Attention is all you need!

Sommaire

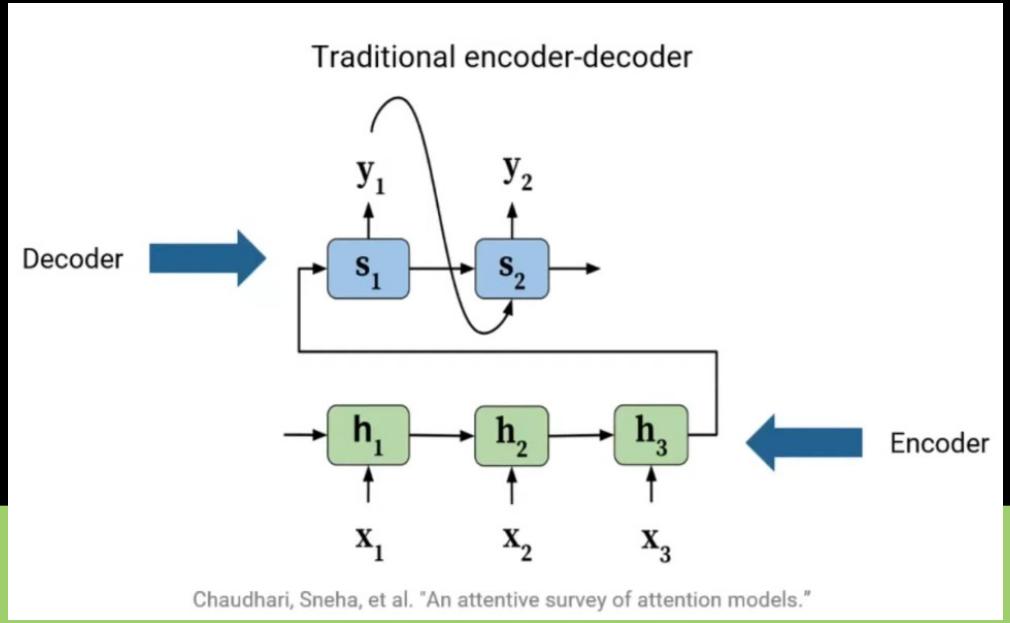
- Introduction
- Détails du modèle
  - Encoder
  - Decoder
- Résultats
- Conclusion

Introduction

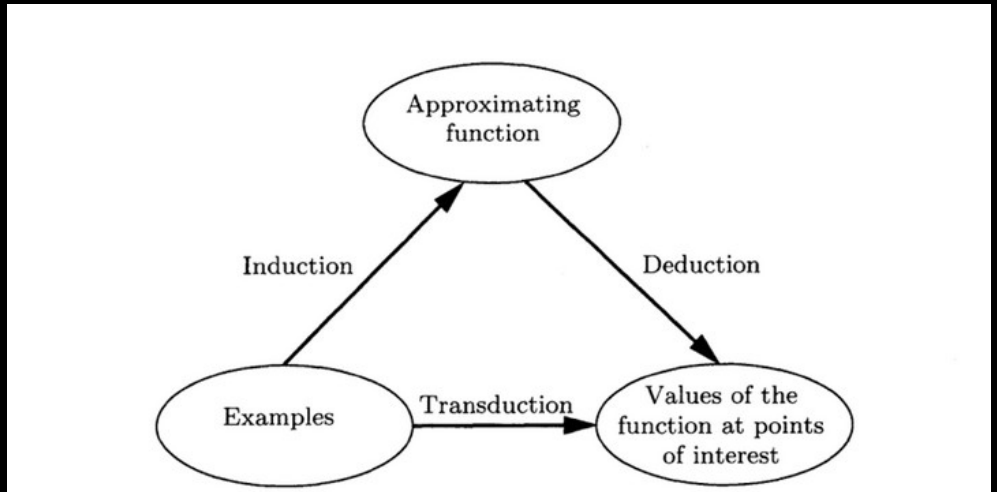
Introduction

Transduction et modélisation de séquences :

- Modélisation du langage
- Traduction
- RNNs, LSTM, gated RNNs

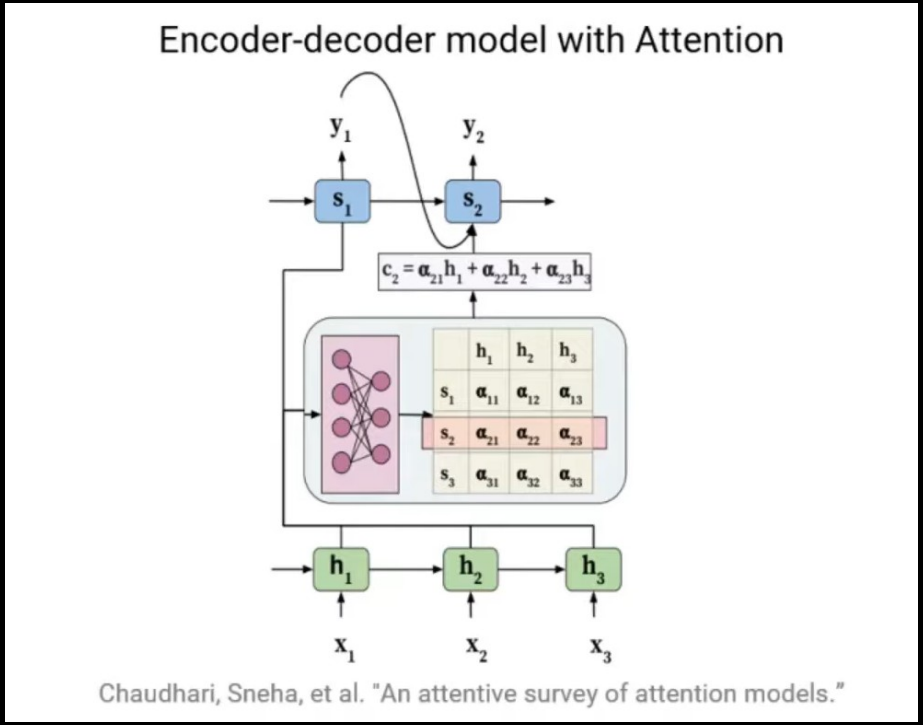


Chaudhari, Sneha, et al. "An attentive survey of attention models."



