	18.1
18L Compressive Transformer, M=1024	**16.0**	**17.1**

Table 7: WikiText-103 test perplexity broken down by word frequency buckets. The most frequent bucket is words which appear in the training set more than $10,000$ times, displayed on the left. For reference, a uniform model would have perplexity $|V| = 2.6e5$ for all frequency buckets. *LSTM comparison from Rae et al. (2018)

	> 10K	1K$-$10K	$100-1$K	< 100	All
LSTM*	12.1	219	1,197	9,725	36.4
TransformerXL (ours)	7.8	61.2	188	1,123	18.1
Compressive Transformer	**7.6**	**55.9**	**158**	**937**	**17.1**
Relative gain over TXL	2.6%	9.5%	21%	19.9%	5.8%

Les transformeurs avec patrons de contenus

1. Minimiser le spectre d'attention
   - Plutôt que d'introduire de la récurrence, d'autres approches s'appliquent à réduire le spectre des attentions afin de diminuer la complexité de calcul et ainsi permettre d'augmenter la taille des textes en entrée.

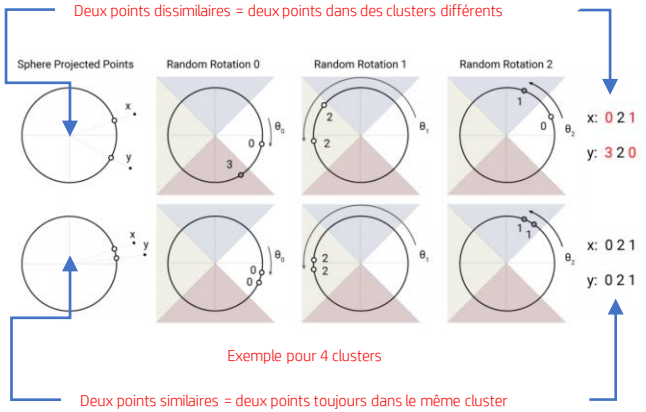


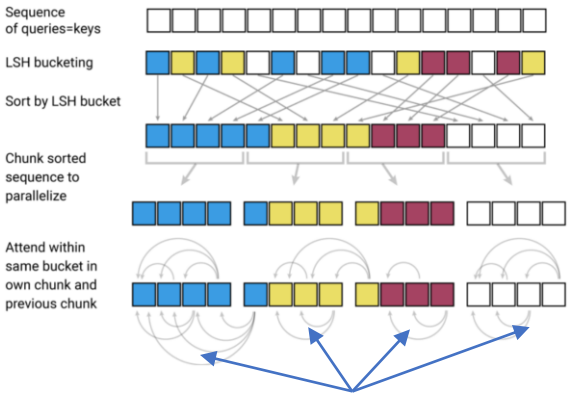
Spectre complet Spectre réduit

2. Minimiser le spectre d'attention par contenus
   - Les patrons de contenus ont pour objectif de comprendre les régularités dans les données

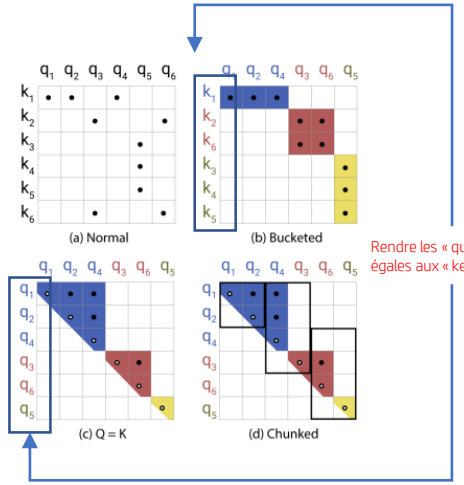
en entrée et de ne focaliser le calcul de l'attention que sur les données en entrée similaires.

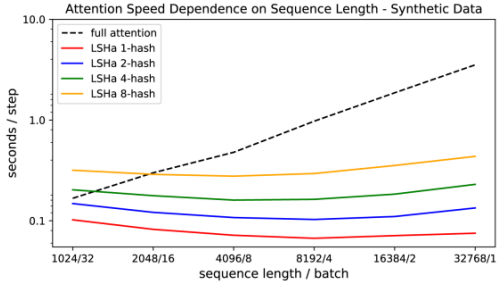
3. Le modèle Reformer
   - (Kitaev et al., 2020) se base sur la méthode de clustering LSH (« Locality Sensitive Hashing ») pour regrouper toutes les requêtes (« queries ») similaires et focaliser l'attention sur ces clusters.





