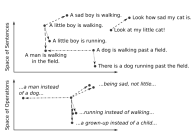
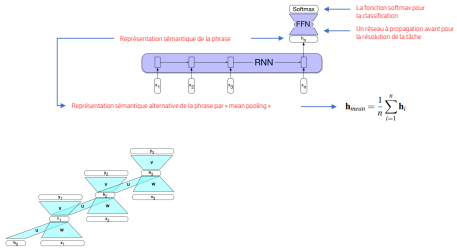
Représentations neuronales au niveau de la phrase

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.

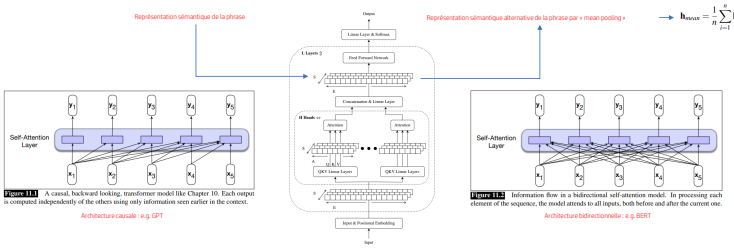


Figure 11.1 A causal, backward looking, transformer model like Chapter 10. Each output is computed independently of the others using only information seen earlier in the context.

Architecture causale : e.g. GPT

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.

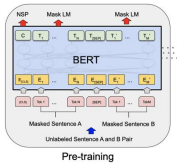
Architecture bidirectionnelle : e.g. BERT

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.`

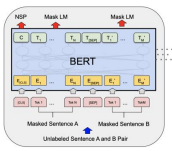
   - Cette technique est utilisée dans le cadre du modèle BERT (Delvin et al., 2019) basé sur des transformeurs bidirectionnels. En particulier, BERT est composé de 12 blocs chacun avec 12 têtes d'attention et la taille des couches cachées est de 768. Les phrases sont décomposées en tokens qui représentent 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 »).
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 y_i du dernier bloc est le plongement du token x_i.
   - Il est possible de faire la moyenne des vecteurs y_i des derniers blocs du transformeur pour atteindre une meilleure représentation.



Pre-training

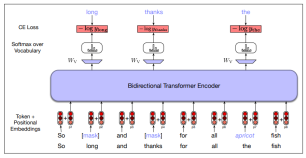


Figure 11.5 Masked language model training. In this example, three of the input tokens are selected, two of which are masked and the third is replaced with an unrelated word. The probabilities assigned by the model to these three items are used as the training loss. (In this and subsequent figures we 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.

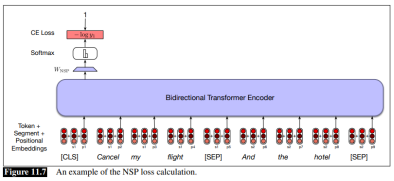
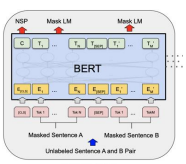


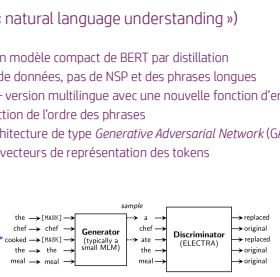
Figure 11.7 An example of the NSP loss calculation.

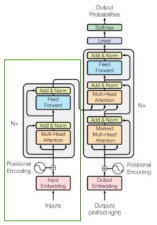


Pre-training

Les modèles du langage pré-entraînés

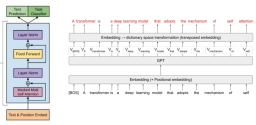
5. Types de modèles pré-entraînés
   - Il existe différents types de modèles du langage pré-entraînés, selon qu'ils ont été entraînés sur des tâches de compréhension de la langue (« natural language understanding ») ou des tâches de génération (« natural language generation ») ou les deux à la fois.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



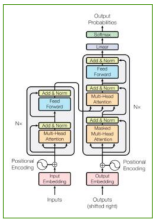


Les modèles du langage pré-entraînés

7. Modèles de type Décodeur (« natural language generation »)
   - GPT (Radford et al., 2018) – modèle causal
   - CTRL (Keskar 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

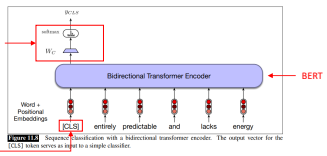


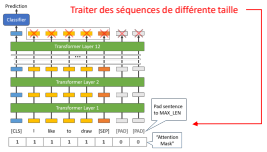
8. Modèles de type Encodeur - Décodeur (NLU + NLG)
   - T5 (Raffel et al., 2020)
   - BART (Lewis et al., 2019) – BERT + GPT
   - M2M-100 (Fan et al., 2020) – modèle multilingue
   - BigBIRD (Zaheer et al, 2020) – optimisé en terme de mémoire utilisée



Le « fine tuning » ou ajustement

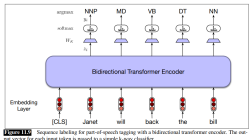
1. Idée générale
   - Le « fine tuning » consiste à utiliser les modèles du langage pré-entraînés comme base de connaissance initiale pour finalement ajuster les poids du réseau pour une application donnée, e.g. analyse de sentiments, l'annotation des entités nommées etc.
   - Intellectuellement, ceci 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.





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.



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.

[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

Le « fine tuning » ou ajustement

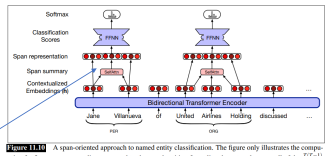
4. Classification de séquences par « fine tuning »
   - Cette tâche consiste à classer des séquences de tokens d'intérêt, comme par exemple les unités polylexicales, des unités étiquetées par des experts (e.g. mémoire épisodique, annotations de psychiatres, etc.). Cette tâche se rapproche des méthodes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

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

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

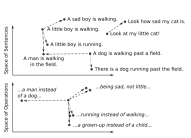
Ainsi, le plongement intermédiaire entre le plongement initial et le plongement final est une moyenne (« average pooling ») de tous les plongements de la séquence pondérés par le mécanisme d'auto-attention.



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

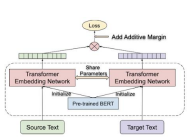
Amélioration de la représentation phrastique

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 ____ .



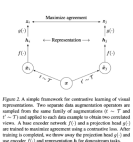


Figure 2: A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder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

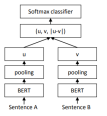


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

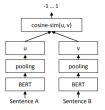


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

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**

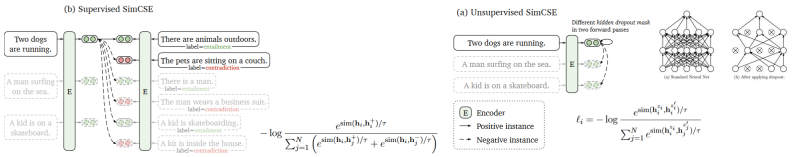
Table 1: Spearman rank correlation ρ between the cosine similarity of sentence representations and the gold labels for various Textual Similarity (STS) tasks. Performance is reported by convention as $\rho \times 100$. STS12-STS16: SemEval 2012-2016, STSb: STSbenchmark, SICK-R: SICK relatedness dataset.

Pooling Strategy	NLI	STSb
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	-
$(u, v, u * v)$	**80.44**	-
$(u, v, \|u - v\|)$	79.23	-
$(u, v, \|u - v\|, u * v)$	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.

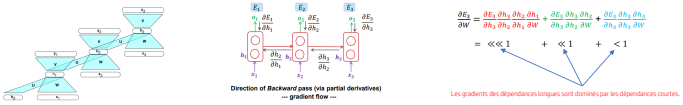


Analyse au niveau du document

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

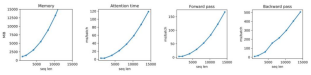
Les limitations des représentations neuronales

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 ».



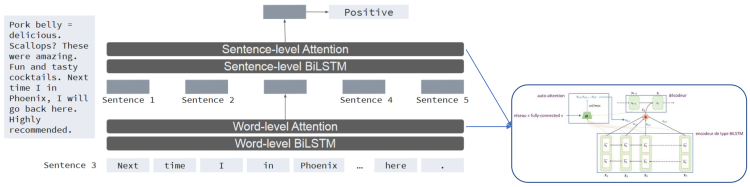
2. Représentation des textes par transformeurs
   - Les transformeurs ont une capacité limitée de représentation du fait de leur non récurrence. Par exemple, pour BERT le nombre de tokens en entrée est limité à 512.
   - Cette limitation est aussi due à la complexité quadratique en coût de calcul: $O(N^2)$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

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



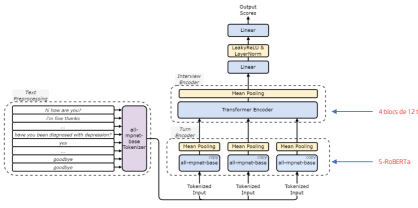
Les modèles hiérarchiques

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

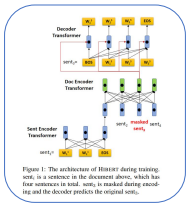


Figure 1: The architecture of HIBERT during training. $sent_i$ is a sentence in the document above, which has four sentences in total. $sent_3$ is masked during encoding and the decoder predicts the original $sent_3$.

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
Inconsist.loss	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$_O$ (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.

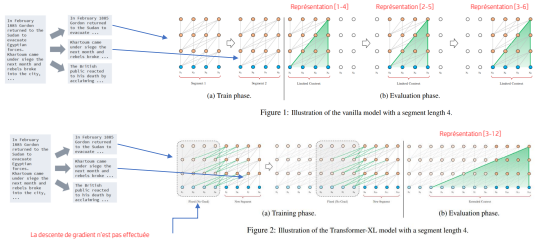


Figure 1: Illustration of the vanilla model with a segment length 4.