Relationship between Induction, Deduction and Transduction
Taken from The Nature of Statistical Learning Theory.

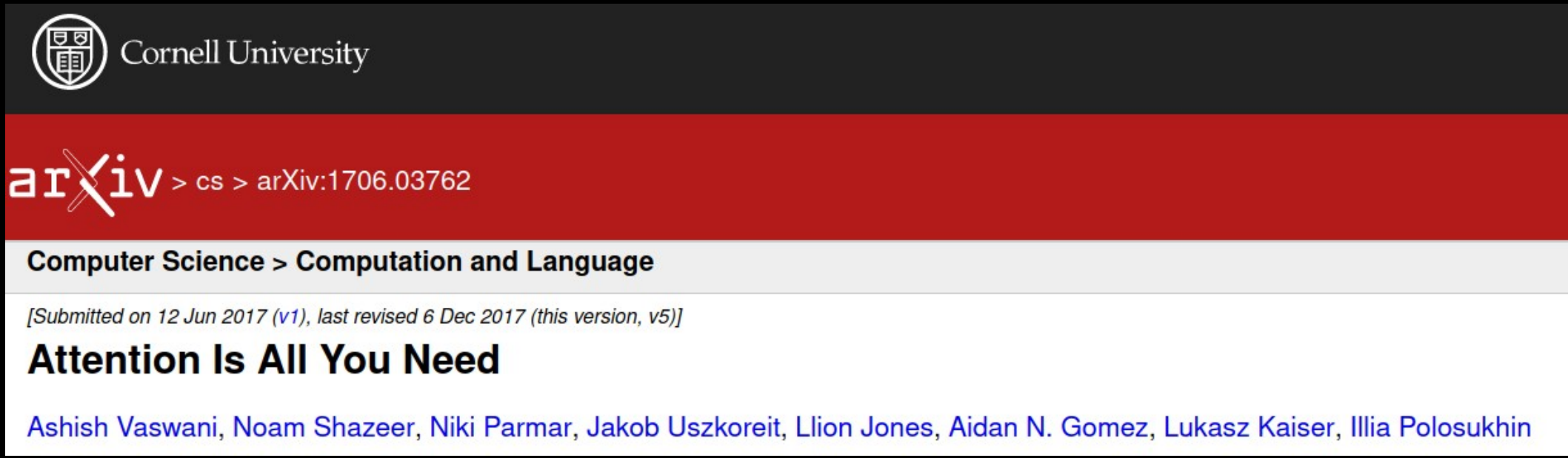
Introduction



Chaudhari, Sneha, et al. "An attentive survey of attention models."

- 2014 : Neural Machine Translation by Jointly Learning to Align and Translate" by Bahdanau et al.
- Affinités entre éléments distants
- Mais RNNs trop lents (GPU...)