L'attention de ne calcule qu'à l'intérieur d'un même cluster

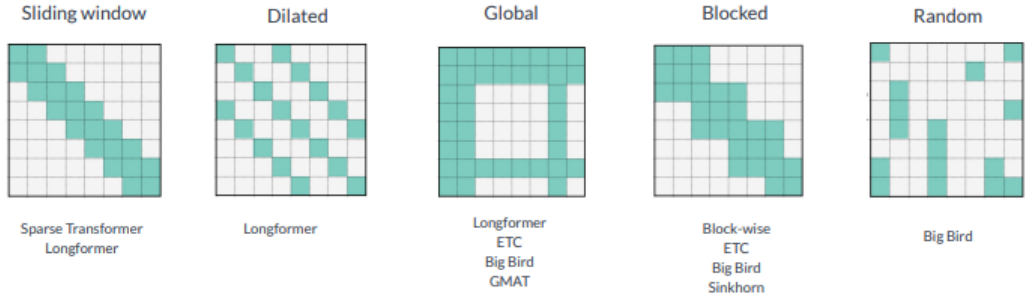




Les transformeurs avec patrons spécifiques

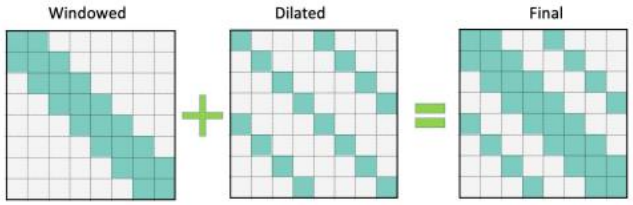
1. Minimiser le spectre d'attention par patrons

- Plutôt que d'apprendre des régularités sur les données, ce qui rajoute une étape supplémentaire, des patrons d'attention spécifiques peuvent être proposés. Ce sont des architectures basées modèle.

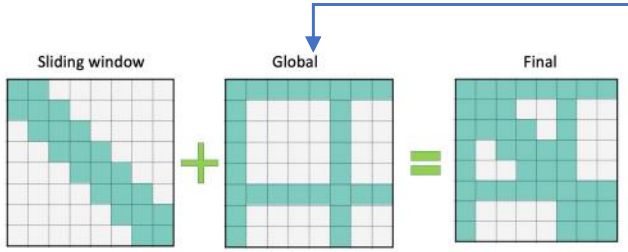


2. Le modèle LongFormer

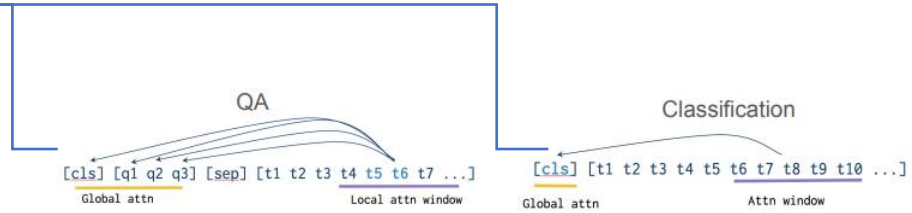
- (Beltagy et al., 2020) propose plusieurs modèles suivant la tâche à apprendre.
- Aucune information globale n'est utilisée pour la modélisation du langage (« language modeling »).
- L'information globale est utilisée pour l'ajustement à certaines tâches (« downstream tasks »).
- Différents blocs et têtes d'attention peuvent avoir des modèles différents.