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

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)

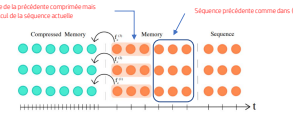


Figure 1: The Compressible Transformer keeps a fine-grained memory of past activations, which are then compressed into coarser *compressed* memories. The above model has three layers, a sequence length $n_s = 3$, memory size $n_m = 6$, compressed memory size $n_{cm} = 6$. The highlighted memories are 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.

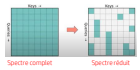
	Valid.	Test
LSTM (Grave et al., 2016)	-	48.7
Temporal CNN (Bai et al., 2018a)	-	45.2
GCNN-14 (Dauphin et al., 2016)	-	37.2
Quasi-RNN Bradbury et al. (2016)	32	33
RMC (Santoro et al., 2018)	30.8	31.9
LSTM+Hebb. (Rae et al., 2018)	29.0	29.2
Transformer (Baevski and Auli, 2019)	-	18.7
18L TransformerXL,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

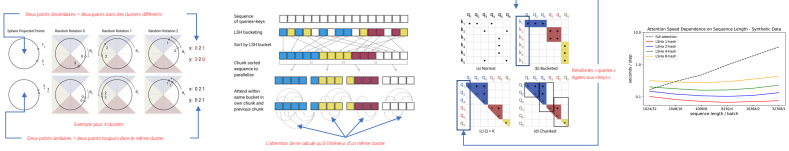
Table 7: WikiText-103 test perplexity broken down by word frequency buckets. The most frequent bucket is words which appear in the training set more than 10, 000 times, displayed on the left. For reference, a uniform model would have perplexity $|V| = 2.6e5$ for all frequency buckets. *LSTM comparison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

	> 10K	1K - 10K	100 - 1K	< 100	All
LSTM*	12.1	219	1,197	9,725	36.4
TransformerXL (ours)	7.8	84.2	488	1,123	18.1
Compressive Transformer	**7.6**	**85.9**	**458**	**937**	**17.1**
Relative gain over TXL	2.6%	9.5%	21%	19.9%	5.8%

Les transformeurs avec patrons de contenus

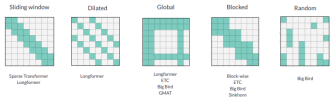
1. Minimiser le spectre d'attention
   - Plutôt que d'introduire de la récurrence, d'autres approches s'appliquent à réduire le spectre des attentions afin de diminuer la complexité de calcul et ainsi permettre d'augmenter la taille des textes en entrée.
2. Minimiser le spectre d'attention par contenus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



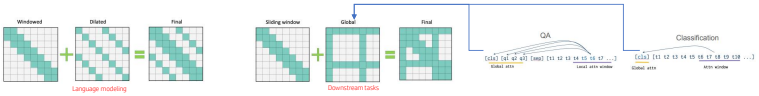


Les transformeurs avec patrons spécifiques

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.



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.

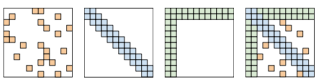


Figure 1: Building blocks of the attention mechanism used in BIGBIRD. White color indicates absence of attention. (a) random attention with $r = 2$, (b) sliding window attention with $w = 3$ (c) global attention with $g = 2$. (d) the combined BIGBIRD model.

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

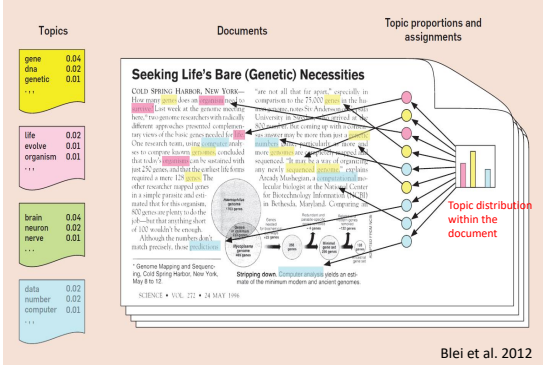
Similaire à BERT pour une même taille d'entrée sans utiliser toutes les attentions.

Model	HotpotQA			NaturalQ		TriviaQA		WikiHop
	Ans	Sup	Joint	LA	SA	Full	Verified	MCQ
HGN [26]	**82.2**	88.5	**74.2**	-	-	-	-	-
GRAN	81.6	88.7	73.9	-	-	-	-	-
ReflectionNet [32]	-	-	-	77.1	**64.1**	-	-	-
RikiNet-v2 [61]	-	-	-	76.1	61.3	-	-	-
ETC (internal) [39]	-	-	-	-	-	84.4	90.3	-
EpsilonBERT [62]	-	-	-	-	-	79.1	86.6	-
MRC-GCN [87]	-	-	-	-	-	-	-	78.3
MultiHop [18]	-	-	-	-	-	-	-	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).

Latent Dirichlet Allocation

- Topics are defined to be a distribution over a fixed vocabulary (vocabulary of entire dataset).
- Each document is defined using distributon over topics and each topic is in-turn a distribution over words in the vocabulary
- Topic distribution defines the contribution of each topic towards the document

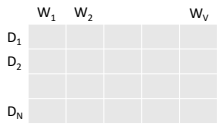


Blei et al. 2012

Latent Dirichlet Allocation

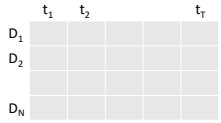
- Dimensionality reduction:
  - # documents: N
  - # words in vocabulary: V

documents can be represented using document-term matrix $\mathbb{R}^{N \times V}$



Assume T topics are learned from this data.

Documents are represented as distribution over topics $\mathbb{R}^{N \times T}$



- Unsupervised learning: can be compared to clustering.

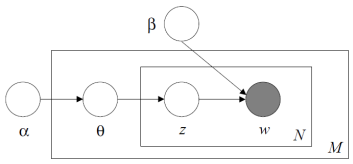
Words are clustered together to form topics based on their co-occurance patterns

Documents are clustered based on their topic distributions.

Blei et al., JMLR 2003

Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generative probablistic model of a corpus. Within LDA, documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words.



Lets assume we want to generate M documents:

1. Choose N: number of words in the document
2. Choose θ ~Dir(α): topic proportions for document w
3. For each of the N words:
   a) Choose a topic z_n ~ Multinomial(θ): topic assignment for document w
   b) Choose a word w_n from p(w|z_n, β)

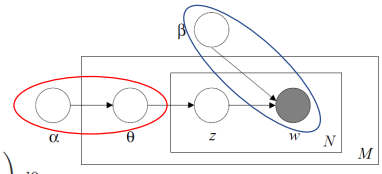
Latent Dirichlet Allocation

Given α and β, the joint distribution of a topic mixture θ, set of *N* topics *z*, and *N* words *w is given by:*

$$p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) = p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta) p(w_n \mid z_n, \beta),$$

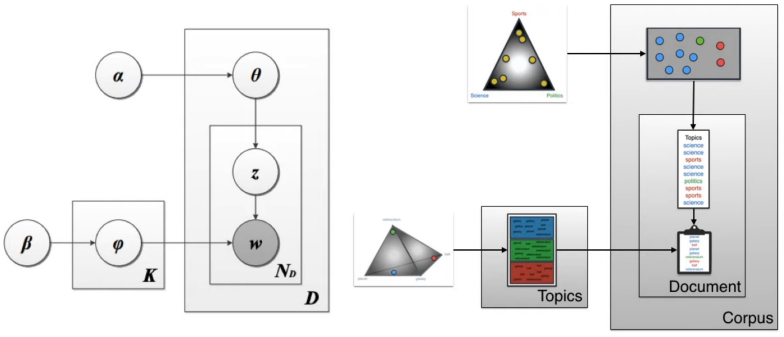
Integrating over θ and summing over *z*, we obtain the marginal distribution of a document:

$$p(\mathbf{w} \mid \alpha, \beta) = \int p(\theta \mid \alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n \mid \theta) p(w_n \mid z_n, \beta) \right) d\theta.$$



Finally, taking product of marginal probabilities of single documents, we obtain the probability of a corpus:

$$p(D \mid \alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d \mid \alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn} \mid \theta_d) p(w_{dn} \mid z_{dn}, \beta) \right) d\theta_d.$$



Latent Dirichlet Allocation

LDA is part of a larger field of *probabilistic modeling.*

We treat the data as arising from a generative process that includes *hidden variables.*

This generative process defines a joint probability distribution over both the observed and hidden variables

We perform data analysis by using joint distribution to compute conditional distribution of the hidden variables given the observed variables.

This conditional distribution is called *posterior distribution.*

observed variables: *w*

hidden variables: θ, z, β

$$p(\theta, z, \beta | w) = \frac{p(\theta, z, \beta, w)}{p(w)}$$

Latent Dirichlet Allocation

- The posterior cannot be computed because the denominator is intractable.
- Topic modeling algorithms form an approximation of the equation by adapting an alternative distribution over the latent topic structures to be close to the true posterior.
- Topic modeling algorithms generally fall into two categories:
  - Sampling based algorithms
  - Variational algorithms
- The most commonly used sampling algorithm for topic modeling is *Gibbs Sampling*

Gibbs Sampling

- Gibbs sampling procedure considers each word token in the text collection in turn.
- Estimate the probability of assigning the current word token to each topic, conditioned on the topic assignments to all the other word tokens.

$$P(z_i = j \mid \mathbf{z}_{-i}, w_i, d_i, \cdot) \propto \frac{C^{WT}_{w_i j} + \beta}{\sum_{w=1}^{W} C^{WT}_{wj} + W\beta} \frac{C^{DT}_{d_i j} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d_i t} + T\alpha}$$

- From this conditional distribution, a topic is sampled and stored as the new assignment for this word token.