Introduction

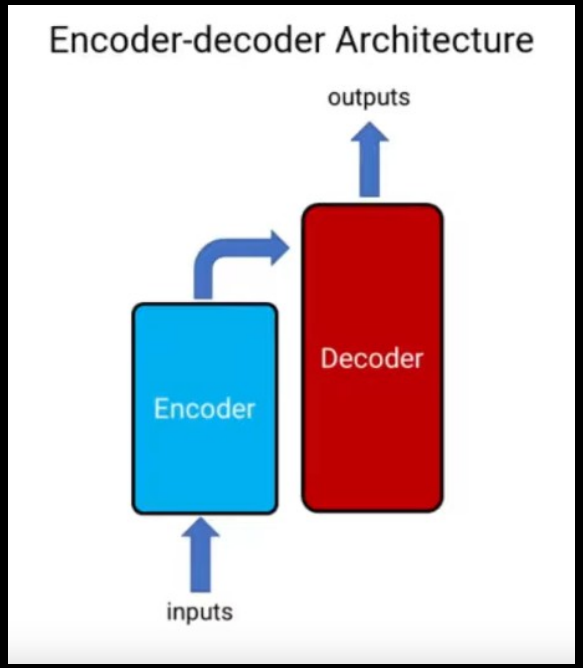


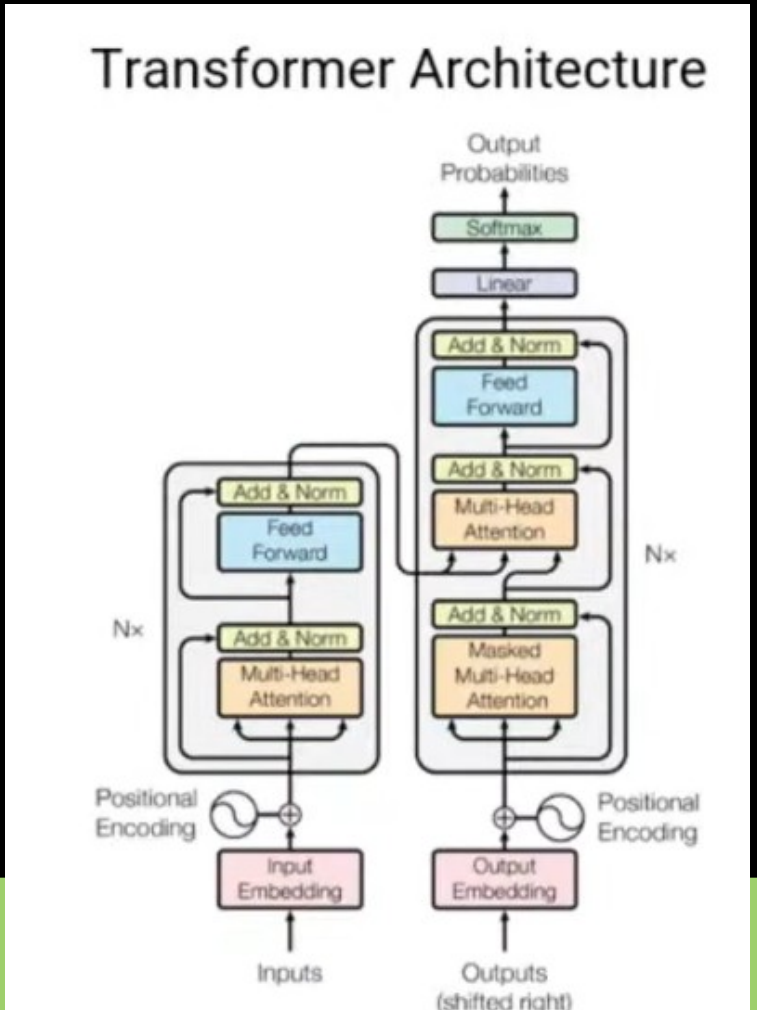
2017 : Introduction d’une nouvelle architecture : les transformers.
Pas de traitement séquentiel --> //

Détails du modèle Encoder & Decoder

Détails de l’encoder Preprocessing...

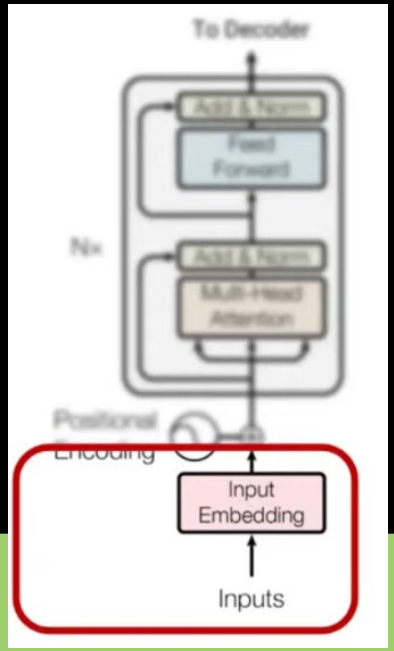
Détails de l’encoder : preprocessing



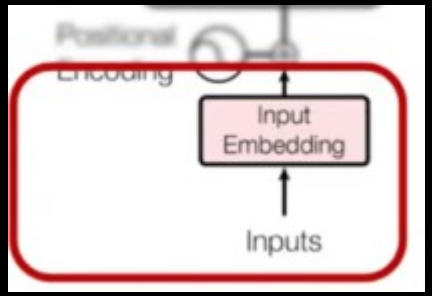


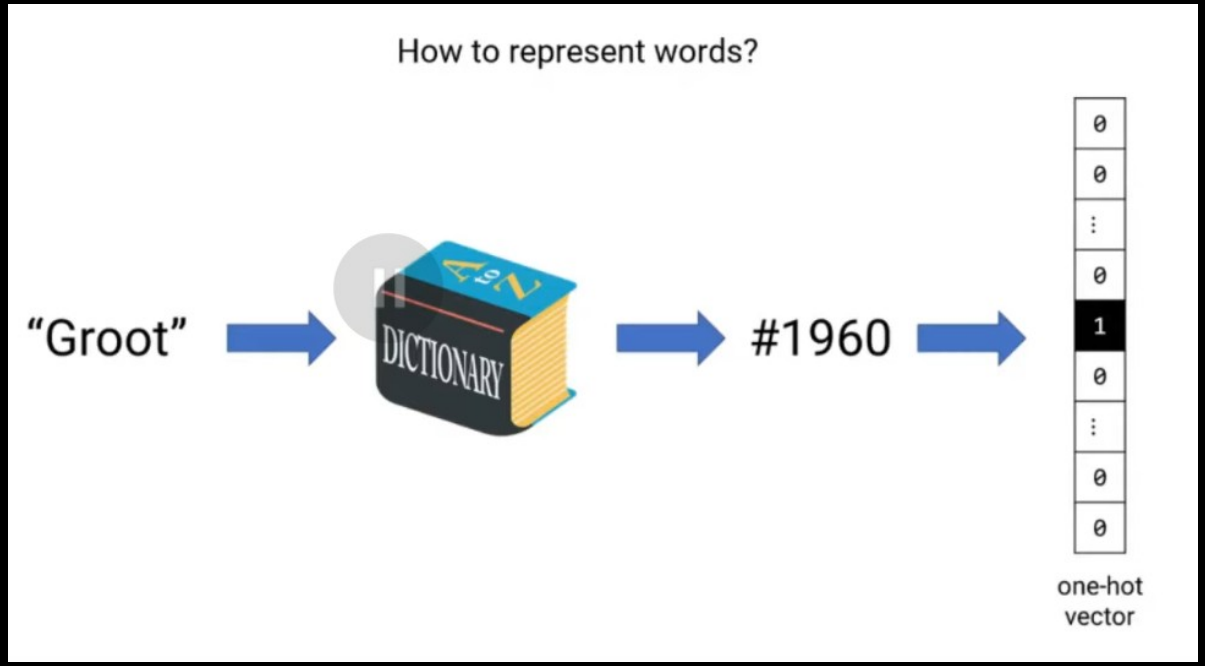
Inputs = séquence de mots

Comment encoder ces mots?