Language modeling



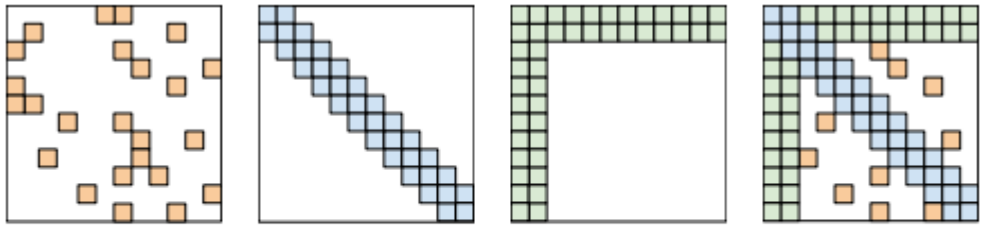
Downstream tasks



Les transformeurs avec patrons spécifiques

3. Le modèle BigBird-ETC

- (Zaheer et al., 2021) propose une alternative aux LongFormers en introduisant des modèles d'attention aléatoires combinés à des modèles à fenêtres et à attention globale.
- L'attention globale porte son attention de façon symétrique à certaines parties du texte seulement.



(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Figure 1: Building blocks of the attention mechanism used in BIGBIRD. White color indicates absence of attention. (a) random attention with $r = 2$, (b) sliding window attention with $w = 3$ (c) global attention with $g = 2$. (d) the combined BIGBIRD model.

Model	MLM	SQuAD	MNLI
BERT-base	64.2	88.5	83.4
Random (R)	60.1	83.0	80.2
Window (W)	58.3	76.4	73.1
R + W	62.7	85.1	80.5
Global + R + W	64.4	87.2	82.9

Similaire à BERT pour une même taille d'entrée sans utiliser toutes les attentions.

Model	HotpotQA			NaturalQ		TriviaQA		WikiHop
	Ans	Sup	Joint	LA	SA	Full	Verified	MCQ
HGN [26]	**82.2**	88.5	**74.2**	-	-	-	-	-
GSAN	81.6	88.7	73.9	-	-	-	-	-
ReflectionNet [32]	-	-	-	77.1	**64.1**	-	-	-
RikiNet-v2 [61]	-	-	-	76.1	61.3	-	-	-
Fusion-in-Decoder [39]	-	-	-	-	-	84.4	90.3	-
SpanBERT [42]	-	-	-	-	-	79.1	86.6	-
MRC-GCN [87]	-	-	-	-	-	-	-	78.3
MultiHop [14]	-	-	-	-	-	-	-	76.5
Longformer [8]	81.2	88.3	73.2	-	-	77.3	85.3	81.9
BIGBIRD-ETC	81.2	**89.1**	73.6	**77.8**	57.9	**84.5**	**92.4**	**82.3**

Table 3: Fine-tuning results on **Test** set for QA tasks. The Test results (F1 for HotpotQA, Natural Questions, TriviaQA, and Accuracy for WikiHop) have been picked from their respective leaderboard. For each task the top-3 leaders were picked not including BIGBIRD-etc. **For Natural Questions Long Answer (LA), TriviaQA, and WikiHop, BIGBIRD-ETC is the new state-of-the-art**. On HotpotQA we are third in the leaderboard by F1 and second by Exact Match (EM).

COURS N°1

Représentation sémantique de texte

Questions supplémentaires?

GREYC
Electronics and Computer Science Laboratory

Normandie Université

UNICAEN

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

cnrs

Plan de l'UE

1. **[CM 1]** Représentation sémantique de texte [GD]
2. **[CM 2]** Cohérence textuelle [MS]
3. **[CM 3]** Modélisation thématique [NA]
4. **[CM 4]** Résumé de textes et traduction automatique [MS]
5. **[CM 5]** Génération langagière I [KM]
6. **[CM 6]** Génération langagière II [KM]
7. **[CM 7]** TAL multimodal [NA]
8. **[CM 8]** TAL et web [MS]
9. **[CM 9]** TAL et handicap visuel [FM]
10. **[CM 10]** TAL et psychiatrie [GD]

11. **[TP 1-5]** Génération neuronal de comptes-rendus médicaux [NA - KM]

TRAITEMENT AUTOMATIQUE DES LANGUES AVANCE

Master Informatique

2ème Année – 1er Semestre

Intervenants CM:

Gaël DIAS (gael.dias@unicaen.fr), Marc SPANIOL, Fabrice MAUREL

Intervenants TP:

Navneet AGARWAL, Kirill MILINTSEVICH

GREYC
Electronics and Computer Science Laboratory

Normandie Université

UNICAEN

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

cnrs