Griffiths and Steyvers (2004)

Gibbs Sampling

- n_{dk}: number of words assigned to topic k in document d
- n_{kw}: number of times word w is assigned to topic k
- n_k: number of times any word is assigned to topic k

Input: words $\mathbf{w} \in$ documents $\mathbf{d}$
Output: topic assignments $\mathbf{z}$ and counts $n_{d,k}, n_{k,w}$, and n_k
begin
  randomly initialize $\mathbf{z}$ and increment counters
  foreach *iteration* **do**
    for $i = 0 \rightarrow N-1$ **do**
      $word \leftarrow w[i]$
      $topic \leftarrow z[i]$
      $n_{d,topic}$-=1; $n_{word,topic}$-=1; n_{topic}-=1
      for $k = 0 \rightarrow K-1$ **do**
        $p(z = k|\cdot) = (n_{d,k} + \alpha_k)\frac{n_{k,w}+\beta_w}{n_k+\beta\times W}$
      end
      $topic \leftarrow$ sample from $p(z|\cdot)$
      $z[i] \leftarrow topic$
      $n_{d,topic}$+=1; $n_{word,topic}$+=1; n_{topic}+=1
    end
  end
  return $\mathbf{z}$, $n_{d,k}, n_{k,w}, n_k$
end

Gibbs Sampling

Example

Assume we have some document with random word-topic assignment

India	enters	world	cup	final
1	3	1	2	4

We have count matrix C^{WT}

	1	2	3	4
India	70	5	0	8
enters	2	3	15	6
world	28	4	12	1
cup	6	43	6	0
final	7	0	9	31

Gibbs Sampling

Example

Assume we have some document with random word-topic assignment

India	enters	world	cup	final
1	3	1	2	4

We have count matrix C^{WT}

	1	2	3	4
India	70	5	0	8
enters	2	3	15	6
world	28	4	12	1
cup	6	43	6	0
final	7	0	9	31

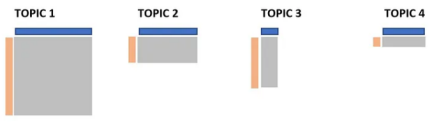
Gibbs Sampling

- *Consider the conribution of each topic towards this document*

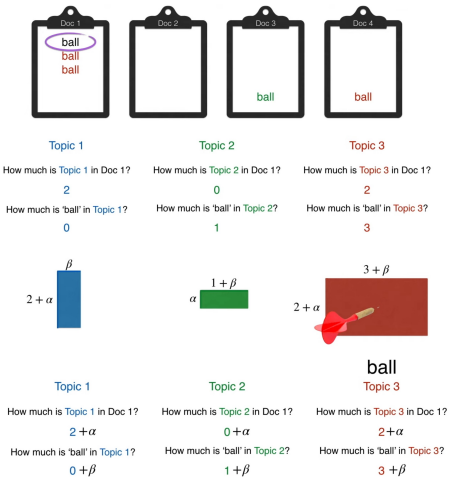
- *Next, we take how many times each topic is assigned to this word*

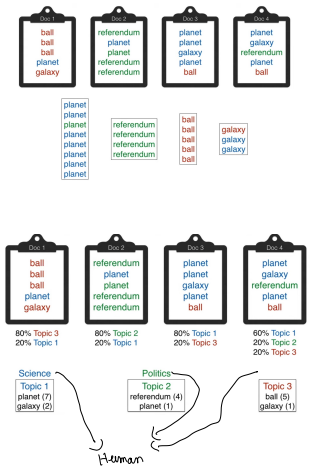
Gibbs Sampling

- *Multiply these values to get conditional probabilities*



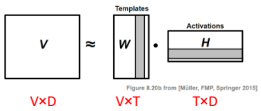
- *Finally, pick one of the topics from this distribution and update the variables accordingly.*
- *Repeat this for every word.*





Nonnegative Matrix Factorization (NMF)

1. Nonnegative Matrix Factorization (NMF) factors input nonnegative matrix V into two nonnegative matrices W and H.



$$V \approx W \cdot H.$$

- W and H are required to have much lower rank that the original matrix V.
- Columns of V contains V-dimensional data vectors
- Columns of W are the word distribution for each topic.
- Rows of H are the activation of given topic across all documents.
- In most cases the factorization does not have an exact solution and requires optimization procedures to find numerical approximations.

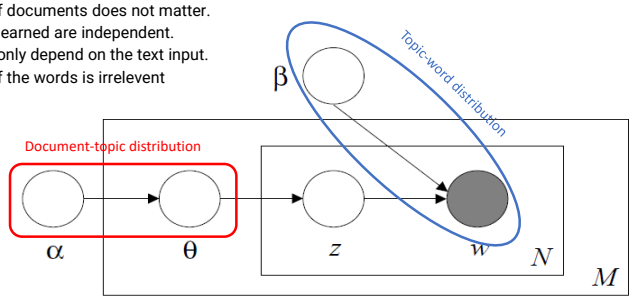
$$||V - WH||^2$$

LDA vs NMF

- LDA is probabilistic while NMF uses matrix factorization.
- LDA extracts independent topics from word distributions. Therefore, topics that are dissimilar in the document may not be identified seperately.
- NMF learns dissimilar topics, but can cause difficulties in interpreting findings.
- NMF usually performs better with short texts, like social media data.
- For both LDA and NMF, the results are highly dependent on hyperparameter tuning.

Topic model variations

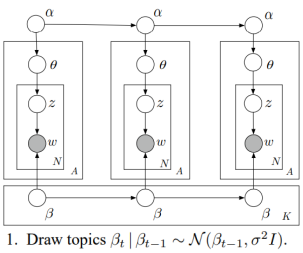
1. Order of documents does not matter.
2. Topics learned are independent.
3. Topics only depend on the text input.
4. Order of the words is irrelevent



Blei, David M., and John D. Lafferty. "Dynamic topic models." *Proceedings of the 23rd international conference on Machine learning*. 2006.

Topic model variations

1. Dynamic topic model
   - LDA assumes that order of documents does not matter.
   - This assumption may be unrealistic when considering long running collections that span years or centuries.
   - Dynamic topic model solves this problem by dividing documents based on time slots.
   - Topic models learned for each time slot are dependent on respective topic from previous time slot.

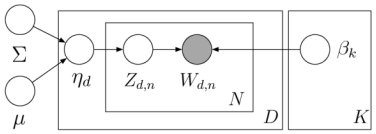


1. Draw topics $\beta_t \mid \beta_{t-1} \sim \mathcal{N}(\beta_{t-1}, \sigma^2 I)$.
2. Draw $\alpha_t \mid \alpha_{t-1} \sim \mathcal{N}(\alpha_{t-1}, \delta^2 I)$.
3. For each document:
   (a) Draw $\eta \sim \mathcal{N}(\alpha_t, a^2 I)$
   (b) For each word:
      i. Draw $Z \sim Mult(\pi(\eta))$.
      ii. Draw $W_{t,d,n} \sim Mult(\pi(\beta_{t,z}))$.

Blei, David M., and John D. Lafferty. "Dynamic topic models." *Proceedings of the 23rd international conference on Machine learning*. 2006.

Topic model variations

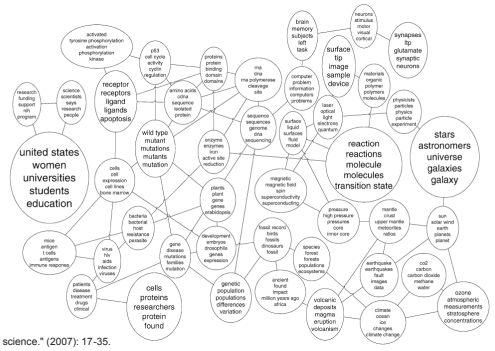
1. Correlated topic model
   - Within LDA, topics are sampled from a Dirichlet distribution are independent which is not realistic for real document collections.
   - CTM draws real values random vectors from multivariate Gaussian distribution inducing dependencies between the components.



Blei, David M., and John D. Lafferty. "A correlated topic model of science." (2007): 17-35.

Topic model variations

1. Correlated topic model
   - Topic graph learned from 16,351 OCR articles from science (1990-1999)



Blei, David M., and John D. Lafferty. "A correlated topic model of science." (2007): 17-35.

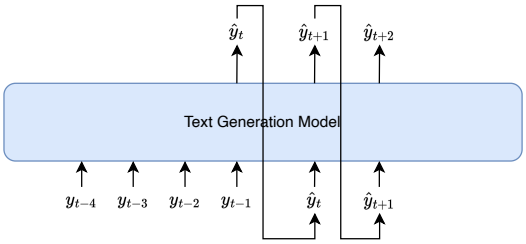
Topic model variations

1. Structured Topic Model (STM)

- LDA learns only based on input text.
- Certain sources may be more likely to write about politics.
- Metadata can include date published, author, publication, likes on social media, etc.
- Within LDA our topic distribution comes from Dirichlet distrubution.
- STM defines topic distributions based on document metadata
- We need to go from X_y *1xp* metadata vector to *1xk* vector of topic distribution.
- We multiply X with pxk weight matrix t.

