KSBi-BIML 2024

Bioinformatics & Machine Learning(BIML) Workshop for Life and Medical Scientists

생명정보학 & 머신러닝 워크샵 (오프라인)

Introduction to Transformers

전민지 _ 고려대학교

KSBI KOREAN SOCIETY FOR BIOINFORMATICS | 한국생명정보학회

본 강의 자료는 한국생명정보학회가 주관하는 BIML 2024 워크샵 오프라인 수업을 목적으로 제작된 것으로 해당 목적 이외의 다른 용도로 사용할 수 없음을 분명하게 알립니다.

이를 다른 사람과 공유하거나 복제, 배포, 전송할 수 없으며 만약 이러한 사항을 위반할 경우 발생하는 **모든 법적 책임은 전적으로 불법 행위자 본인에게 있음을 경고**합니다.

KSBi-BIML 2024

Bioinformatics & Machine Learning(BIML)
Workshop for Life and Medical Scientists

안녕하십니까?

한국생명정보학회가 개최하는 동계 교육 워크샵인 BIML-2024에 여러분을 초대합니다. 생명정보학 분야의 연구자들에게 최신 동향의 데이터 분석기술을 이론과 실습을 겸비해 전달하고자 도입한 전문 교육 프로그램인 BIML 워크샵은 2015년에 시작하여 올해로 벌써 10년 차를 맞이하게 되었습니다. BIML 워크샵은 국내 생명정보학 분야의 최초이자 최고 수준의 교육프로그램으로 크게 인공지능과 생명정보분석 두 개의 분야로 구성되어 있습니다. 올해 인공지능 분야에서는 최근 생명정보 분석에서도 응용이 확대되고 있는 다양한 인공지능 기반 자료모델링 기법들에 대한 현장 강의가 진행될 예정이며, 관련하여 심층학습을 이용한 단백질구조예측, 유전체분석, 신약개발에 대한 이론과 실습 강의가 함께 제공될 예정입니다. 또한 단일세포오믹스, 공간오믹스, 메타오믹스, 그리고 롱리드염기서열 자료 분석에 대한 현장 강의는 많은 연구자의 연구 수월성 확보에 큰 도움을 줄 것으로 기대하고 있습니다.

올해 BIML의 가장 큰 변화는 최근 연구 수요가 급증하고 있는 의료정보자료 분석에 대한 현장 강의를 추가하였다는 것입니다. 특히 의료정보자료 분석을 많이 수행하시는 의과학자 및 의료정보 연구자들께서 본 강좌를 통해 많은 도움을 받으실 수 있기를 기대하고 있습니다. 또한 다양한 생명정보학 분야에 대한 온라인 강좌 프로그램도 점차 증가하고 있는 생명정보 분석기술의 다양화에 발맞추기 위해 작년과 비교해 5강좌 이상을 신규로 추가했습니다. 올해는 무료 강좌 5개를 포함하여 35개 이상의 온라인 강좌가 개설되어 제공되며, 연구 주제에 따른 연관된 강좌 추천 및 강연료 할인 프로그램도 제공되며, 온라인을 통한 Q&A 세션도 마련될 예정입니다. BIML-2024는 국내 주요 연구중심 대학의 전임 교원이자 각 분야 최고 전문가들의 강의로 구성되었기에 해당 분야의 기초부터 최신 연구 동향까지 포함하는 수준 높은 내용의 강의가 될 것이라 확신합니다.

BIML-2024을 준비하기까지 너무나 많은 수고를 해주신 운영위원회의 정성원, 우현구, 백대현, 김태민, 김준일, 김상우, 장혜식, 박종은 교수님과 KOBIC 이병욱 박사님께 커다란 감사를 드립니다. 마지막으로 부족한 시간에도 불구하고 강의 부탁을 흔쾌히 허락하시고 훌륭한 현장 강의와 온라인 강의를 준비하시는데 노고를 아끼지 않으신 모든 강사분들께 깊은 감사를 드립니다.

2024년 2월

한국생명정보학회장 이 인 석

강의 시간표

DAY1 : 2월 24일 (토)

시간	강 의 (자연과학대학 28동 101호)
12:30-12:50	등록
12:50-13:00	공지사항 전달
13:00-14:30	**의료빅데이터/인공지능 총론** 김헌성 교수(가톨릭대학교)
14:30-14:45	휴식
14:45-16:15	**의료영상 인공지능의 이해 및 의료영상 레이블링 실습** 백서연 교수(연석대학교)
16:15-16:30	휴식
16:30-18:00	**의료 정보처리 자동화 실습 / 독자적인 어플리케이션 만들기** 김선근 대표(원닥 주식회사), 서사도 조교

시간	강 의 (자연과학대학 28동 102호)
12:30-12:50	등록
12:50-13:00	공지사항 전달
13:00-14:20	**EMR 데이터를 활용한 머신러닝 기반 예후예측:** **Decision Tree-based Models + EMR 샘플 데이터 실습 (MIMIC sample dataset)** 고태훈 교수(가톨릭대학교)
14:20-14:40	휴식
14:40-16:00	**Chest X-ray 영상을 활용한 딥러닝 기반 폐질환 진단:** **Convolutional Neural Network + 의료영상 샘플 데이터 실습 (NIH Chest X-ray14)** 고태훈 교수(가톨릭대학교)
16:00-16:20	휴식
16:20-17:40	**심전도 데이터를 활용한 딥러닝 기반 부정맥 탐지: Recurrent Neural Network +** **Transformer + 심전도 샘플 데이터 실습 (MIT-BIH Arrhythmia Database)** 고태훈 교수(가톨릭대학교)

DAY1 : 2월 26일 (월)

시간	강 의 (자연과학대학 28동 101호)
09:00-09:20	등록
09:20-09:30	공지사항 전달
09:30-10:50	**DNN (이론)** 이상근 교수(고려대학교)
10:50-11:00	휴식
11:00-12:10	**CNN (이론)** 이상근 교수(고려대학교)
12:10-13:40	점심
13:40-15:10	**RNN, ChatGPT, XAI (이론)** 이상근 교수(고려대학교)
15:10-15:20	휴식
15:20-16:50	**CNN/RNN 모델 구조 정의, 학습 알고리즘 적용, 성능 평가, 시각화 방법 (Tensorflow 실습)** 이정현 조교, 한성민 조교

시간	강 의 (자연과학대학 28동 102호)
09:00-09:20	등록
09:20-09:30	공지사항 전달
09:30-11:00	**Best practice for single-cell data analysis** 박종은 교수(KAIST)
11:00-11:10	휴식
11:10-12:40	**Practice1: Scanpy basic workflow** 정성민 조교, 고용준 조교
12:40-14:10	점심
14:10-15:30	**Public database, data integration, reference mapping, multiomics** 박종은 교수(KAIST)
15:30-15:40	휴식
15:40-16:50	**Practice2: Advanced single-cell analysis (siVI universe)** 정성민 조교, 고용준 조교

DAY1 : 2월 27일 (화)

시간	강 의 (자연과학대학 28동 101호)
09:00-09:20	등록
09:20-09:30	공지사항 전달
09:30-10:50	**AI-based protein structure prediction** **- Intro to protein structure prediction** **- Early AI-based approaches** **- AlphaFold and RoseTTAFold** 백민경 교수(서울대학교)
10:50-11:00	휴식
11:00-12:10	**단백질 구조 예측 실습** **- ColabFold를 활용한 단백질 구조 및 상호작용 예측** **- Tips &Tricks for better structure modeling** 백민경 교수(서울대학교)
12:10-13:40	점심
13:40-15:10	**AI-based protein design** **- Intro to protein design** **- Protein backbone design using RFdiffusion** **- Protein sequence design using ProteinMPNN** 백민경 교수(서울대학교)
15:10-15:20	휴식
15:20-16:50	**단백질 디자인 실습** **- RFdiffusion 및 ProteinMPNN의 활용법 실습** 백민경 교수(서울대학교)

시간	강 의 (자연과학대학 28동 102호)
09:00-09:20	등록
09:20-09:30	공지사항 전달
09:30-11:00	**Introduction to Single-cell biology** 최정민 교수(고려대학교)
11:00-11:10	휴식
11:10-12:40	**i. Unsupervised Spatial transcriptome analysis** **ii. Tumor Boundary Determination in Spatial Transcriptomics** 유광민 조교, 이문영 조교
12:40-14:10	점심
14:10-15:30	**i. Deconvolution Analysis Using Single-cell RNA Sequencing and Spatial Transcriptomics** **ii. Cell-Cell Interaction Analysis in Spatial Transcriptomics** 김지현 조교, 최승지 조교
15:30-15:40	휴식
15:40-16:50	**i. Open Chromatin Region Analysis and Biological Interpretation of Using scATAC-seq Dataset** **ii. Construction of Gene Regulatory Networks Based on Integrated Analysis of scATAC-seq and scRNA-seq Datasets** 천하림 조교, 이호진 조교

DAY1 : 2월 28일 (수)

시간	강 의 (자연과학대학 28동 101호)
09:00-09:20	등록
09:20-09:30	공지사항 전달
09:30-11:00	**Introduction to Transformers (이론)** 전민지 교수 (고려대학교)
11:00-11:10	휴식
11:10-12:40	**Introduction to Transformers (실습)** 봉현수 조교, 임우택 조교
12:40-14:10	점심
14:10-15:40	**Deep learning in Bioinformatics** 노미나 교수(한양대학교)
15:40-15:50	휴식
15:50-17:20	**Deep learning model을 이용한 실습** 박예솔 조교

시간	강 의 (자연과학대학 28동 102호)
09:00-09:20	등록
09:20-09:30	공지사항 전달
09:30-10:50	**마이크로바이옴 기본 이론** 이선재 교수(GIST)
10:50-11:00	휴식
11:00-12:10	**16S rRNA amplicon seq. - DADA2** 조준우 조교, 백재우 조교
12:10-13:40	점심
13:40-14:40	**최신 메타지놈 분석 기법의 현황** 이선재 교수(GIST)
14:40-14:50	휴식
14:50-16:50	**Shotgun metagenome 분석 (Linux)** 조준우 조교, 백재우 조교

DAY1 : 2월 29일 (목)