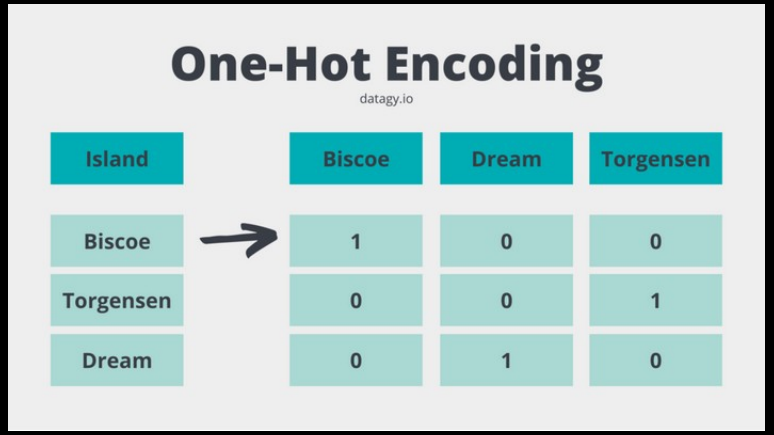
Détails de l’encoder : preprocessing



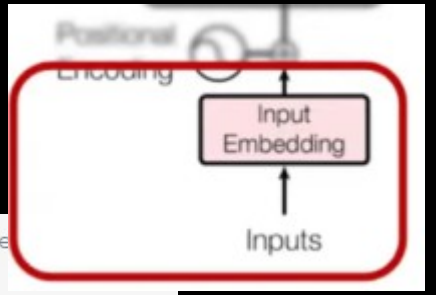


Première idée : one-hot vector

-- > One hot encoder



Détails de l'encoder : preprocessing




```
# Loading a Sample DataFrame
import pandas as pd

df = pd.DataFrame({
    'Name': ['Joan', 'Matt', 'Jeff', 'Melissa', 'Devi'],
    'Gender': ['Female', 'Male', 'Male', 'Female', 'Female'],
    'House Type': ['Apartment', 'Detached', 'Apartment', None,
'Semi-Detached']
    })

print(df)

# Returns:
#       Name  Gender     House Type
# 0     Joan  Female      Apartment
# 1     Matt    Male       Detached
# 2     Jeff    Male      Apartment
# 3  Melissa  Female           None
# 4     Devi  Female  Semi-Detached
```

```
# Understanding the Pandas get_dummie
import pandas as pd
pd.get_dummies(
    data,
    prefix=None,
    prefix_sep='_',
    dummy_na=False,
    columns=None,
    sp
    dt
    dt
)
```

`data=` represents the data from which to get the dummy indicators (either array-like, Pandas Series, or Pandas DataFrame)

`prefix=` represents the string to append to DataFrame column names

`prefix_sep=` represents what delimiter to use

`dummy_na=` represents whether to add a column or not for missing values

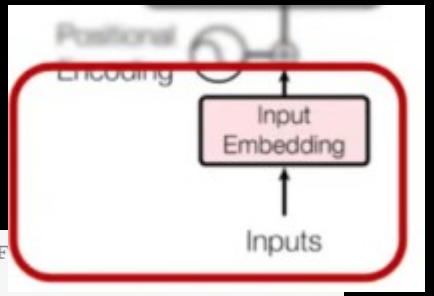
`columns=` represents the names of the columns to be encoded

`sparse=` represents whether the data should be a sparse array or a regular NumPy array