Autoregressive Models

- In autoregressive text generation models, at each time step t, our model takes in a sequence of tokens of text as input $y_{<t}$ and outputs a new token, $\hat{y}_t$



During a Single Step

- At each time step t, our model computes a vector of scores for each token in our vocabulary, $S \in \mathbb{R}^V$:

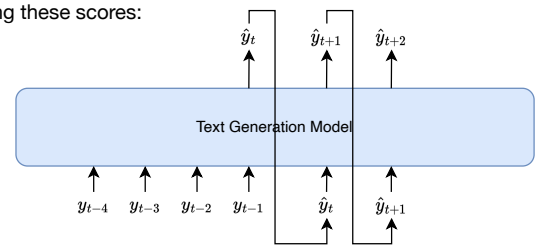
$$S = f(\{y_{<t}\}, \theta)$$

- Then, we compute a probability distribution P over $w \in V$ using these scores:

$$P(y_t \mid \{y_{<t}\}) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

Autoregressive Models

- At each time step t, our model computes a vector of scores for each token in our vocabulary, $S \in \mathbb{R}^V$. Then, we compute a probability distribution P over $w \in V$ using these scores:



Inference and Training

- At inference time, our decoding algorithm defines a function to select a token from this distribution:

$$\hat{y}_t = g(P(y_t \mid \{y_{<t}\}))$$

- We train the model to minimize the negative loglikelihood of predicting the next token in the sequence:

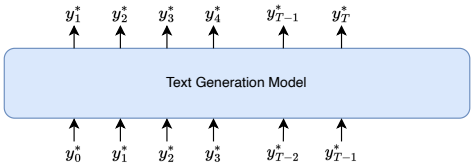
$$L_t = -\log P(y_t^* \mid \{y_{<t}^*\})$$

Maximum Likelihood Training

Teacher Forcing

- Trained to generate the next word y_t^* given a set of preceding words $\{y^*\}_{<t}$

$$L = -\sum_{t=1}^{T-t} \log P(y_t^* \mid \{y_{<t}^*\})$$



Decoding

- At each time step t, our model computes a vector of scores for each token in our vocabulary, $S \in \mathbb{R}^V$:

$$S = f(\{y_{<t}\}, \theta)$$

- Then, we compute a probability distribution P over $w \in V$ using these scores:

$$P(y_t \mid \{y_{<t}\}) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

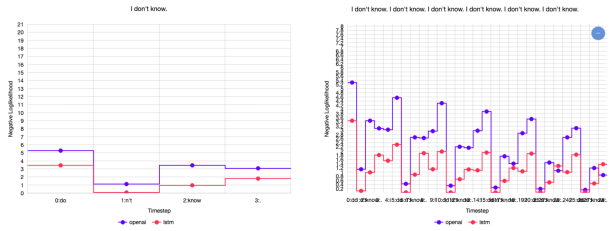
- Our decoding algorithm defines a function to select a token from this distribution:

$$\hat{y}_t = g(P(y_t \mid \{y_{<t}\}))$$

Greedy Methods

- Argmax Decoding
  - Selects the highest probability token in $P(y_t \mid y_{<t})$
- Beam Search

Repetition in Greedy Methods

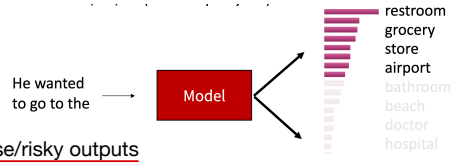


Top-k sampling

- Problem: Vanilla sampling makes every token in the vocabulary an option
  - Even if most of the probability mass in the distribution is over a limited set of options, the tail of the distribution could be very long
  - Many tokens are probably irrelevant in the current context
  - Why are we giving them individually a tiny chance to be selected?
  - Why are we giving them as a group a high chance to be selected?
- Solution: Top-k sampling
  - Only sample from the top k tokens in the probability distribution

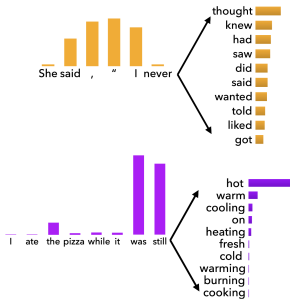
Top-k sampling

- Only sample from the top k tokens in the probability distribution
- Common values are k = 5, 10, 20 (but it's up to you!)



- Increase k for more diverse/risky outputs
- Decrease k for more generic/safe outputs

Top-k sampling



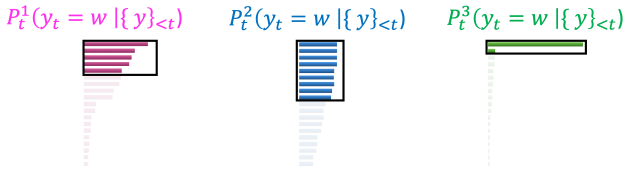
Top-*k* sampling can cut off too ***quickly***!

Top-*k* sampling can also cut off too ***slowly***!

Top-p (nucleus) sampling

- Problem: The probability distributions we sample from are dynamic
  - When the distribution P_t is flatter, a limited k removes many viable options
  - When the distribution P_t is peakier, a high k allows for too many options to have a chance of being selected
- Solution: Top-p sampling
  - Sample from all tokens in the top p cumulative probability mass (i.e., where mass is concentrated)
  - Varies k depending on the uniformity of P_t

Top-p (nucleus) sampling



Softmax Temperature

- On timestep t, the model computes a prob distribution P_t by applying the softmax function to a vector of scores $s \in \mathbb{R}^V$

$$P(y_t \mid \{y_{<t}\}) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

- You can apply a temperature hyperparameter τ to the softmax to rebalance P_t:

$$P(y_t \mid \{y_{<t}\}) = \frac{\exp(S_w/\tau)}{\sum_{w' \in V} \exp(S_{w'}/\tau)}$$

Softmax Temperature

- Raise the temperature $\tau > 1$: P_t becomes more uniform
  - More diverse output (probability is spread around vocab)
- Lower the temperature $\tau < 1$: P_t becomes more spiky
  - Less diverse output (probability is concentrated on top words)

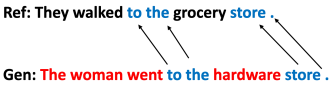
Re-ranking

- Decode a bunch of sequences
  - 10 candidates is a common number
- Define a score to approximate quality of sequences and re-rank by this score
  - Simplest is to use perplexity
    - Careful! Remember that repetitive methods can generally get high perplexity.

Decoding

- Decoding is still a challenging problem in natural language generation
- Human language distribution is noisy and doesn't reflect simple properties (i.e., probability maximization)
- Different decoding algorithms can allow us to inject biases that encourage different properties of coherent natural language generation
- Some of the most impactful advances in NLG of the last few years have come from simple, but effective, modifications to decoding algorithms

Content Overlap Metrics



- Compute a score that indicates the similarity between generated and gold-standard (human-written) text
- Fast and efficient and widely used
- Two broad categories:
  - N-gram overlap metrics (e.g., BLEU, ROUGE, METEOR, CIDEr, etc.)
  - Semantic overlap metrics (e.g., PYRAMID, SPICE, SPIDEr, etc.)

Content Overlap Metrics

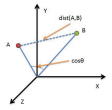
- They're not ideal for machine translation
- They get progressively much worse for tasks that are more open-ended than machine translation
  - Worse for summarization, where extractive methods that copy from documents are preferred
  - Much worse for dialogue, which is more open-ended that summarization
  - Much, much worse story generation, which is also open-ended, but whose sequence length can make it seem you're getting decent scores!

Dans le cas de la summarization, où des méthodes extractives (copier des parties du texte source) sont souvent préférées, les métriques de recouvrement de contenu peuvent ne pas bien refléter la qualité du résumé, car la redondance peut être considérée comme une bonne pratique.

Les métriques de recouvrement de contenu peuvent être encore moins adaptées pour évaluer la qualité des dialogues. Les conversations sont souvent évaluées en fonction de la cohérence, de la pertinence contextuelle, et de la manière dont elles maintiennent une interaction significative, des aspects qui ne sont pas capturés efficacement par la simple comparaison du contenu.

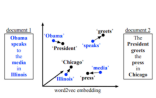
La génération d'histoires est une tâche très ouverte où la créativité et la cohérence narrative sont cruciales. Les métriques de recouvrement de contenu peuvent ne pas saisir ces aspects, en particulier lorsque les histoires sont longues et complexes.

Model-Based Metrics



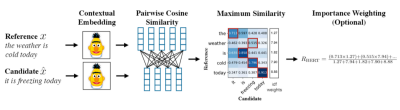
Vector Similarity:
Embedding based similarity for semantic distance between text.

- **Embedding Average** (Liu et al., 2016)
- **Vector Extrema** (Liu et al., 2016)
- **MEANT** (Lo, 2017)
- **YISI** (Lo, 2019)



Word Mover's Distance:
Measures the distance between two sequences (e.g., sentences, paragraphs, etc.), using word embedding similarity matching.
(Kusner et.al., 2015; Zhao et al., 2019)

BERTSCORE:
Uses pre-trained contextual embeddings from BERT and matches words in candidate and reference sentences by cosine similarity.
(Zhang et.al. 2020)



Human Evaluation

- Ask humans to evaluate the quality of generated text
- Human judgments are regarded as the gold standard
- Humans are inconsistent

Non-autoregressive Models

Application	Example Source (S) and target (T) text	Use seq2seq
Machine translation	S: Turing studied at King's College, where he was awarded first-class honours in mathematics. T: Turing studierte am King's College, wo er erstklassige Auszeichnungen in Mathematik erhielt.	✓
Summarization	S: Court members Deborah Poritz and Peter Verniero did not participate in the Nelson case. T: Court members didn't participate in the case.	?
Sentence fusion	S: Turing was born in 1912. Turing died in 1954. T: Turing was born in 1912 and he died in 1954.	✗
Grammar correction	S: New Zealand have a cool weather. T: New Zealand has cool weather.	✗

Non-autoregressive Models

- Most NLP tasks apart from Machine Translation are monolingual
- Sources and targets often overlap
  - Generating the target from scratch is wasteful
  - Can reconstruct most of the target from the source via basic operations like KEEP, DELETE, INSERT

Turing	was	born	in	1912	.	Turing	died	in	1954	.
KEEP	KEEP	KEEP	KEEP	KEEP	DEL INS	PRON	KEEP	KEEP	KEEP	KEEP
Turing	was	born	in	1912	and	he	died	in	1954	.