시간	강 의 (자연과학대학 28동 101호)
09:00-09:20	등록
09:20-09:30	공지사항 전달
09:30-10:50	**화학정보학 기초(Cheminformatics) / 약물특성 및 약물다움(druglikeness)** **Molecular Notations &Descriptors / AI 신약개발을 위한 Databases** **AI 신약개발을 위한 Programming 기초** 김동섭 교수(KAIST)
10:50-11:00	휴식
11:00-12:10	**Google Colab에 RDKit 설치 / 화합물 정보 읽기 실습** **Bioactivity database 검색 및 정보 읽기 실습** **Molecular descriptor (fingerprint) 생성 및 similarity 계산 실습** 정수재 조교, 나민주 조교
12:10-13:40	점심
13:40-15:10	**AI 신약개발을 위한 기계학습법 기초 / QSAR 모델링 기초 / AI 신약개발을 위한 딥러닝 모델** **Virtual screening (ligand-based, structure-based) 및 de novo design** 김동섭 교수(KAIST)
15:10-15:20	휴식
15:20-16:50	**QSAR modeling 전체 과정 실습/ 화합물의 Bioactivity 예측 모델 개발** **Virtual screening 과정을 통한 신약후보물질 발굴 실습** 정수재 조교, 나민주 조교

시간	강 의 (자연과학대학 28동 102호)
09:00-09:20	등록
09:20-09:30	공지사항 전달
09:30-11:00	**Single cell multiomics 이론 / Gene regulatory network 이론** 김준일 교수(숭실대학교)
11:00-11:10	휴식
11:10-12:40	**Seurat/Signac, ArchR, TENET+ 실습** 김현규 조교, 정회빈 조교
12:40-14:10	점심
14:10-15:40	**롱리드 시퀀싱 소개 및 유전체 조립 실습** 김준 교수(충남대학교)
15:40-15:50	휴식
15:50-17:20	**변이 분석 및 시각화 실습** 김준 교수(충남대학교)

강의개요

Introduction to Transformers

Transformer 모델은 2017년에 소개된 이후로 자연어 처리를 비롯한 다양한 분야에서 혁신적인 변화를 가져왔다. Transformer의 핵심인 '어텐션 메커니즘'은 모델이 입력 데이터의 다양한 부분에 '주목'하게 함으로써, 이전의 순차적인 처리 방식을 사용하는 모델들이 가지고 있던 한계를 극복했다. 이로 인해 더 긴 시퀀스를 효과적으로 처리할 수 있게 되었고, 따라서 단백질 시퀀스, 약물 구조 등 생명의료 분야에서도 그 활용성이 높다.

본 강의에서는 Transformer 모델의 구조를 이해하고, 생명의료 분야에서의 응용 사례를 살펴본다. 또한 실습을 통해 직접 사전 학습된 Transformer 기반 모델을 활용하는 방법을 학습하는 것을 목표로 한다.

강의는 다음의 내용을 포함한다:

- Transformer 개요
- Transformer 응용
- Transformer 실습

* 교육생준비물: 노트북 (메모리 8GB 이상, 디스크 여유공간 30GB 이상)

* 선수과목: Python

* 강의 난이도: 중급

* 강의: 전민지 교수 (고려대학교 의과대학),
봉현수 조교 (명지대학교 융합소프트웨어학과)
임우택 조교 (고려대학교 의과대학)

Curriculum Vitae

Speaker Name: Minji Jeon, Ph.D.

▶ **Personal Info**

Name	Minji Jeon
Title	Assistant Professor
Affiliation	Korea University

▶ **Contact Information**

Address	161, Jeongneung-ro, Seongbuk-gu, Seoul, 02708
Email	mjjeon@korea.ac.kr
Phone Number	010-2354-7084

Research Interest

AI-driven drug discovery, machine learning, bioinformatics

Educational Experience

2012	B.S. in Computer Science, Korea University, Korea
2014	M.S. in Interdisciplinary Graduate Program in Bioinformatics, Korea University, Korea
2018	Ph.D. in Computer Science, Korea University, Korea

Professional Experience

2018-2019	Research Professor, Korea University, Korea
2020-2022	Postdoctoral Fellow, Icahn School of Medicine at Mount Sinai, USA
2022-	Assistant Professor, Korea University, Korea

Selected Publications (5 maximum)

1. Zhaoping Xiong†, **Minji Jeon†**, Robert J Allaway†, Jaewoo Kang, Donghyeon Park, Jinhyuk Lee, Hwisang Jeon, Miyoung Ko, Hualiang Jiang, Minyue Zheng, Aik Choon Tan, Xindi Guo, The Multi-Targeting Drug DREAM Challenge Community, Kristen K Dang, Alex Tropsha, Chana Hecht, Tirtha K. Das, Heather A. Carlson, Ruben Abagyan, Justin Guinney, Avner Schlessinger*, Ross Cagan* "Crowdsourced identification of multi-target kinase inhibitors for RET- and TAU-based disease: the Multi-Targeting Drug DREAM Challenge" PLoS computational biology 17.9 (2021): e1009302.
2. **Minji Jeon†**, Kathleen M. Jagodnik†, Eryk Kropiwnicki, Daniel J. Stein, Avi Ma'ayan* "Prioritizing Pain-Associated Targets with Machine Learning" Biochemistry 60.18 (2021): 1430-1446.
3. **Minji Jeon**, Donghyeon Park, Jinhyuk Lee, Hwisang Jeon, Miyoung Ko, Sunkyu Kim, Yonghwa Choi, Aik-Choon Tan, Jaewoo Kang* "ReSimNet: Drug Response Similarity Prediction using Siamewe Neural Networks" Bioinformatics 35.24 (2019): 5249-5256.

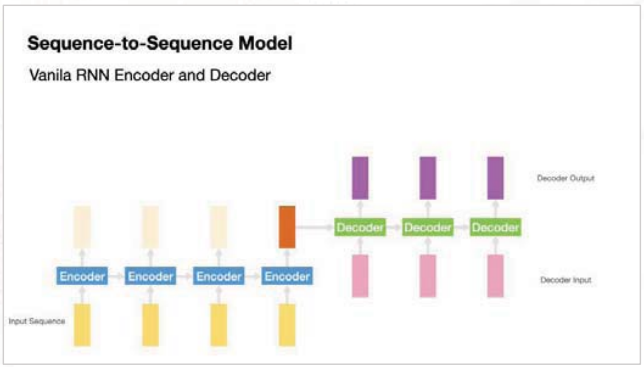
4. Michael Patrick Menden, Dennis Wang, Yuanfang Guan, Michael Mason, Bence Szalai, Krishna C Bulusu, Thomas Yu, Jaewoo Kang, **Minji Jeon**, Russ Wolfinger, Tin Nguyen, Mikhail Zaslavskiy, AstraZeneca-Sanger Drug Combination DREAM Consortium, In Sock Jang, Zara Ghazoui, Mehmet Eren Ahsen, Robert Vogel, Elias Chaibub Neto, Thea Norman, Eric KY Tang, Mathew J Garnett, Giovanni Di Veroli, Stephen Fawell, Gustavo Stolovitzky, Justin Guinney, Jonathan R Dry, Julio Saez-Rodriguez*, "Community assessment to advance computational prediction of cancer drug combinations in a pharmacogenomic screen" Nature Communications, 10.1 (2019): 2674.
5. **Minji Jeon**, Sunkyu Kim, Sungjoon Park, Heewon Lee, Jaewoo Kang* "In silico drug combination discovery for personalized cancer therapy" BMC systems biology, 2018, 12.2: 16.

KSBi-BIML 2024

Introduction to Transformers

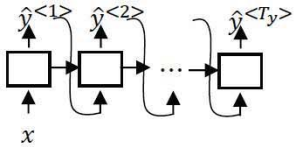
고려대학교 전민지

Transformers Intuition

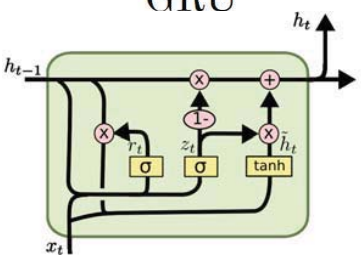


Increased complexity, sequential

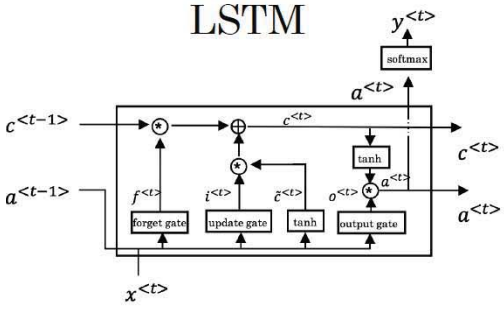
RNN



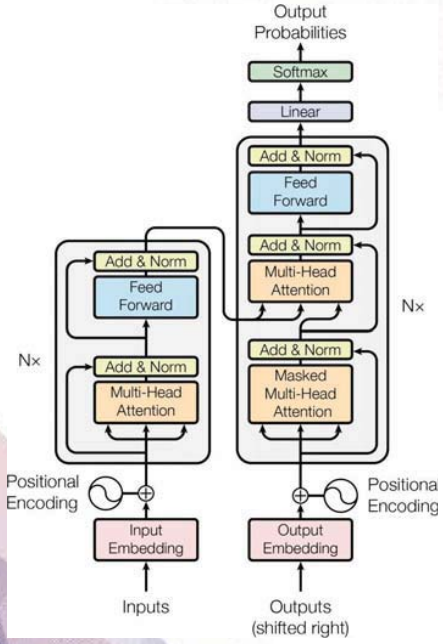
GRU

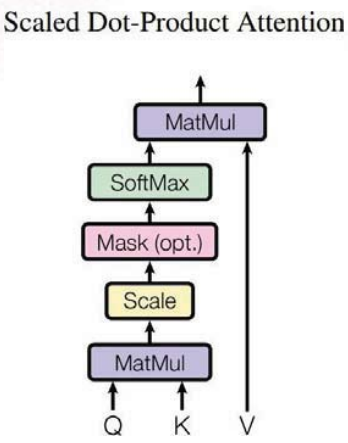


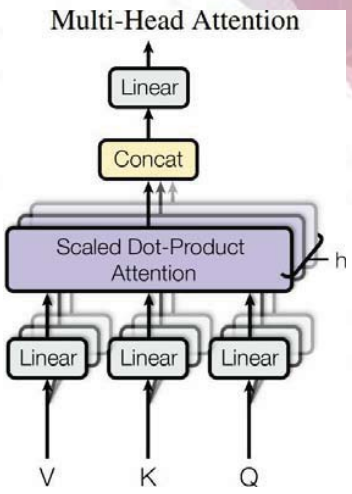
LSTM



Transformers Intuition



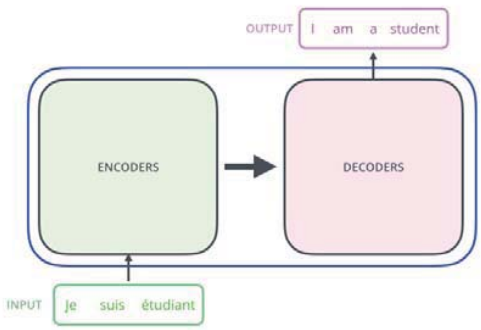




Transformers Intuition

- Transformer (Vaswani et al., 2017)
  - A model that uses attention to boost the speed with which these models can be trained and easy to parallelize
  - A high level look:

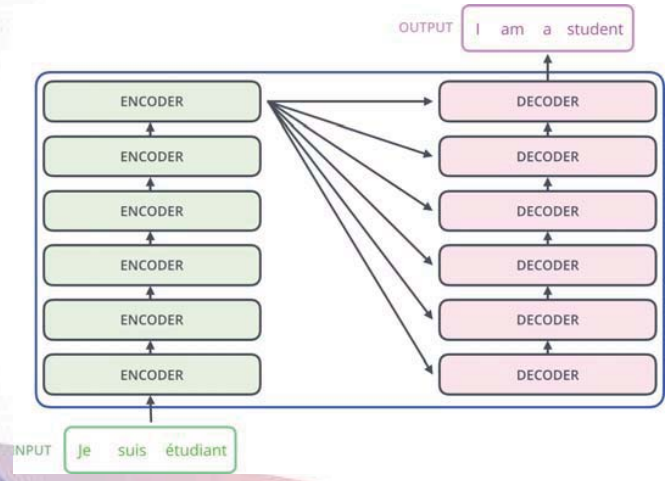
  - Inside the transformer, there are an encoding component and a decoding components and connections between them



Transformers Intuition

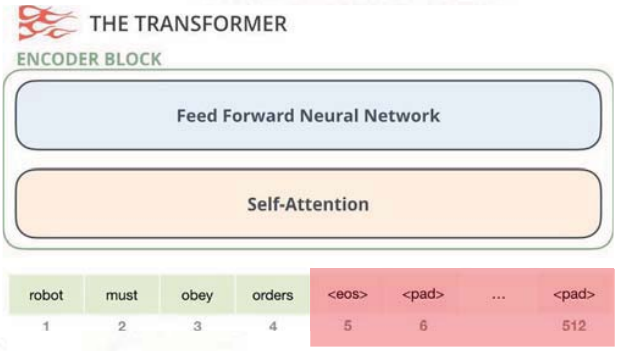
- The encoding component is a stack of encoders
- The decoding component is a stack of decoders of the same number

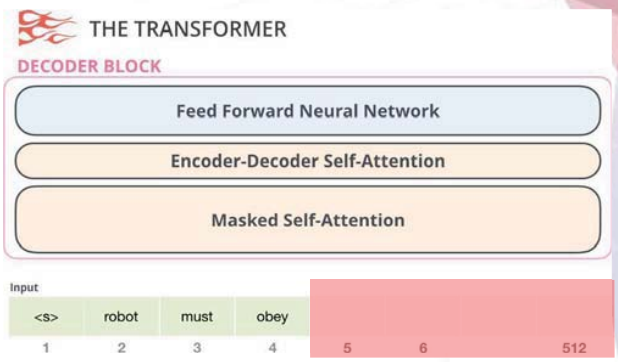
Note: The original paper stacks six of them on top of each other, but there is nothing magical about the number six



Transformers Intuition

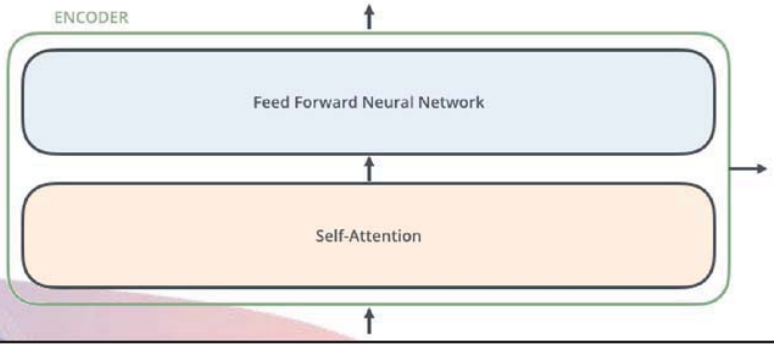
- Encoding block vs. Decoding block = Unmasked vs. Masked





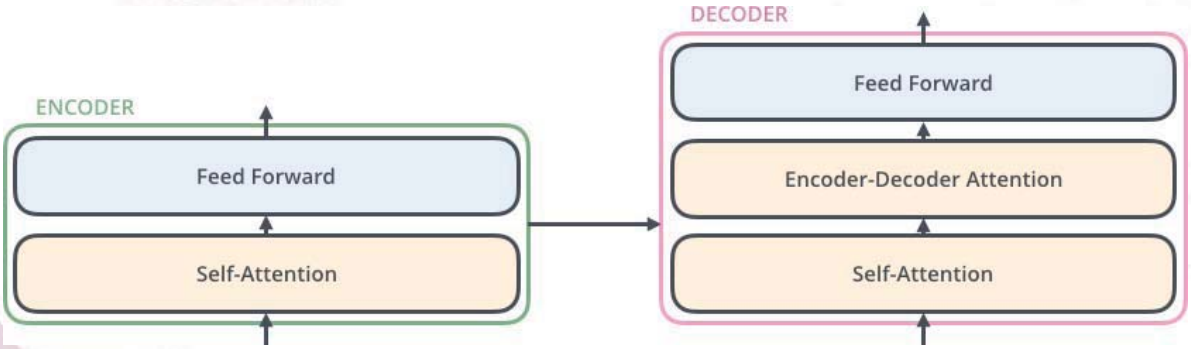
Transformers Intuition

- **The encoder** are all identical in structure (they ***do not*** share the weights), each of which is broken down into two sub-layers
  - **The encoder's input first flow through a self-attention layer** (a layer that helps the encoder look at other words in the input sentence as it encodes a specific word)
  - The output of the self-attention layer are fed to a feed-forward neural network
  - The exact same feed-forward network is independently applied to each position



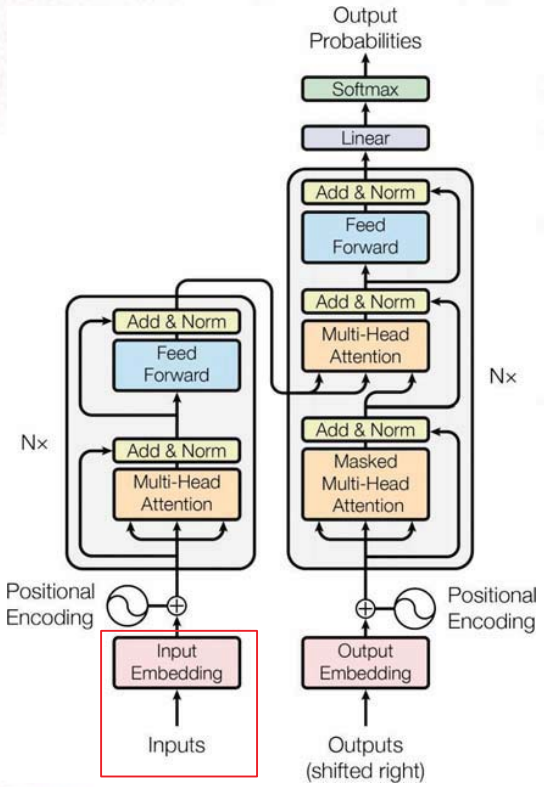
Transformers Intuition

- **The decoder** has both those layers, but between them is an attention layer that helps the decoder focus on relevant parts of the input sentence



Transformer Details: Input Embeddings

- **1. Input Embeddings**



Transformer Details: Input Embeddings

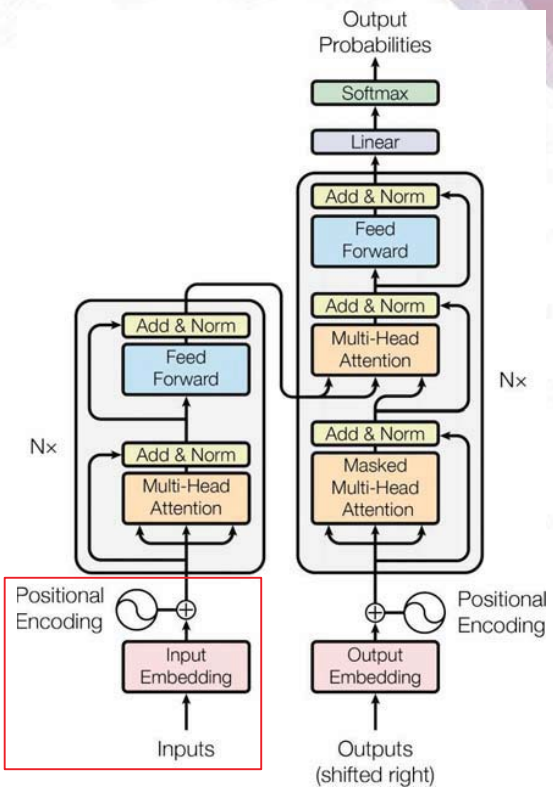
- Let's begin by turning **each input word into a vector** using an embedding algorithm

Each word is embedded into a vector of size 512. We'll represent those vectors with these simple boxes.