`drop_first=` represents whether to drop the first level or not

`dtype=` represents the data type for new columns

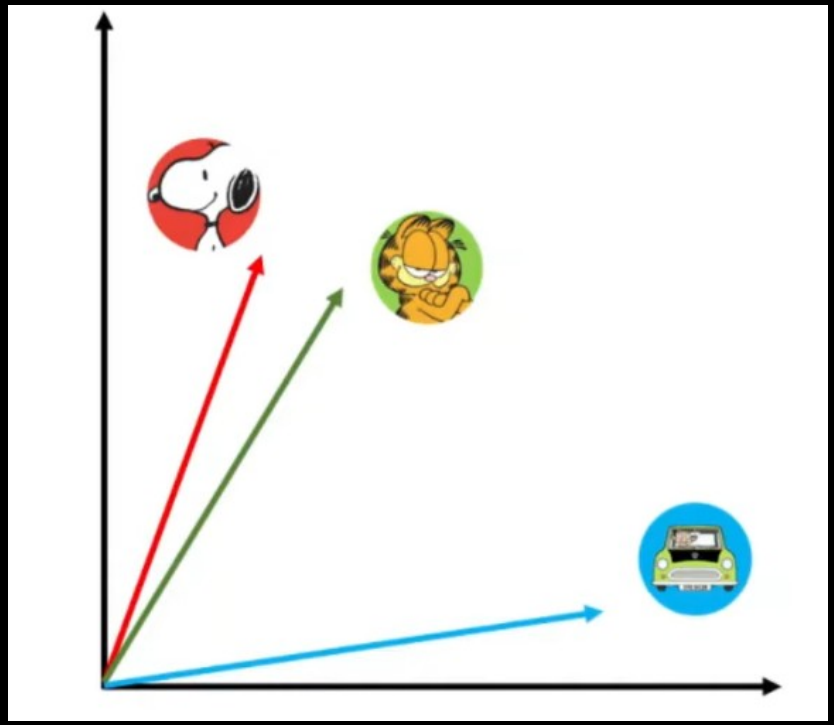
Détails de l'encoder : preprocessing

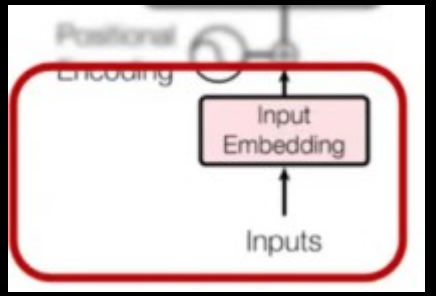



```
# One-Hot Encoding a Single DataFrame Series
import pandas as pd

df = pd.DataFrame({
    'Name': ['Joan', 'Matt', 'Jeff', 'Melissa', 'Devi'],
    'Gender': ['Female', 'Male', 'Male', 'Female', 'Female'],
    'House Type': ['Apartment', 'Detached', 'Apartment', None,
'Semi-Detached']
    })

print(pd.get_dummies(df['Gender']))

# Returns:
#    Female  Male
# 0       1     0
# 1       0     1
# 2       0     1
# 3       1     0
# 4       1     0
```

```
# One-Hot Encoding and Returning a DataF
import pandas as pd

df = pd.DataFrame({
    'Name': ['Joan', 'Matt', 'Jeff', 'Melissa', 'Devi'],
    'Gender': ['Female', 'Male', 'Male', 'Female', 'Female'],
    'House Type': ['Apartment', 'Detached', 'Apartment', None,
'Semi-Detached']
    })

ohe = pd.get_dummies(data=df, columns=['Gender'])
print(ohe)

# Returns:
#       Name     House Type  Gender_Female  Gender_Male
# 0     Joan      Apartment              1            0
# 1     Matt       Detached              0            1
# 2     Jeff      Apartment              0            1
# 3  Melissa           None              1            0
# 4     Devi  Semi-Detached              1            0
```