Applications

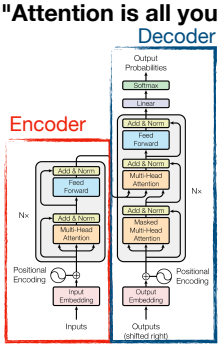
- Grammatical Error Correction
- Text Simplification
- Sentence fusion
- Style transfer
- Text normalisation
- Text summarisation
- Automatic post-editing for machine translation

Advantages

- Text editing models need less training data
- They are faster at the inference
- They are more faithful
- We can control what model adds or removes
- We can incorporate external knowledge

Transformer Architecture

Vaswani, Ashish, et al. "Attention is all you need." (2017)



Byte-Pair Encoding

Training

- First, we compute the unique set of words in the corpus
- Split each word into characters
- Start merging most frequent pairs

- ("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)
- ("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

Byte-Pair Encoding
Training

- First, we compute the unique set of words in the corpus
- Split each word into characters
- Start merging most frequent pairs
- The most frequent pair is `("u", "g")`

- ("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)
- ("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

Vocabulary: `["b", "g", "h", "n", "p", "s", "u", "ug"]`

Corpus: `("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "ug" "s", 5)`

Byte-Pair Encoding
Training

- First, we compute the unique set of words in the corpus
- Split each word into characters
- Start merging most frequent pairs
- The most frequent pair is `("u", "n")`

- ("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)
- ("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

Vocabulary: `["b", "g", "h", "n", "p", "s", "u", "ug", "un"]`

Corpus: `("h" "ug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("h" "ug" "s", 5)`

Byte-Pair Encoding
Training

- First, we compute the unique set of words in the corpus
- Split each word into characters
- Start merging most frequent pairs
- The most frequent pair is `("h", "ug")`

- ("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)
- ("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12), ("b" "u" "n", 4), ("h" "u" "g" "s", 5)

Vocabulary: `["b", "g", "h", "n", "p", "s", "u", "ug", "un", "hug"]`

Corpus: `("hug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("hug" "s", 5)`

Byte-Pair Encoding
Tokenization

1. Normalization
2. Pre-tokenization
3. Splitting the words into individual characters
4. Applying the merge rules learned in order on those splits

Byte-Pair Encoding
Tokenization

1. Normalization
2. Pre-tokenization
3. Splitting the words into individual characters
4. Applying the merge rules learned in order on those splits

Learned rules:

`("u", "g") -> "ug"`

`("u", "n") -> "un"`

`("h", "ug") -> "hug"`

Byte-Pair Encoding
Tokenization

1. Normalization
2. Pre-tokenization
3. Splitting the words into individual characters
4. Applying the merge rules learned in order on those splits

Learned rules:

`("u", "g") -> "ug"`

`("u", "n") -> "un"`

`("h", "ug") -> "hug"`

`"bug" -> "b", "ug" | "mug" -> "[UNK]", "ug"`

WordPiece Tokenization

Training

- Same as BPE but different merging rule

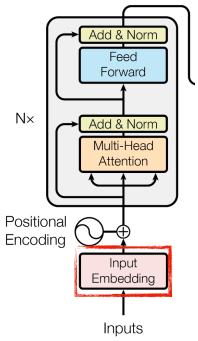
```
score=(freq_of_pair)/
(freq_of_first_element×freq_of_second_element)
```

WordPiece Tokenization

Tokenization

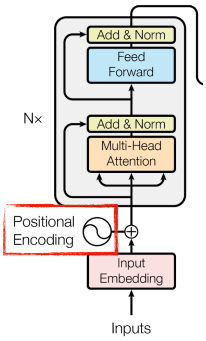
- Same as BPE but the WordPiece starts by searching for the longest sub-token in the vocabulary

Transformer Encoder



- Byte-pair Encoding (GPT, GPT-2, RoBERTa, BART, and DeBERTa) or WordPiece (BERT) tokenization
- Randomly initialized learnable embedding layer

Transformer Encoder



- Byte-pair Encoding (GPT, GPT-2, RoBERTa, BART, and DeBERTa) or WordPiece (BERT) tokenization
- Randomly initialized learnable embedding layer
- Positional encoding to add information about words' position in the sequence

Positional Encoding

- Since Transformer doesn't have recursion it lacks information about words' positions

Let t be the desired position in an input sentence, $\vec{p_t} \in \mathbb{R}^d$ be its corresponding encoding, and d be the encoding dimension (where $d \equiv_2 0$)
Then $f : \mathbb{N} \to \mathbb{R}^d$ will be the function that produces the output vector $\vec{p_t}$ and it is defined as follows:

$$\vec{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k + 1 \end{cases}$$

where

$$\omega_k = \frac{1}{10000^{2k/d}}$$

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Positional Encoding

Intuition

0 :	0 0 0 0	8 :	1 0 0 0
1 :	0 0 0 1	9 :	1 0 0 1
2 :	0 0 1 0	10 :	1 0 1 0
3 :	0 0 1 1	11 :	1 0 1 1
4 :	0 1 0 0	12 :	1 1 0 0
5 :	0 1 0 1	13 :	1 1 0 1
6 :	0 1 1 0	14 :	1 1 1 0
7 :	0 1 1 1	15 :	1 1 1 1

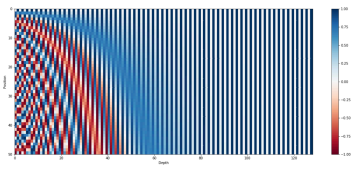
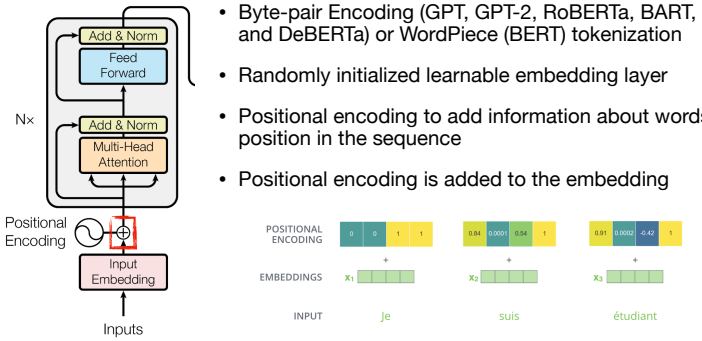


Figure 2 - The 128-dimensional positional encoding for a sentence with the maximum lenght of 50. Each row represents the embedding vector $\vec{p_t}$

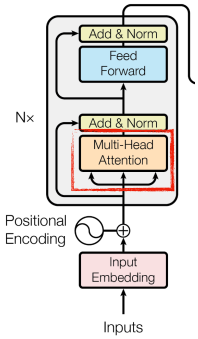
https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Transformer Encoder



- Byte-pair Encoding (GPT, GPT-2, RoBERTa, BART, and DeBERTa) or WordPiece (BERT) tokenization
- Randomly initialized learnable embedding layer
- Positional encoding to add information about words' position in the sequence
- Positional encoding is added to the embedding

Transformer Encoder



- Byte-pair Encoding (GPT, GPT-2, RoBERTa, BART, and DeBERTa) or WordPiece (BERT) tokenization
- Randomly initialized learnable embedding layer
- Positional encoding to add information about words' position in the sequence
- Positional encoding is added to the embedding
- MHA used to redistribute the information among the inputs (contextualization)

Multi-Head Self-Attention

Vaswani, Ashish, et al. "Attention is all you need." (2017)

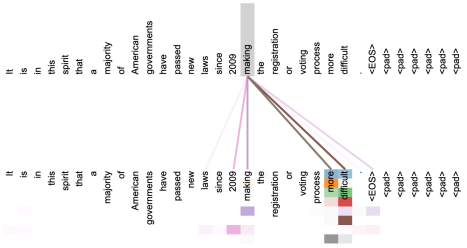
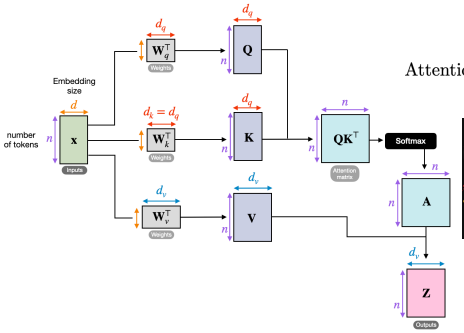


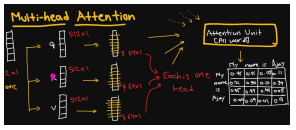
Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.

Multi-Head Self-Attention

https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html

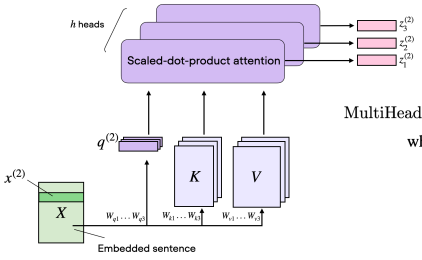


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



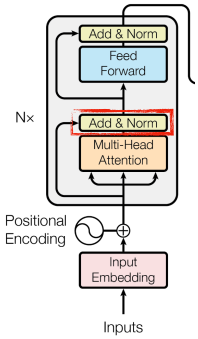
Multi-Head Self-Attention

https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html



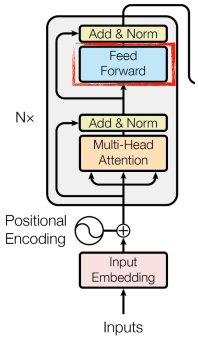
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_\text{i}^Q, KW_\text{i}^K, VW_\text{i}^V)$$

Transformer Encoder



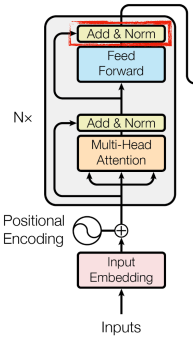
- Byte-pair Encoding (GPT, GPT-2, RoBERTa, BART, and DeBERTa) or WordPiece (BERT) tokenization
- Randomly initialized learnable embedding layer
- Positional encoding to add information about words' position in the sequence
- Positional encoding is added to the embedding
- MHA used to redistribute the information among the inputs (contextualization)
- Add the input information to the result of MHA and normalize

Transformer Encoder



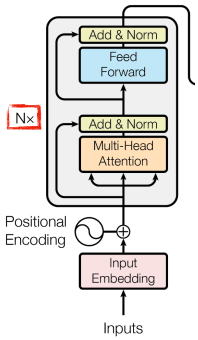
- Byte-pair Encoding (GPT, GPT-2, RoBERTa, BART, and DeBERTa) or WordPiece (BERT) tokenization
- Randomly initialized learnable embedding layer
- Positional encoding to add information about words' position in the sequence
- Positional encoding is added to the embedding
- MHA used to redistribute the information among the inputs (contextualization)
- Add the input information to the result of MHA and normalize
- Just a regular feed-forward network