  - The embedding only happens in the bottom-most encoder
  - The abstraction that is common to all the encoders is that they receive a list of vectors each of the size 512
  - In the bottom encoder that would be the word embeddings, but in other encoders, it would be the output of the encoder that is directly below
  - The size of this list is a hyperparameter we can set – basically it would be the length of the longest sentence in our training dataset

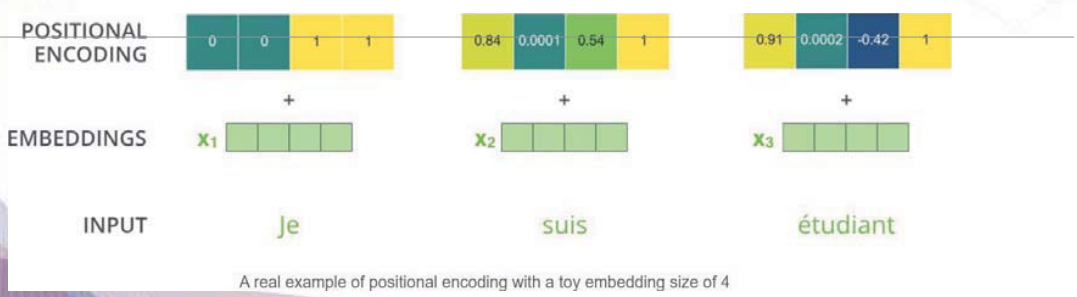
Transformer Details: Positional Encoding

- **2. Positional Encoding**



Transformer Details: Positional Encoding

- Positional encoding
  - A way to account for the order of the words in the input sequence
  - A vector added to each input embedding
  - Provides meaningful distances between the embedding vectors once they are projected into Q/K/V vectors and during dot-product attention

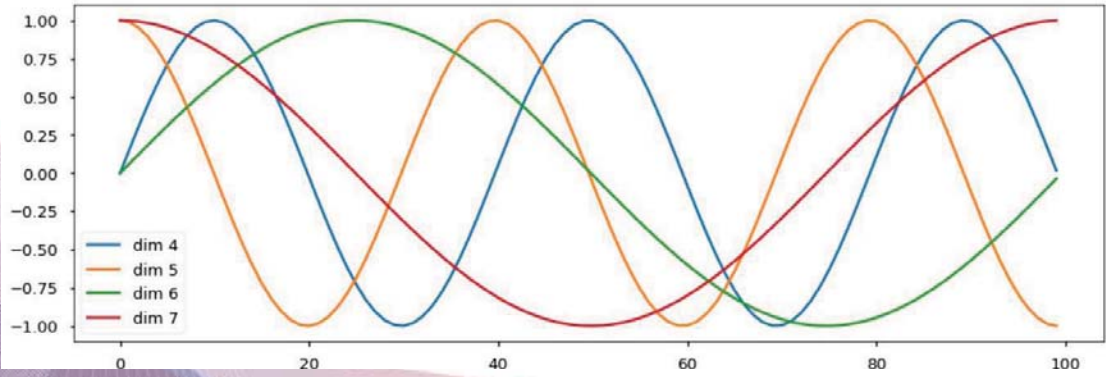


A real example of positional encoding with a toy embedding size of 4

Transformer Details: Positional Encoding

- Two properties that a good positional encoding scheme should have
  - The norm of encoding vector is the same for all positions
  - The further the two positions, the larger the distance

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$



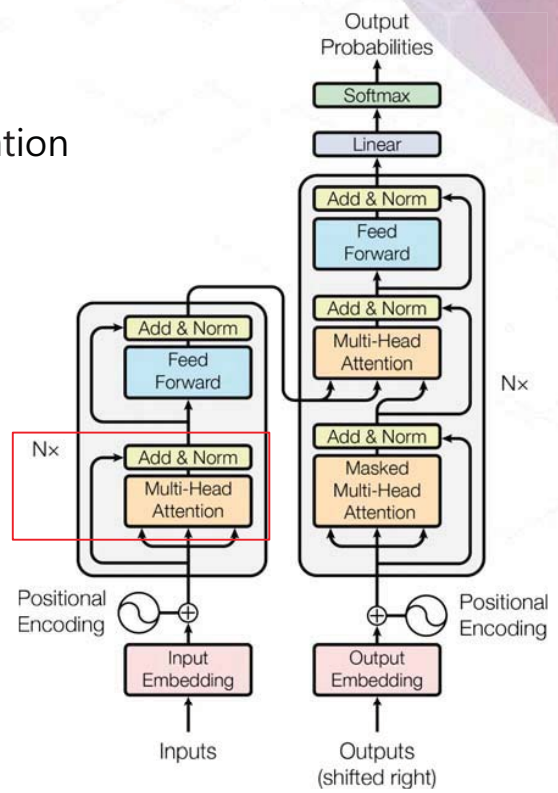
Transformer Details: Positional Encoding

- A Simple Example (n = 10, dim = 10)
  - Distances between two positional encoding vectors

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
X1	0.000	1.275	2.167	2.823	3.361	3.508	3.392	3.440	3.417	3.266
X2	1.275	0.000	1.104	2.195	3.135	3.511	3.452	3.442	3.387	3.308
X3	2.167	1.104	0.000	1.296	2.468	3.067	3.256	3.464	3.498	3.371
X4	2.823	2.195	1.296	0.000	1.275	2.110	2.746	3.399	3.624	3.399
X5	3.361	3.135	2.468	1.275	0.000	1.057	2.176	3.242	3.659	3.434
X6	3.508	3.511	3.067	2.110	1.057	0.000	1.333	2.601	3.169	3.118
X7	3.392	3.452	3.256	2.746	2.176	1.333	0.000	1.338	2.063	2.429
X8	3.440	3.442	3.464	3.399	3.242	2.601	1.338	0.000	0.912	1.891
X9	3.417	3.387	3.498	3.624	3.659	3.169	2.063	0.912	0.000	1.277
X10	3.266	3.308	3.371	3.399	3.434	3.118	2.429	1.891	1.277	0.000

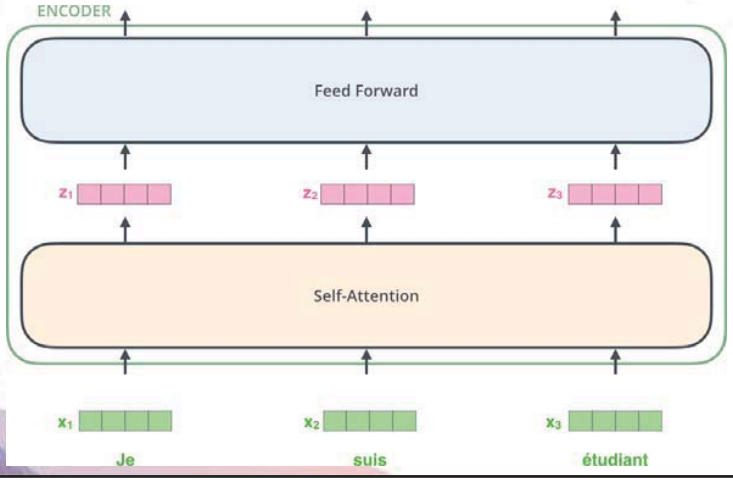
Transformer Details: Self-Attention

- Multi-Head Attention
- Residual connection & Normalization



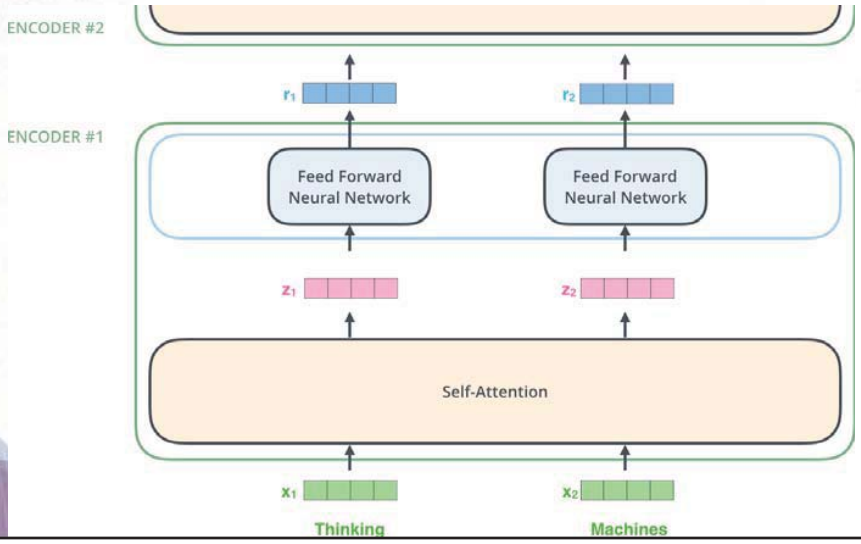
Transformer Details: Self-Attention

- After embedding the words, each of them flows through each of the two layers of the encoder
  - Word in each position flows through its own path in the encoder
    - There are dependencies between these paths in the self-attention layer
    - The feed-forward layer does not have those dependencies (parallelization becomes possible)



Transformer Details: Self-Attention

- Encoding procedure
  - An encoder receives a list of vectors as input
  - It processes this list by passing these vectors into a 'self-attention' layer, then into a feed-forward neural network, then sends out the output upwards to the next encoder



Transformer Details: Self-Attention

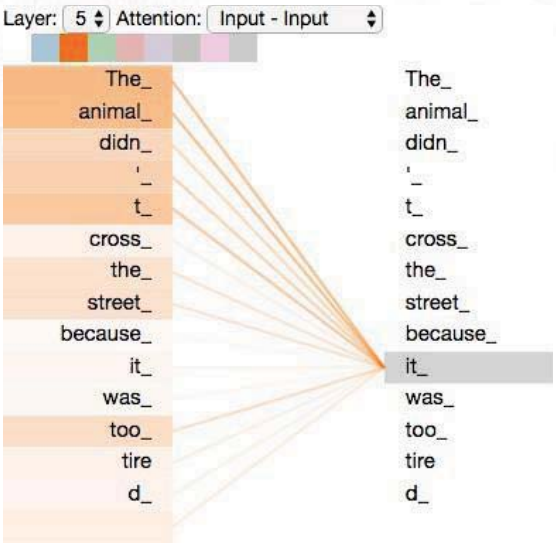
- Self-Attention at a High Level
  - Input sentence to translate:

    The animal didn't cross the street because **it** was too tired

  - What does "**it**" refer to? street or animal?
  - Simple question to a human but not as simple to an algorithm
  - Self attention allows it to look at other positions in the input sequence for clues that can help lead to a better encoding for this word
  - Self-attention is the method the Transformer uses to bake the "understanding" of other relevant words into the one we're currently processing

Transformer Details: Self-Attention

- Self-Attention example

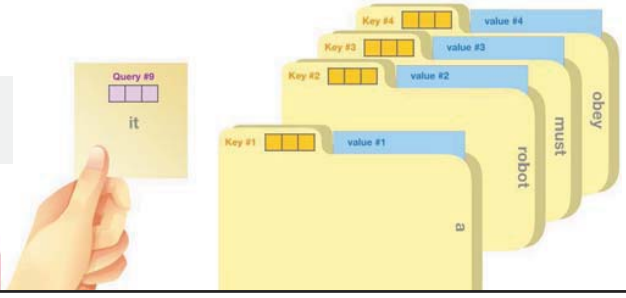


https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb#scrollTo=OJKU36QAfqOC

Transformer Details: Self-Attention

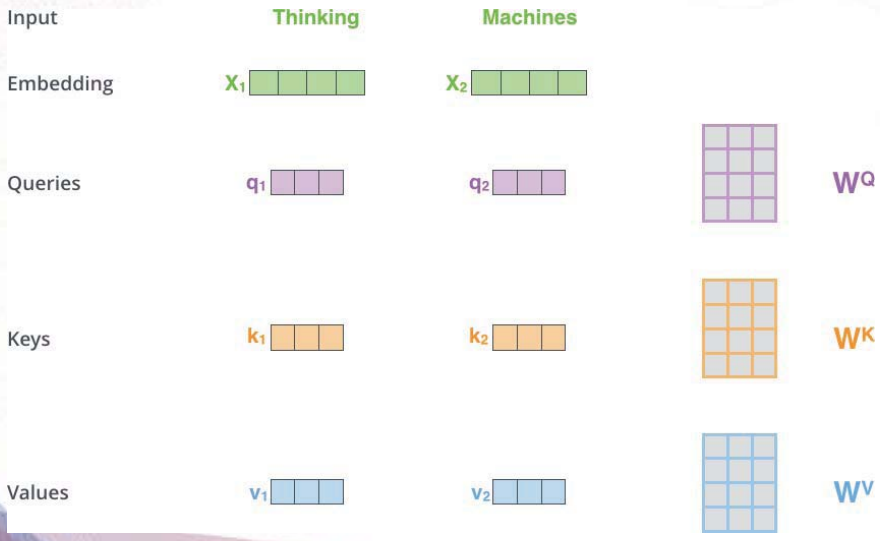
- **Step 1: Create three vectors from each of the encoder's input vectors**
  - **Query**: The query is a representation of the current word used to score against all the other words (using their keys). We only care about the query of the token we're currently processing.
  - **Key**: Key vectors are like labels for all the words in the segment. They're what we match against in our search for relevant words.
  - **Value**: Value vectors are actual word representations, once we've scored how relevant each word is, these are the values we add up to represent the current word.

"A robot must obey the orders given it by human beings"



Transformer Details: Self-Attention

- **Step 1: Create three vectors from each of the encoder's input vectors**
  - These vectors (Q/K/V) are created by multiplying the embedding by three matrices (W^Q, W^K, W^V) that we trained during the training process



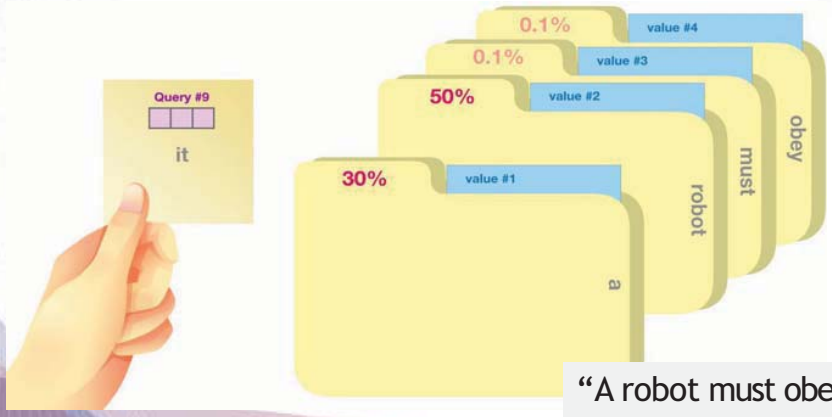
Transformer Details: Self-Attention

- **Step 1: Create three vectors from each of the encoder's input vectors**

  - Note) These new vectors are smaller in dimension that the embedding vector
  - Q, K, and V are 64-dim. while embedding and encoder input/output vectors are 512-dim.
  - They do not have to be smaller but it is an architecture choice to make the computation of multi-headed attention (mostly) constant

Transformer Details: Self-Attention

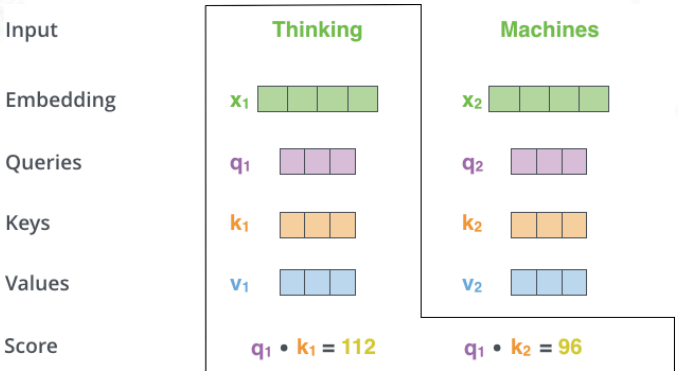
- **Step 2: Calculate self-attention to get a score**
  - We need to score each word of the input sentence against this word ("it")
  - The score determines how much focus to place on other parts of the input sentence as we encode a word at a certain position
  - Multiplying the query vector by each key vector produces a score for each folder
  - (technically: dot product followed by softmax)



"A robot must obey the orders given **it** by human beings"

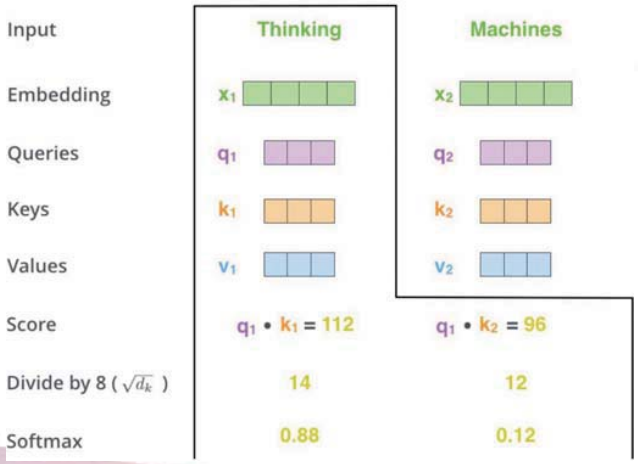
Transformer Details: Self-Attention

- **Step 2: Calculate self-attention to get a score**
  - The score is calculated by taking the dot product of the query vector with the key vector of the respective word we are scoring



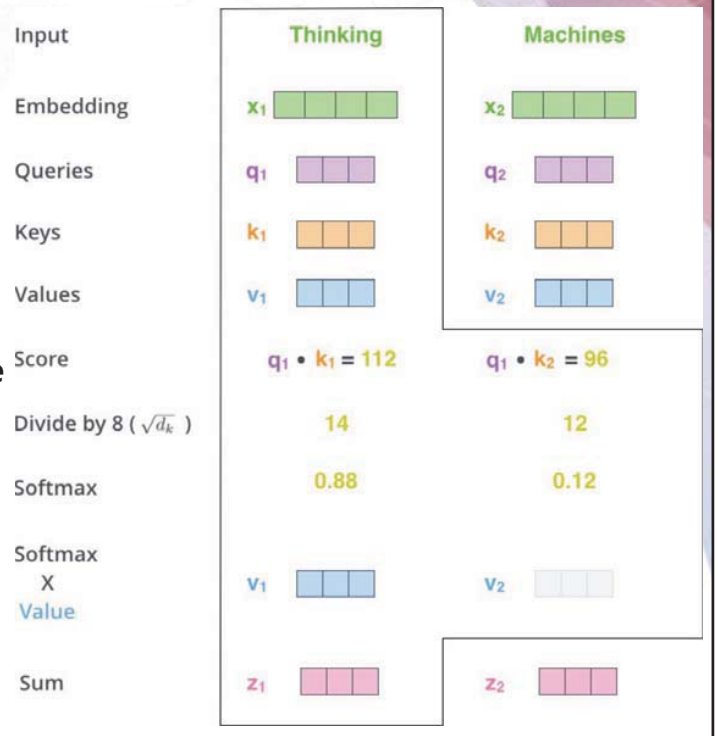
Transformer Details: Self-Attention

- **Step 3: Divide the score by $\sqrt{d_k}$ (= 8 in the original paper since d_k = 64)**
  - This leads to having more stable gradients
- **Step 4: Pass the result through a softmax operation**
  - The softmax score determines how much each word will be expressed at this position



Transformer Details: Self-Attention

- **Step 5: Multiply each value vector by the softmax score**
  - to keep intact the values of the words we want to focus on
  - drown-out irrelevant words
- **Step 6: Sum up the weighted value vector which produces the output of the self-attention layer at this position**