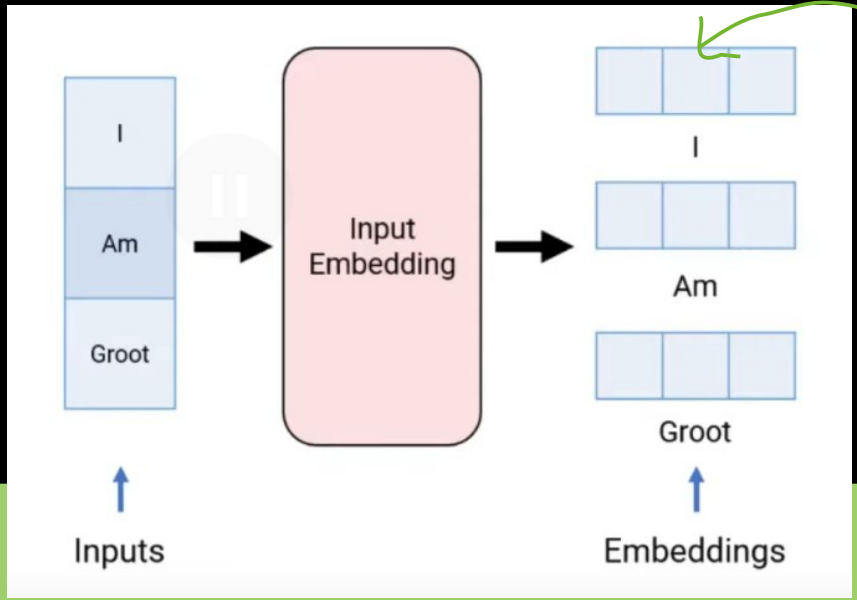
Détails de l’encoder : preprocessing



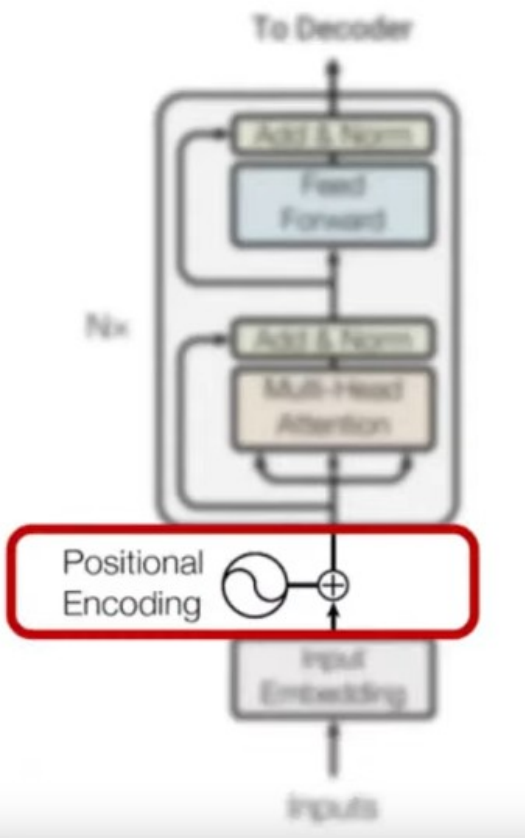


Meilleure idée : word embeddings

Mots similaires ↔
représentations similaires

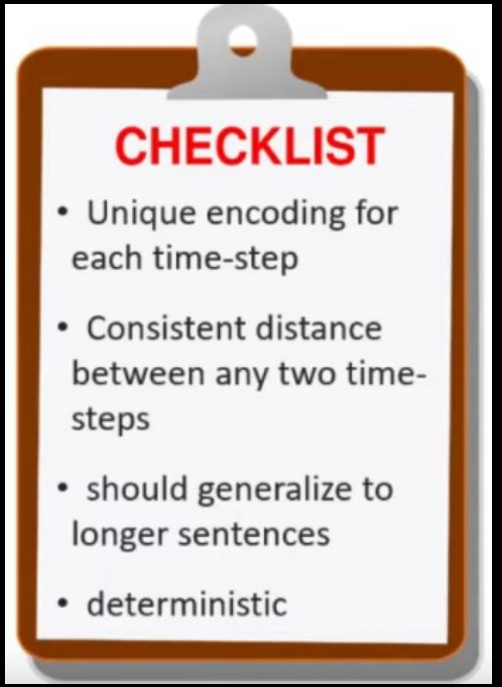


Détails de l’encoder : preprocessing



Positional encoding = rajouter de l’info sur les positions relatives des mots

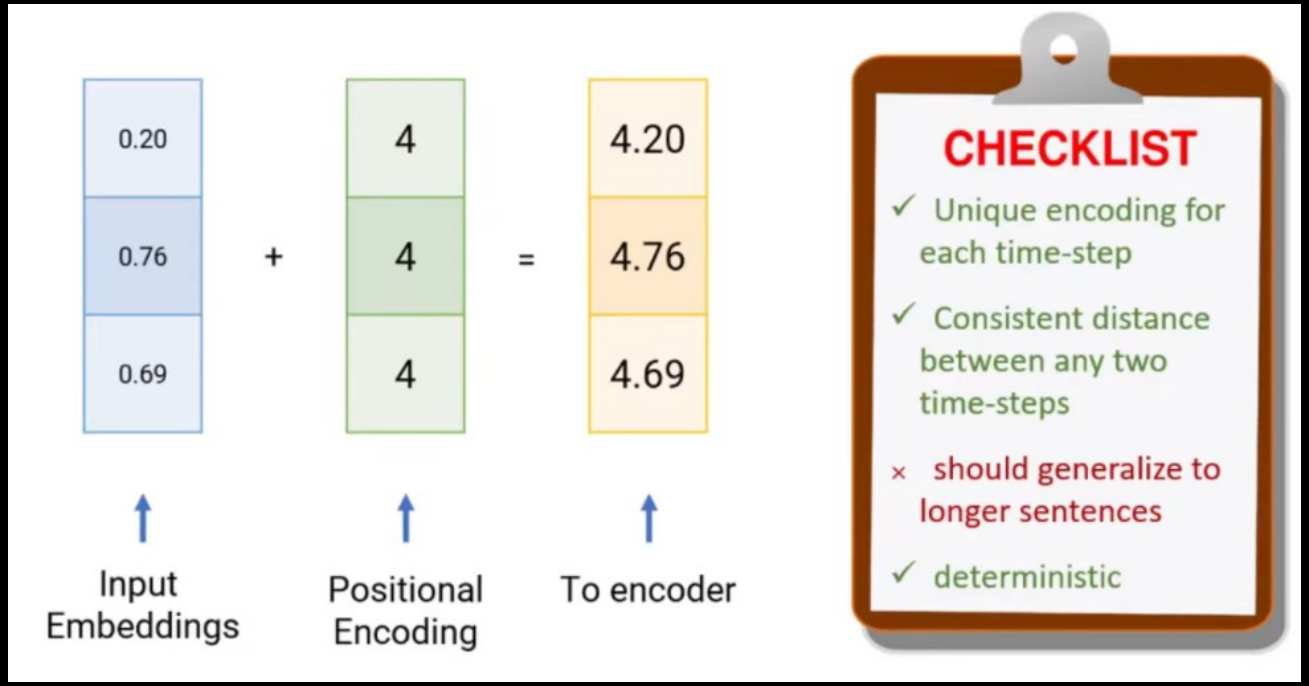
-- > Faire la somme d’un nouveau vecteur (positions relatives) et des vecteurs précédents (mots)