Transformer Encoder



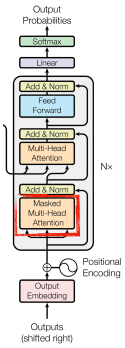
- Byte-pair Encoding (GPT, GPT-2, RoBERTa, BART, and DeBERTa) or WordPiece (BERT) tokenization
- Randomly initialized learnable embedding layer
- Positional encoding to add information about words' position in the sequence
- Positional encoding is added to the embedding
- MHA used to redistribute the information among the inputs (contextualization)
- Add the input information to the result of MHA and normalize
- Just a regular feed-forward network
- Another residual connection and layer normalization

Transformer Encoder

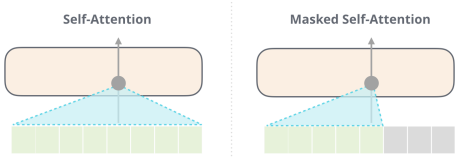


- Byte-pair Encoding (GPT, GPT-2, RoBERTa, BART, and DeBERTa) or WordPiece (BERT) tokenization
- Randomly initialized learnable embedding layer
- Positional encoding to add information about words' position in the sequence
- Positional encoding is added to the embedding
- MHA used to redistribute the information among the inputs (contextualization)
- Add the input information to the result of MHA and normalize
- Just a regular feed-forward network
- Another residual connection and layer normalization
- Repeat N times

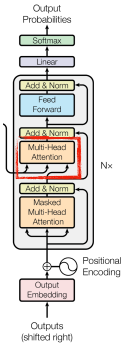
Transformer Decoder



- Has the same elements as encoder
- Masked MHA is used so that the model cannot "look into the future"

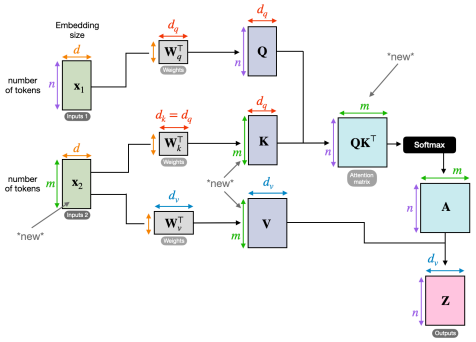


Transformer Decoder

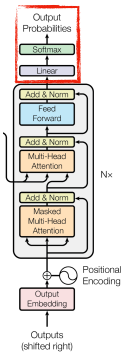


- Has the same elements as encoder
- Masked MHA is used so that the model cannot "look into the future"
- Cross-attention with encoder outputs

Cross-Attention



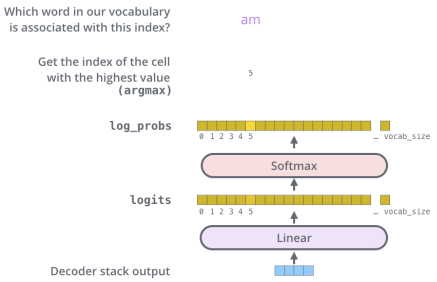
Transformer Decoder



- Has the same elements as encoder
- Masked MHA is used so that the model cannot “look into the future”
- Cross-attention with encoder outputs
- Final layer to output the next token probabilities

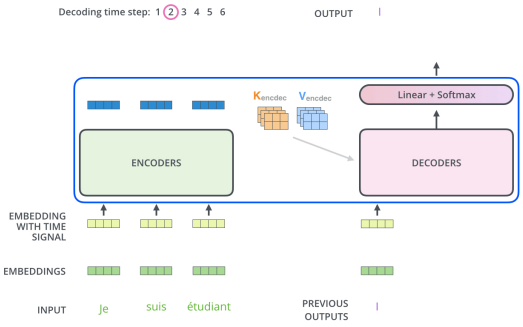
Transformer Final Layer

https://jalammar.github.io/illustrated-transformer/



Encoder-Decoder in Practice

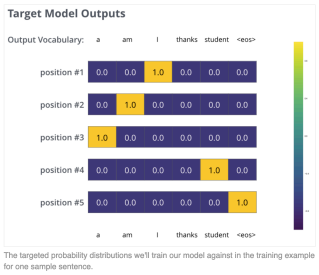
https://jalammar.github.io/illustrated-transformer/

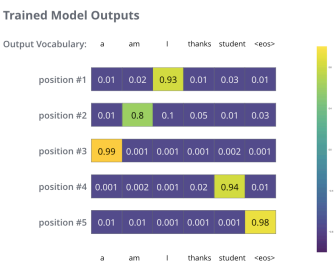


Training

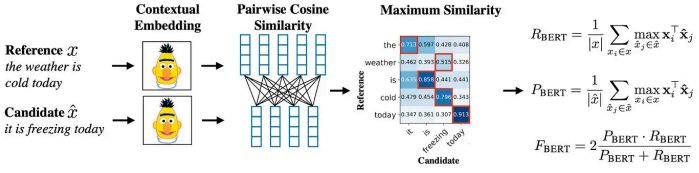
Cross-Entropy Loss

$$H(P^*|P) = -\sum_i \underbrace{P^*(i)}_{\text{TRUE CLASS DISTRIBUTION}} \log \underbrace{P(i)}_{\text{PREDICTED CLASS DISTRIBUTION}}$$





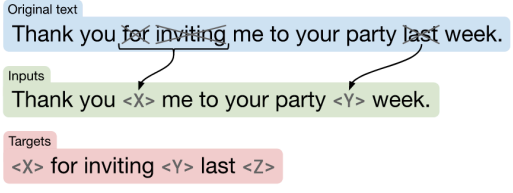
BERT Score



Source: Bertscore: Evaluating text generation with bert

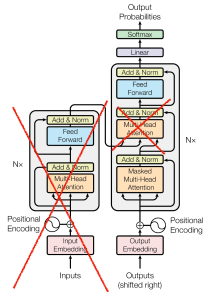
T5

Pre-training



GPT-2

Radford, Alec, et al. "Language models are unsupervised multitask learners." (2019)



Sentiment Analysis

Sentiment analysis (often refered to as opinion mining) is the process of gathering and analyzing people's opinions, thoughts and impressions regarding various topics, products, subjects and services.

- Rapid growth of internet based applications such as social media and blogs.
- People's opinions can be beneficial for
  - Corporations: product reviews on amazon, sentiment towards a certain feature or service.
  - Governments: public opinion on policies, expected voting tendencies before election.
  - Entertainment industry: movie reviews, game reviews.
  - Travel: reviews of restaurants, hotels, tourist attractions, etc.

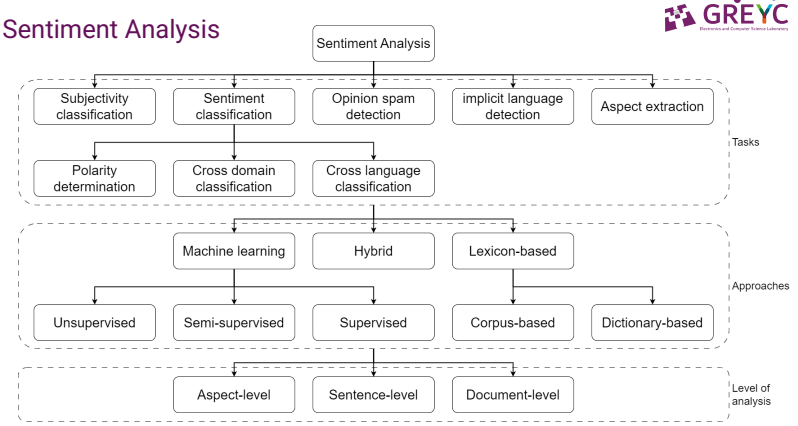
Sentiment Analysis

Sentiment analysis is a broad concept that consists of many different tasks, approaches and types of analysis.

Cambri at al. argue that a holistic approach is required , and only classification or categorization is not sufficient. They present it as a 3 layered problem that includes 15 NLP problems:

- Syntactic layer: Microtext normalization, sentence boundary disambiguation, POS tagging, text chunking and lemmatization.
- Semantics layer: Word sense disambiguation, concept extraction, named entity recognition, anaphora resoluion and subjectivity detection.
- Pragmatics layer: Personality recognition, sarcasm detection, metaphor understanding, aspect extraction and polarity detection.

Sentiment Analysis



Tasks

❑ Sentiment classification

- Most widely known and researched task in sentiment analysis.
- It can be divided into three major sub-tasks: polarity classification, cross-domain classification and cross-language classification.
- Polarity is usually classified as positive or negative with some researchers including a third category neutral.
- Cross-domain classification models transfer knowledge learned from data-rich source domain to a target domain where data and\or labels are limited.
  - Extract domain invarient features whose distribution in the source domain is close to that in target domain
- Cross-language classification fulfills the same function but for languages.
  - An example can be to train the model in source language with abundant data and testing it on target language by translating the input to source language.

Tasks

❑ Subjectivity classification

- The goal of subjectivity classifiation is to restrict unwanted objective data objects for further processing.
- It detects subjective clues, words that carry emotion or subjective notion like 'expensive', 'easy', 'better', etc.
- These clues are used to classify objects as subjective or objective.

Ce livre coûte 10 euros → Cannot be used for sentiment analysis

Ce livre est cher → Can be used for sentiment analysis

Tasks

- Opinion spam detection
  - Opinion spams refer to fake or false reviews intelligently written to either promote or discredit a product.
  - Three main features are considered within this context:
    - Review content: the actual text of the review
    - Meta-data: information like IP address, geo-location, user-id, etc.
    - Real-life knowledge: this meathod utilizes learned experiences to classify spam. For example, if a aproduct has good reputation and suddenly inferior ratings are given over a period, reviews of that period might be suspected.