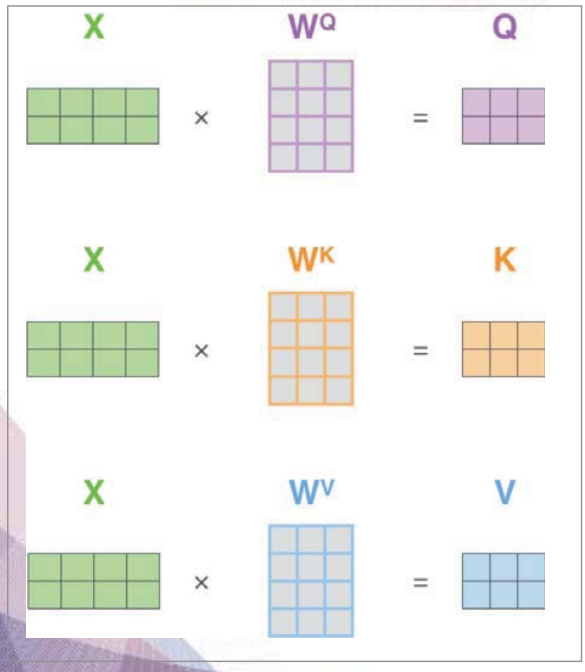
Transformer Details: Self-Attention

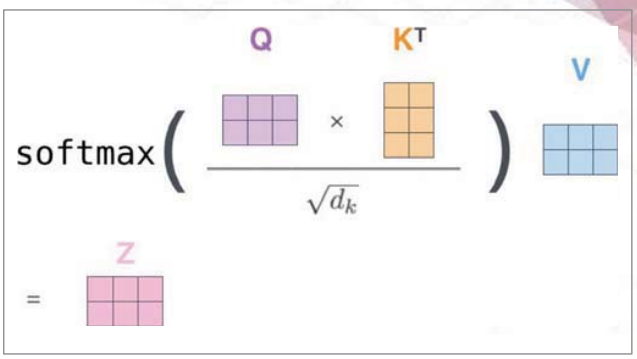
- **Step 5: Multiply each value vector by the softmax score**
- **Step 6: Sum up the weighted value vector which produces the output of the self- attention layer at this position**

Word	Value vector	Score	Value X Score
<s>		0.001	
a		0.3	
robot		0.5	
must		0.002	
obey		0.001	
the		0.0003	
orders		0.005	
given		0.002	
it		0.19	
		Sum:	

Transformer Details: Self-Attention

- Matrix calculation of self-attention

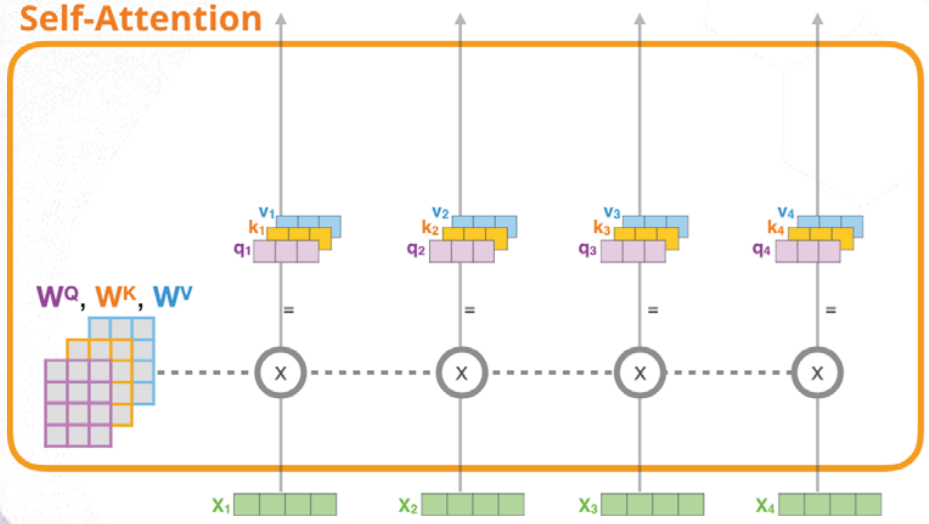




Transformer Details: Self-Attention

- Another illustration of Self-Attention

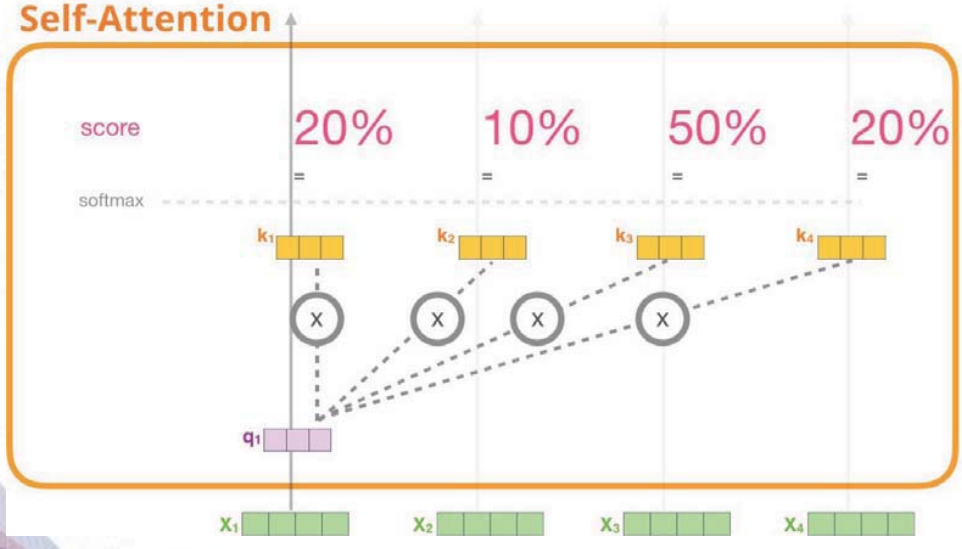
1) For each input token, create a query vector, a key vector, and a value vector by multiplying by weight Matrices W^Q, W^K, W^V



Transformer Details: Self-Attention

- Another illustration of Self-Attention

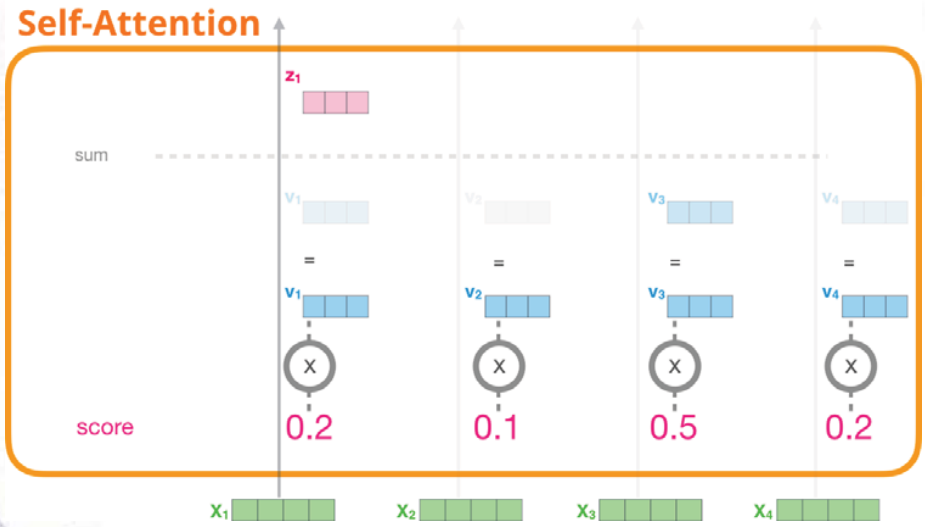
2) Multiply (dot product) the current query vector, by all the key vectors, to get a score of how well they match



Transformer Details: Self-Attention

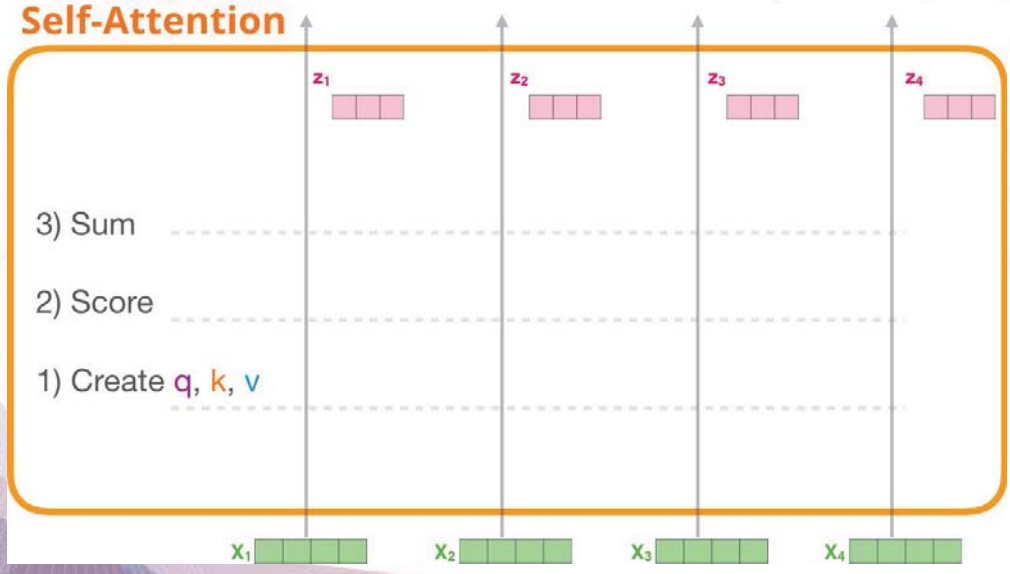
- Another illustration of Self-Attention