The Avengers defeated Thanos

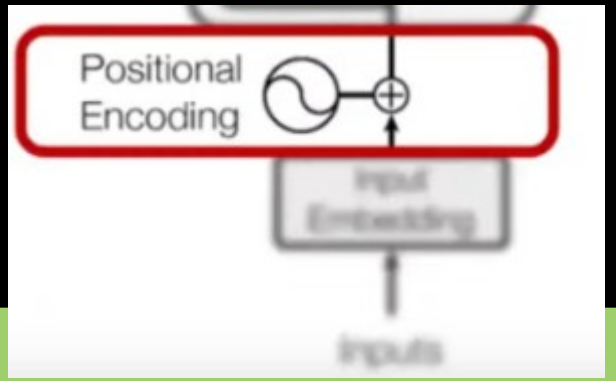
Thanos defeated The Avengers

Détails de l’encoder : preprocessing

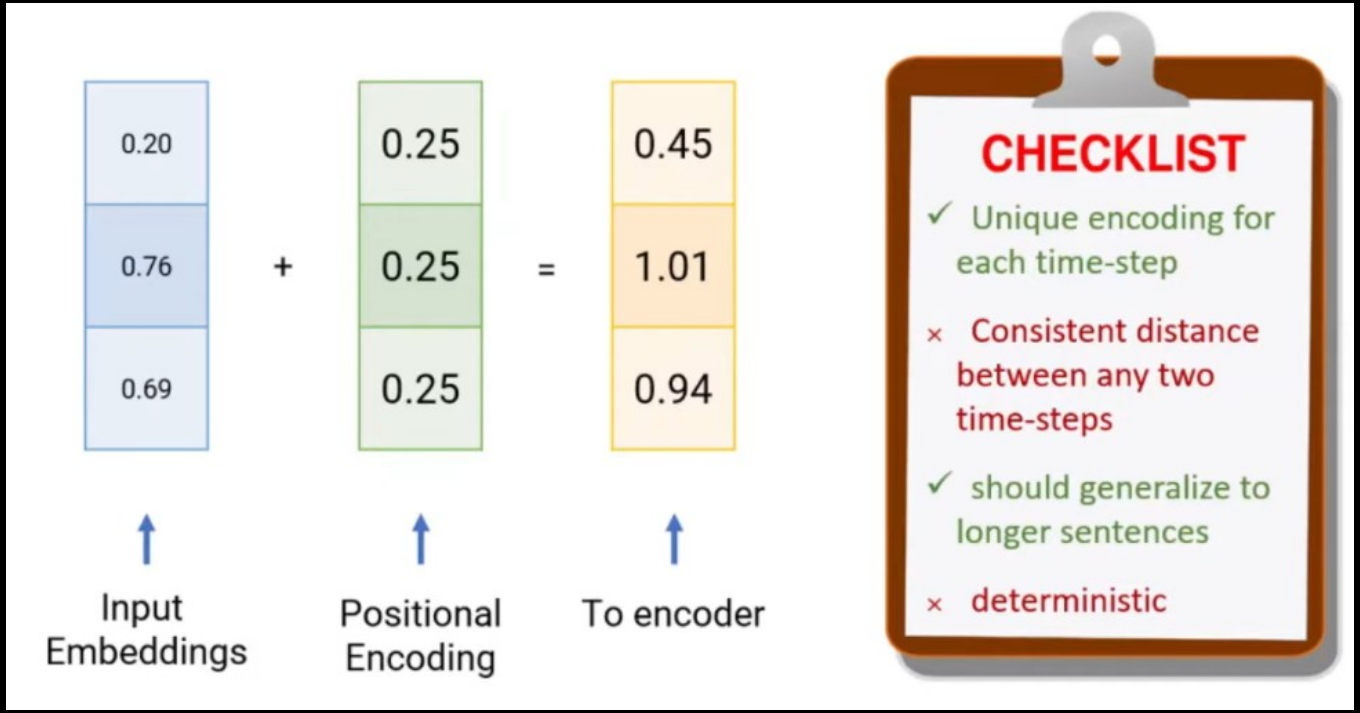


Rajouter 1 à la somme des vecteurs précédents pour chaque nouveau mot…

>>

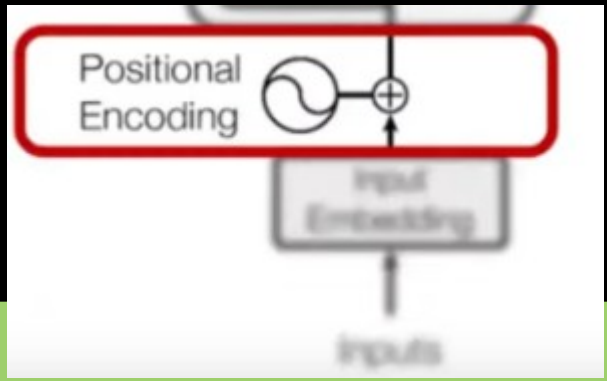


Détails de l’encoder : preprocessing

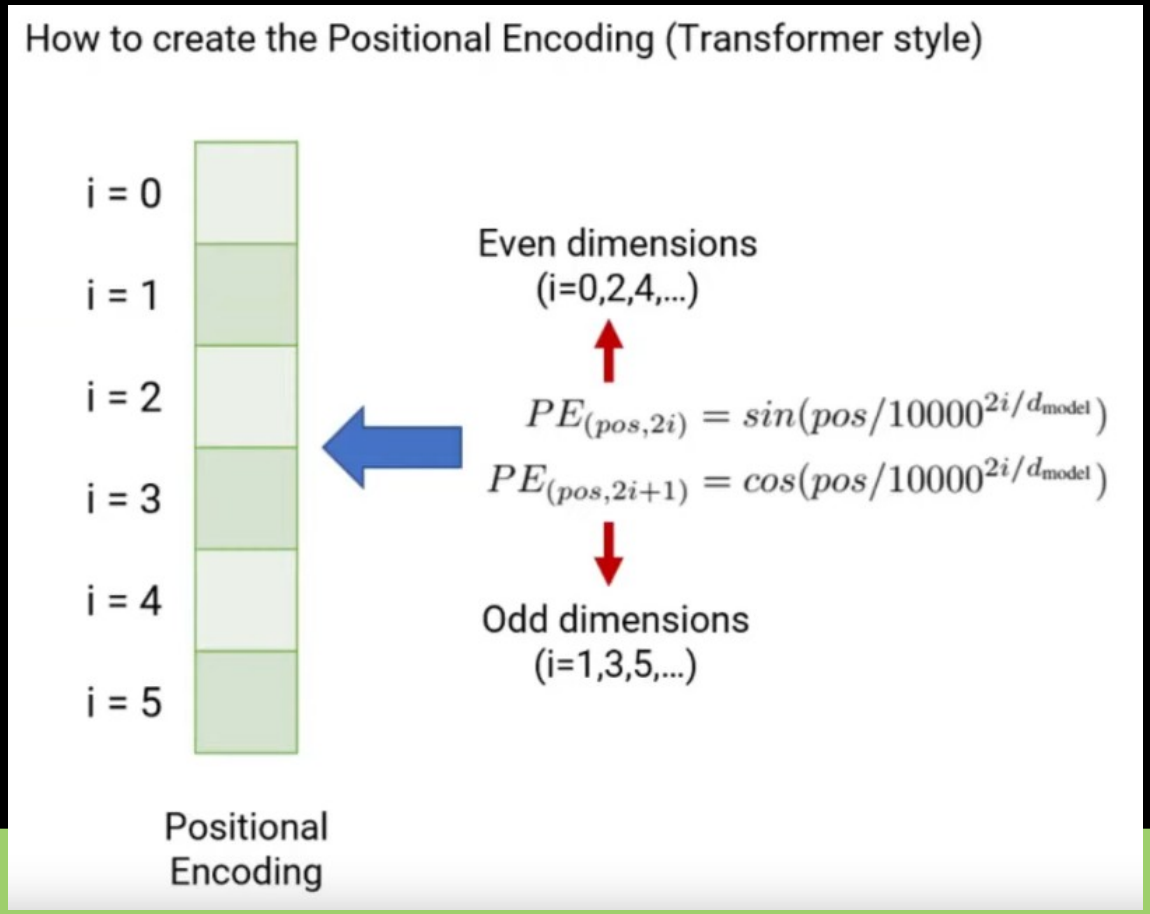


Rajouter un nombre entre 0 et 1

!deterministe <- plusieurs mots / delta T



Détails de l’encoder : preprocessing

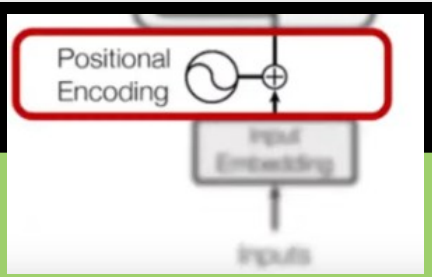