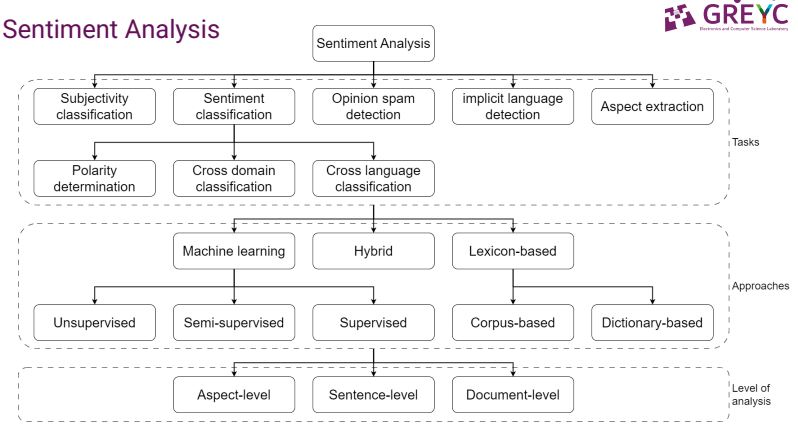
Tasks

- Implicit language detection
  - Implicit language includes humor, sarcasm ans irony.
  - There is vagueness and ambiguity in this form of speech, which is sometimes hard to detect.
  - An implicit meaning can sometimes completly flip the polarity of a sentence.
  - Example, « I love pain », pain is a factual word with negative polarity. The contradiction between pain and love can indicate sarcasm.
  - Traditional methods for detection include exploring emoticons, expressions of laughter and heavy punctuation mark usage.

Tasks

- Aspect extraction
  - Aspect extraction refers to retriving the target entity and the aspects of the target entity in the document. The target entity can be a person, product, event, etc.
  - Aspect extraction is particularly important in sentiment analysis of social media and blogs that often do not have predefined topics.
  - Most traditional method for this is frequency based where most frequent nouns and compound nouns are considered as candidates for aspects.
    - Not all nouns are aspects
  - Syntax-based methods find aspects by means of syntactic relations they are in. For example, identifying aspects that are preceded by a modifying adjective that is a sentiment word.
    - Many relations need to be found for complete coverage

Sentiment Analysis



Lexicon based approaches

- Traditional approach for sentiment analysis that scans through the text for words that express positive or negative feelings to humans.
- It shows to be extremly dependent on domain of interest due to differences in language usage between domains.
- There are two main approaches to creating sentiment lexicons: dictionary based and corpus based.
- Dictionary based appraches start with an initial list of terms and iteratively expand the lexicon by adding synonyms and antonyms of the the terms to the list.
- They work best for general purpose use.
- Corpus based approaches starts with general purpose list of words and finds others terms from domain specific copus based on co-occurring word patterns.

Image Captioning

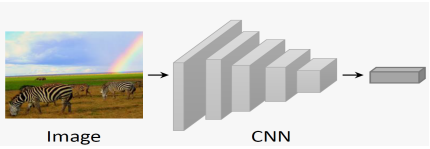
Global CNN features

With advent of CNNs, all models consuming visual inputs have been improved in terms of performance

In the most simple recipe, the activation of one of the last layers of a CNN is employed to extract high-level representations, which are then used within language models for generating final text.

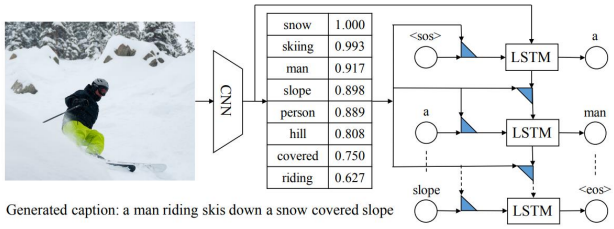
Advantages
- Simplicity
- Compactness of representations

Disadvantages
- Excessive compression of information
- Lack of granularity
- Unable to produce specific and fine-grained descriptions



Global CNN features

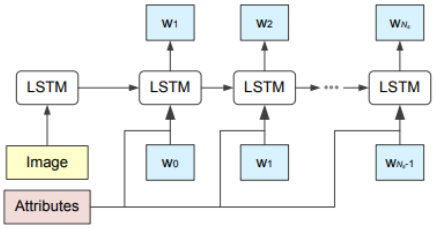
Gan et al., CVPR 2017
- ResNet-152 pre-trained on imagenet dataset



Generated caption: a man riding skis down a snow covered slope

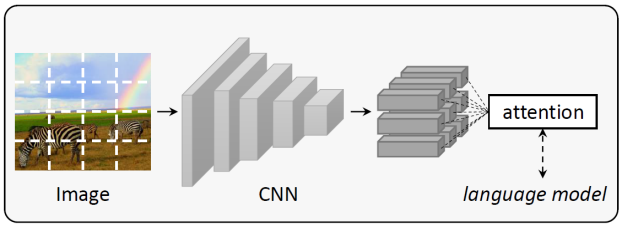
Global CNN features

Yao et al., Boosting image captioning with attributes, ICCV 2017
- 1,000 most common words on COCO as the high-level attributes and train the attribute detectors with MIL model (Fang et al., 2015)
- GoogleNet for image encoding



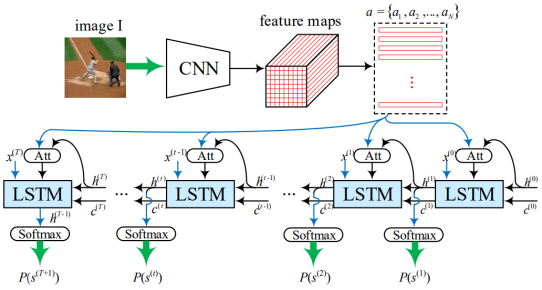
Attention based models

Grid based attention
- Improves upon the drawbacks of global representations and increases the granularity level of visual encoding
- Motivated from use of attention in machine translation.



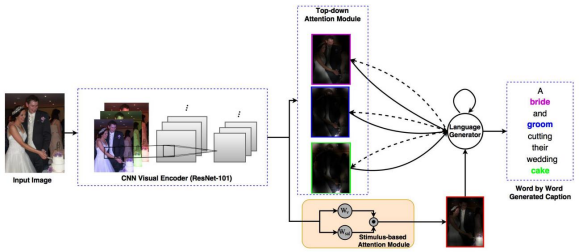
Attention based models

Ge et al., Exploring Overall Contextual Information for Image Captioning in Human-Like Cognitive Style, ICCV 2019



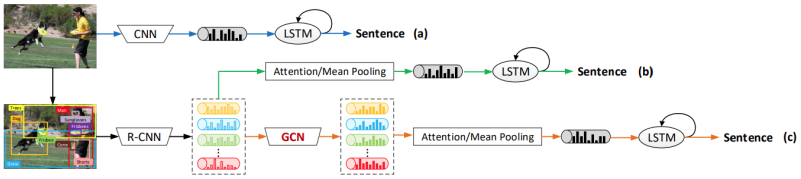
Attention based models

Chen et al., Boosted attention: leveraging human attention for image captioning, ECCV, 2018
- Improve the attention mechanism by incorporating human attention input into the model.



Graph based models

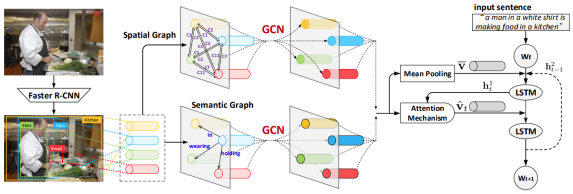
- Region based attention treats all regions equally without taking into account the interactions between them.
- Some studies consider using graphs based models for improved encoding of image regions by incorporating relations between different regions.



Graph based models

Spatial and semantic graphs

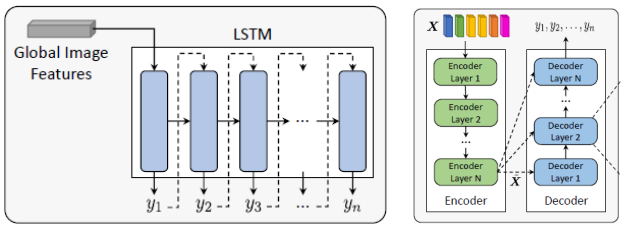
- A semantic relation classifier is trained on a large corpus and directly used for semantic graph generation.
- Spatial graphs are built and assigned depending on their Intersection over Union (IoU), relative distance and angle.



Yao T, Pan Y, Li Y, Mei T. Exploring visual relationship for image captioning. In Proceedings of the European conference on computer vision (ECCV) 2018 (pp. 684-699).

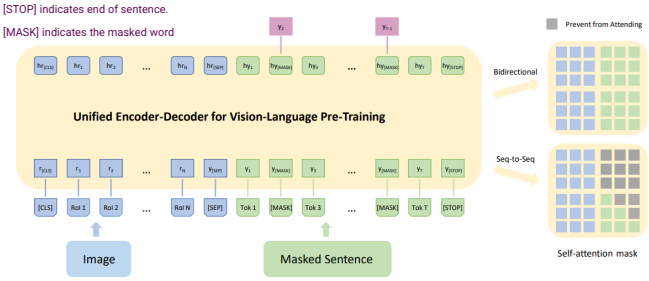
Language models

- The main language modeling strategies applied to image captioning are:
  1. LSTM based
  2. Transformer based fully attentive approaches
  3. Image-text early fusion (BERT-like)



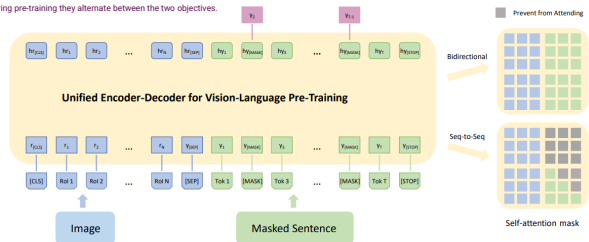
Vision-Language Transformer Model

- Unifies the transformer encoder and decoder into a single model.
- Model input consists of class-aware region embeddings, word embeddings and three special tokens [CLS], [SEP] and [STOP].
- [CLS] indicates the start of the visual input.
- [SEP] indicates the seperation betzeen visual and text input.
- [STOP] indicates end of sentence.
- [MASK] indicates the masked word



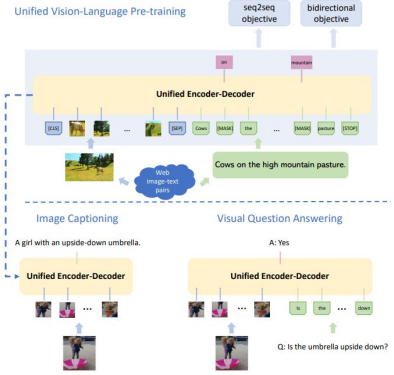
Vision-Language Transformer Model (pre-training)

- 15% of the text tokens are replaced by [MASK] token, random token or original token.
- The hidden state from the last transformer block is projected to word likelihoods where the masked token is predicted as classification problem.
- Through this reconstruction the model learns the dependencies in the context and forms a language model.
- Two objectives are considered within this model:
  - Bidirectional: every token can attend to every other token.
  - Seq-to-seq: tokens cannot attend to future tokens. It satisfies the auto-regressive property.
  - During pre-training they alternate between the two objectives.