3) Multiply the value vectors by the scores, then sum up



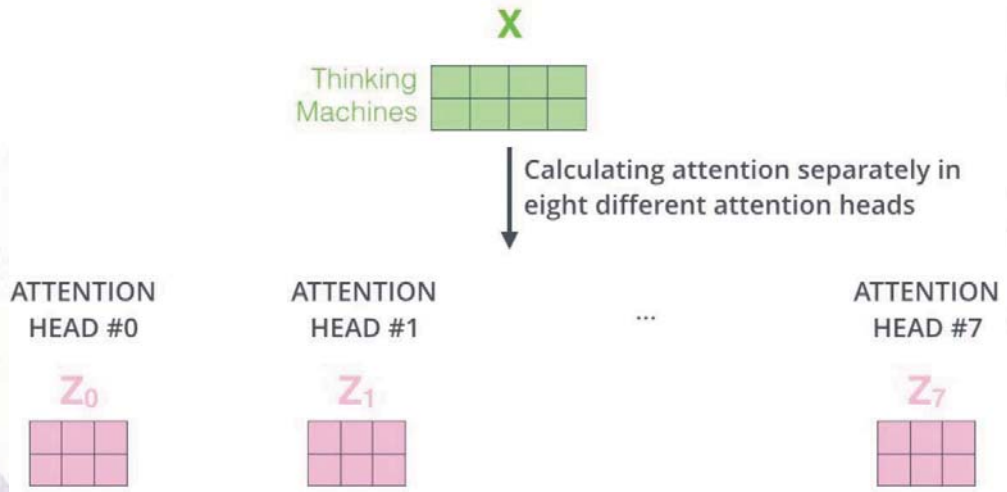
Transformer Details: Self-Attention

- Another illustration of Self-Attention
  - Do the same operation for each path to end up with a vector representing each token containing appropriate context of that token



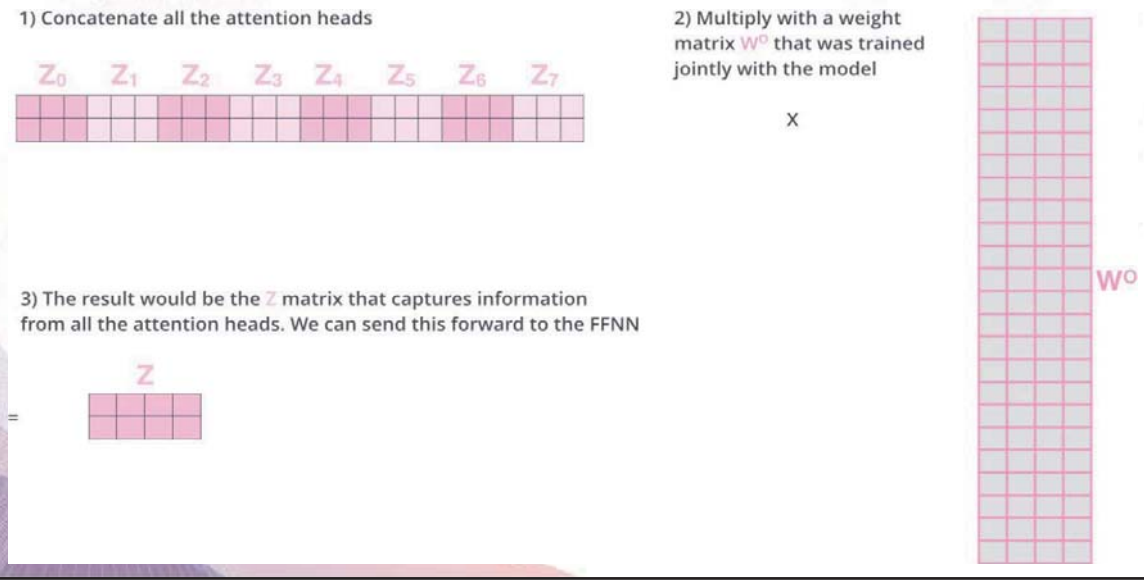
Transformer Details: Multi-headed-Attention

- Multi-headed attention
  - Expand the model's ability to focus on different positions



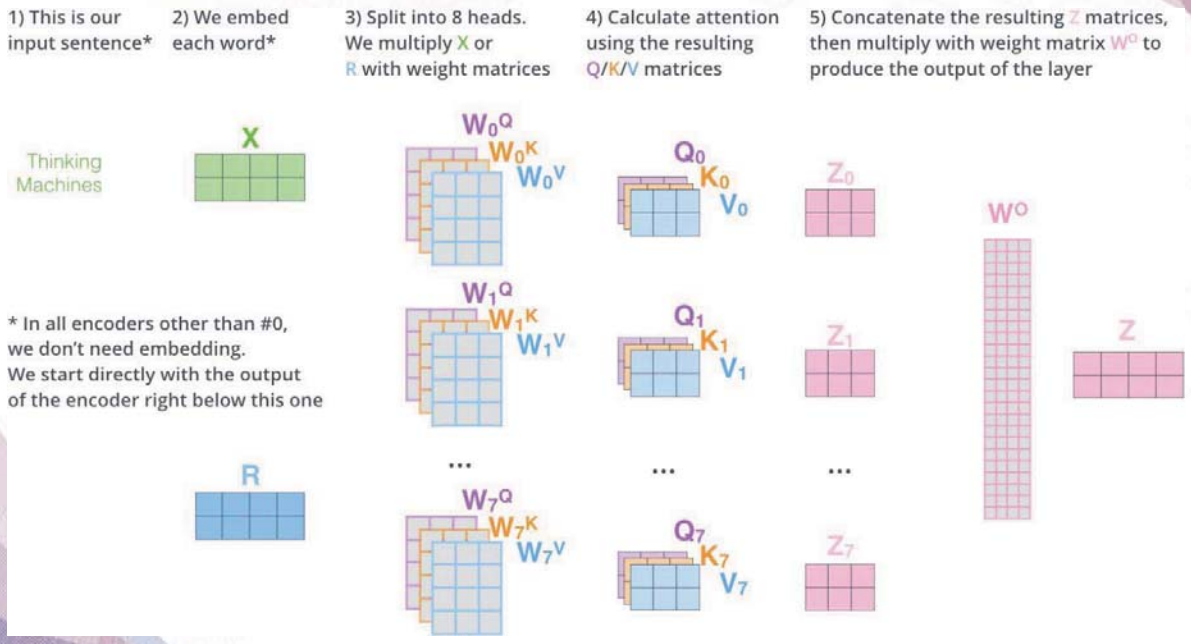
Transformer Details: Multi-headed-Attention

- Multi-headed attention
  - Attention heads are concatenated and multiplied by an additional weight matrix to be used as an input of feed-forward neural network



Transformer Details: Multi-headed-Attention

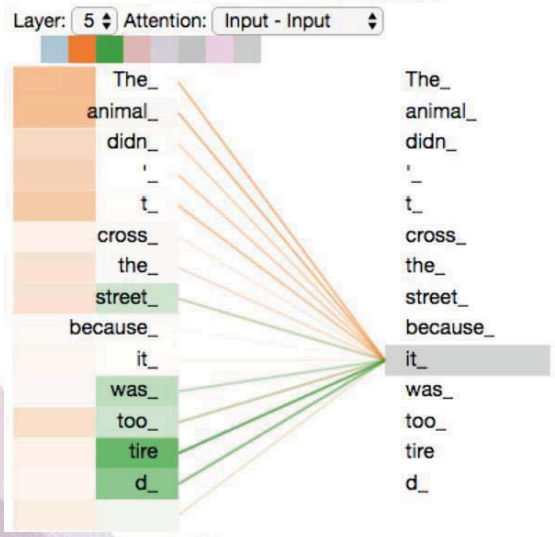
- Multi-headed attention



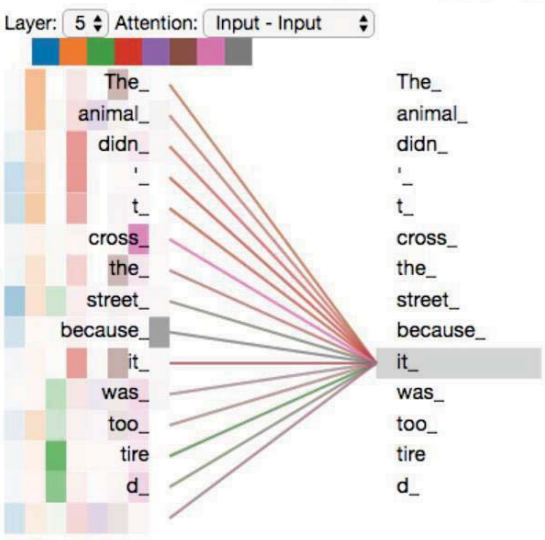
Transformer Details: Multi-headed-Attention