Vecteur de 512 lignes

$d_{model}=512$

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \vdots \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1} \qquad (\omega_k = \frac{1}{1000^{\frac{2k}{d}}})$$

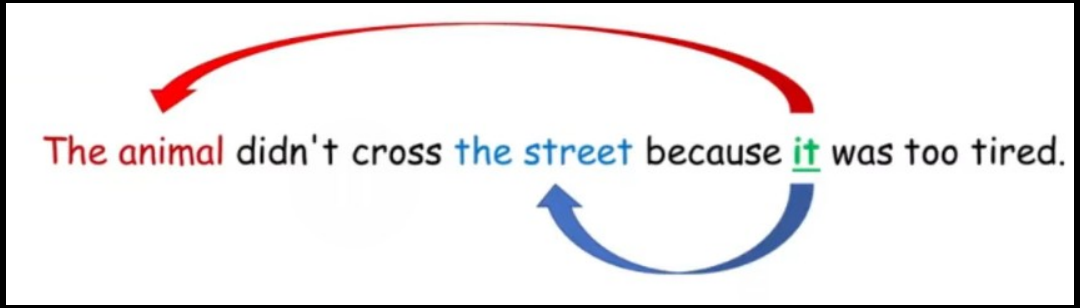
https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

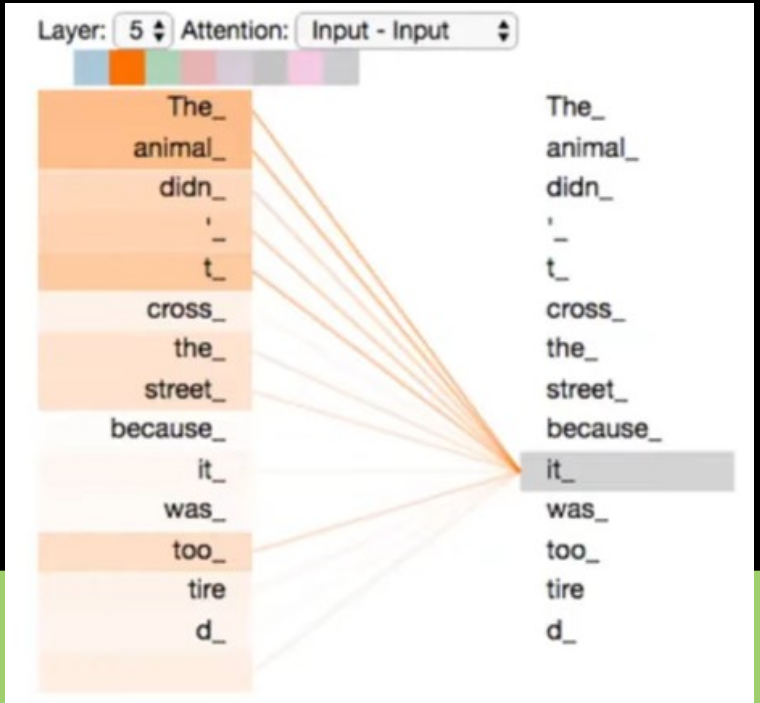


Fin du preprocessing de l’encoder!

Détails de l’encoder : Multi-head self-attention mechanism

Détails de l’encoder : self attention





Détails de l’encoder : self attention