VLP model for image-captioning

- For image captioning we fine-tune using seq2seq objective.
- During inference
  - Encode image region along with special tokens ([CLS] and [SEP] tokens).
  - We then start the generation by feeding in the [MASK] token and sampling a work from word likelihood output.
  - Replace the [MASK] token in the input sequence with sampled word and add new [MASK] token to the end of sequence.
  - Generation terminates when [STOP] token is chosen.



Training strategies

Cross-Entropy Loss

- Most used objective for image captioning.
- The aim is to minimize the negative log-likelihood of the current word given the previous ground-truth words.
- The loss works at word level and optimizes the probability of each word without considering long range dependencies.

$$L_{XE}(\theta) = -\sum_{i=1}^{n} \log\left(P\left(y_i \mid y_{1:i-1}, \boldsymbol{X}\right)\right)$$

Training strategies

Masked Language Model

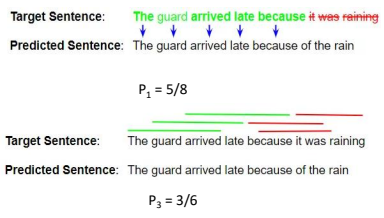
- Idea is to randomly mask a subset of input tokens and train the model to predict these masked tokens based on remaining tokens, both previous and subsequent.
- The model relies more on context making it more robust.
- Training on these models is much slower since they only train in masked tokens and not entire sentence.

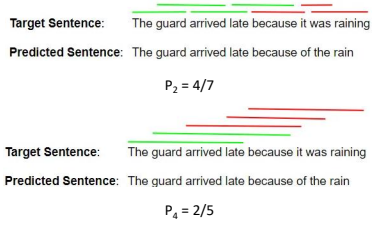
Evaluation

BLEU score

- Target sentence: The guard arrived late because it was raining
- Predicted sentence: The guard arrived late because of the rain

We first calculate precision scores for 1-gram through 4-grams.





Evaluation

BLEU score

- Brevity penalty: it penalizes sentences that are too short.

If the predicted sentence is just « the », the 1-gram precision is 1/1=1, indicating perfect score.

$$Brevity\ Penalty = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c <= r \end{cases}$$

c = predicted sentence lenght

r = target sentence lenght

$$Bleu\ (N) = Brevity\ Penalty \cdot Geometric\ Average\ Precision\ Scores\ (N)$$

$$Geometric\ Average\ Precision\ (N) = \ (p_1)^{\frac{1}{4}} \cdot (p_2)^{\frac{1}{4}} \cdot (p_3)^{\frac{1}{4}} \cdot (p_4)^{\frac{1}{4}}$$

Evaluation

ROUGE

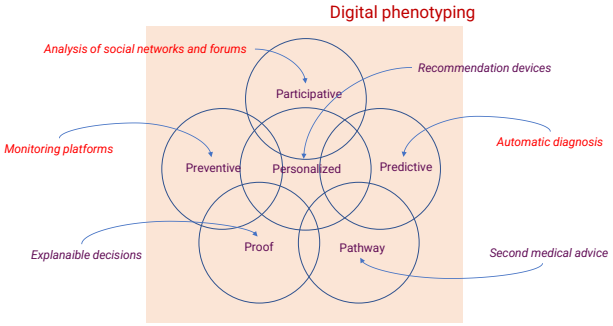
- Compared to BLEU score that focuses on precision, ROUGE focuses on recall.

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

6P Medicine

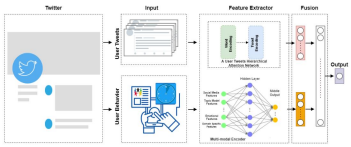
- 1P - Personalized: Personalized medicine consists of adapting a medical treatment according to the individual characteristics of a patient.
- 2P - Preventive: Preventive medicine focuses on wellness, and consists of measures taken for disease prevention.
- 3P - Predictive: Predictive medicine is a branch of medicine that aims to identify patients at risk of developing a disease.
- 4P - Participative: Medicine should be participatory, leading patients to be more responsible for their health and care.
- 5P - Proof: Medicine must be based on evidence of medical service to patients, especially when it relies on connected health and telemedicine.
- 6P - Pathway: Coordinating multiple interventions (medical, social, occupational medicine, etc.) such that the healthcare pathway is progressively articulated, according to the pathology and its evolution.

6P Medicine and AI



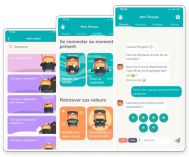
Social Network Analysis

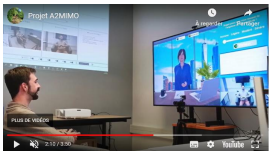
- Social networks are an important support for Participative medicine, which automatic analysis might allow Preventive/Predictive actions.
- It is common for people who suffer from mental health problems to often disclose their feelings and their daily struggles with mental health issues on social media as a way of relief.
- Twitter, Reddit, Doctissimo, to name but a few platforms have become an excellent resource to automatically discover people who are under depression.
- [Zogan et al., 2021] propose a depression detection framework by tackling textual, behavioral, temporal, and semantic modalities.



Monitoring Platforms and (Embodied) Chatbots

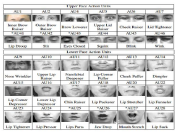
- A chatbot is a system that is able to converse and interact with human users using spoken, written, and visual languages (embodied).
- Chatbots can be useful preventive tools for individuals who are reluctant to seek mental health advice due to stigmatization.
- [Abd-alrazaq et al., 2019] studied 41 different embodied and non-embodied chatbots. Most tackle depression and autism.
- Among other scientific issues, therapeutic alliance is the key factor for the success of chatbots and ECAs.





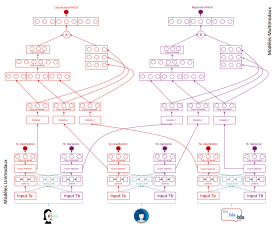
Multimodal Estimation of PHQ-8

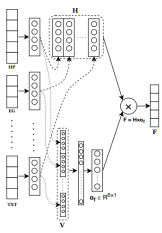
- In a patient-therapist interview, different signals shoud be combined for a correct diagnosis.
- Within the DAIC-WOZ dataset, the following signals are available:
  - Visual signals : expression of sadness, gaze escape, etc.
    - Facial Landmarks (FL), Head Pose (HP), Eye Gaze (EG), Action Unit (AU).
  - Speech signals : veiled voice, monotonous tone, etc.
    - Formant (FMT), COVAREP (COV).
  - Language signals : negative vocabulary, lack of perspective, etc.
    - Universal Sentence Encoder (TR).

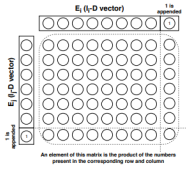


Multimodal Estimation of PHQ-8

- Combining classification and regression of depression estimators.
- An attention fusion network is used to combine inputs.
- Intra-modality inputs signals are combined with tensors.

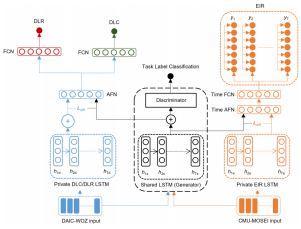


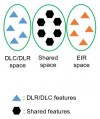




Emotional Language for the Estimation of PHQ-8

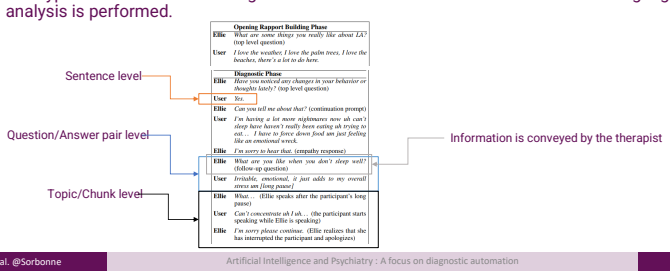
- In [Qureshi et al., 2020], we hypothesize that the estimation of depression level can benefit from the concurrent learning of emotion intensity.
- The CMU-MOSEI dataset comprises 3,228 videos from 1,000 different speakers over 250 topics. Videos were gathered from an online video platform, where users emit their opinions in the form of monologues.





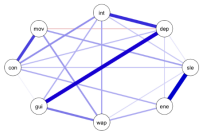
Analysis of Structured Interviews

- First observation: most of the related works have been dealing with the interview on a line by line basis; the hypothesis being that sentence representation is the correct one.
- Second observation: some of the related works only deal with the patient information; the hypothesis being that only the patient information is important for the diagnosis.
- Our hypothesis is that better diagnosis can be established if the correct level of language analysis is performed.



Symptom-based Analysis

- Most related works have been tackling depression level estimation as a simple task (depressed or non depressed). More advanced models have been trying to predict the PHQ-8 score (between 0 and 24) directly or propose to solve the intermediate 5-class problem (none-minimal, mild, moderate, moderately severe, severe depression).
- In Psychiatry, there is a shift towards richer representations of psychiatric syndromes that can take into account the dimensional and heterogeneous nature of the clinical pictures of the same psychiatric diagnosis. One particular approach that is gaining attention concerns symptom network analysis.
- We develop similar models as previously to acknowledge if they can handle the prediction of individual symptom values, where each of the 8 symptoms is a value between 0 and 3.



Symptom-based Analysis

- In order to better understand results, we present a radar plot analysis that shows that adequate behavior of the model is obtained.