- Multi-headed attention

Attention with two heads

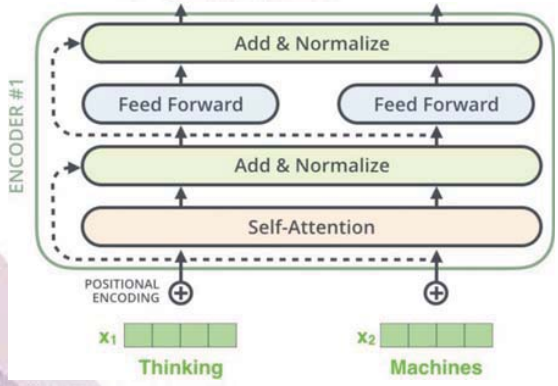


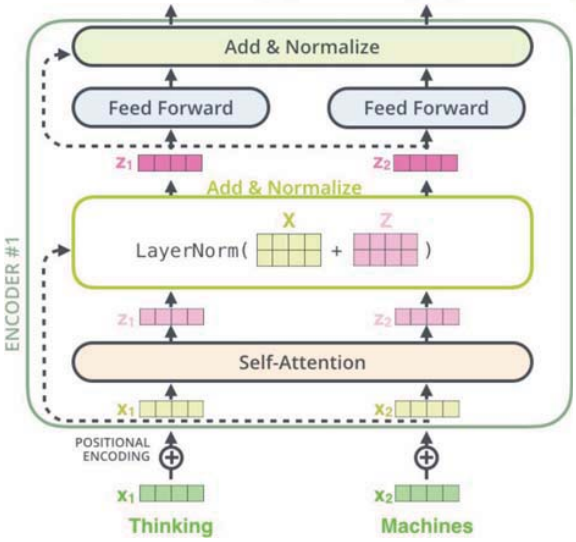
Attention with eight heads



Transformer Details: Residual Connection

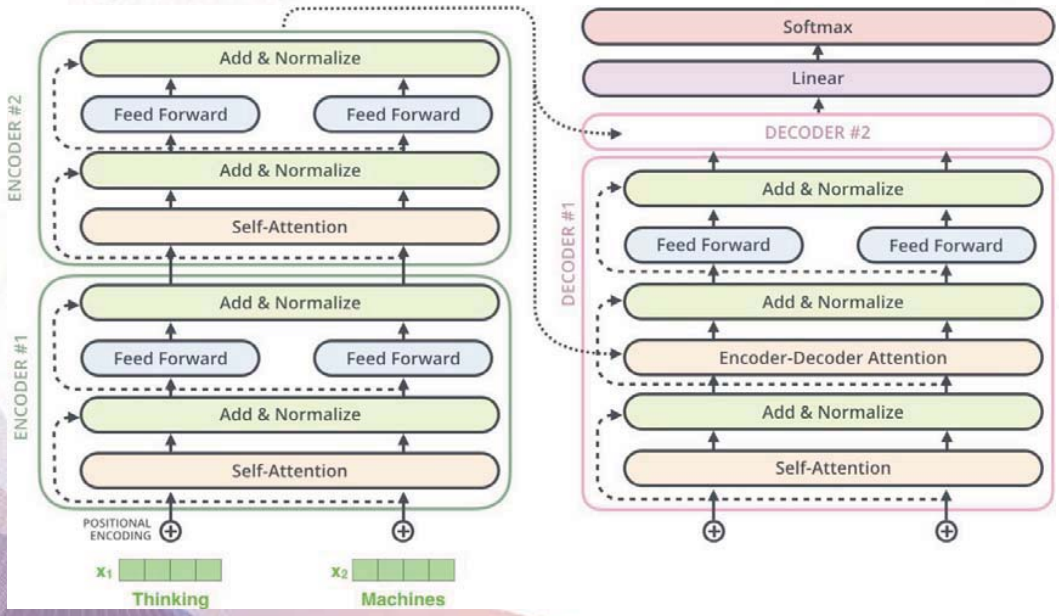
- Residual connection
  - Each sub-layer (self-attention, FFNN) in each encoder has a residual connection around it followed by a layer-normalization step





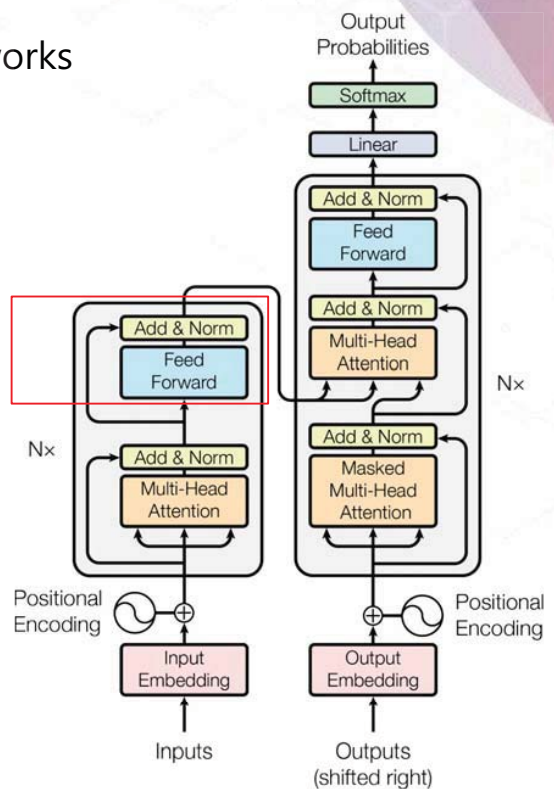
Transformer Details: Residual Connection

- Residual connection
  - This goes for the sub-layers of the decoder as well
  - Ex: 2 stacked encoders and decoders



Transformer Details: Feed Forward

- Position-wise Feed-Forward Networks

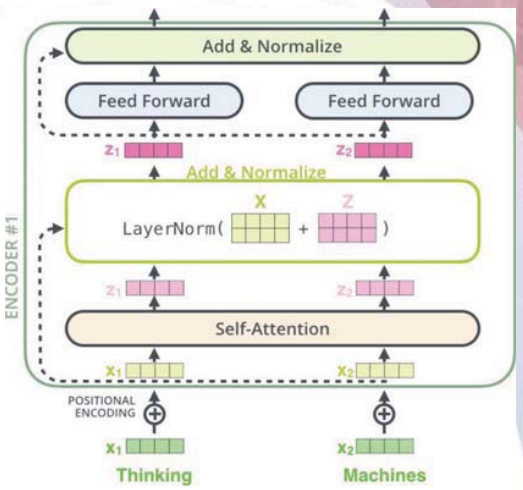


Transformer Details: Feed Forward

- Position-wise Feed-Forward Networks
  - Fully connected feed-forward network
  - Applied to each position separately and identically

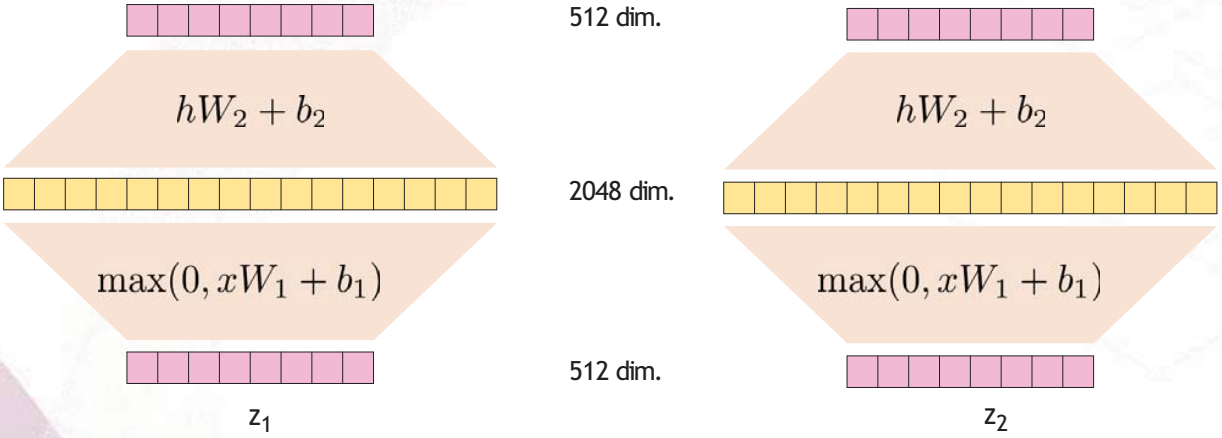
$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

  - The linear transformations are the same across different positions
  - They use different parameters from layer to layer



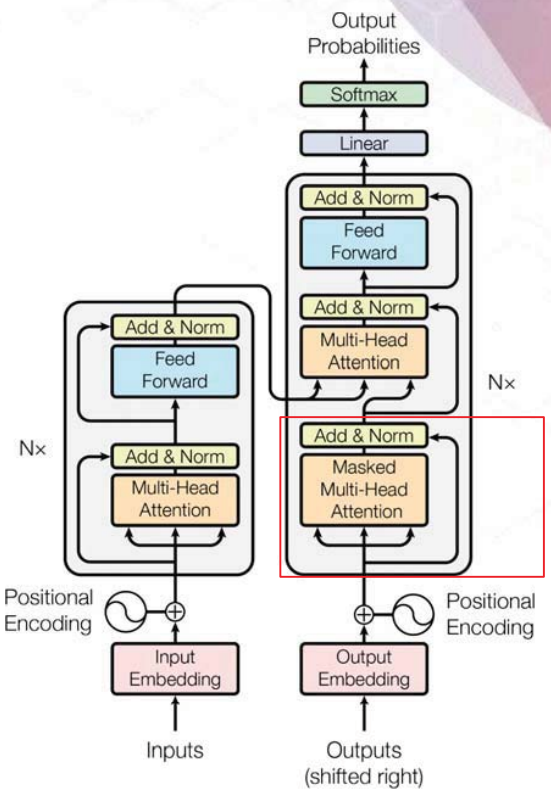
Transformer Details: Feed Forward

- Position-wise Feed-Forward Networks



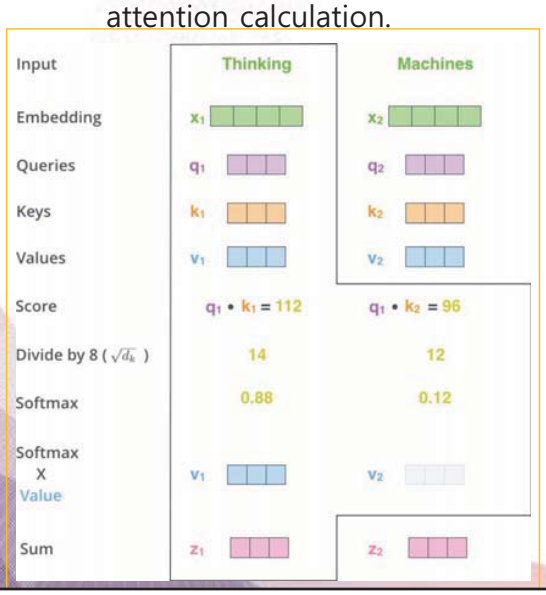
Transformer Details: Masked Multi-Head Attention

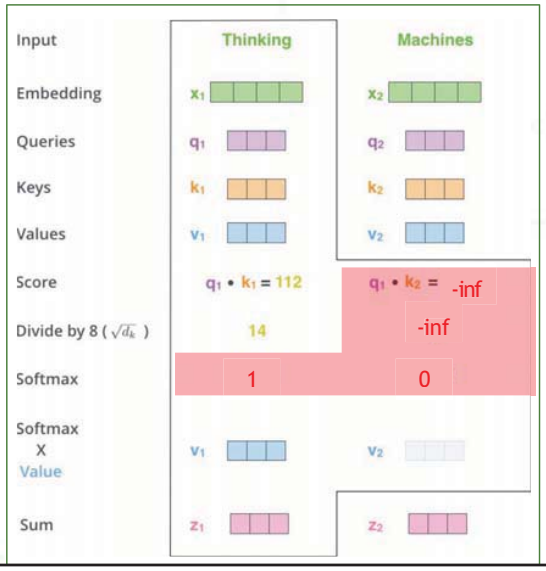
- Masked Multi-head Attention



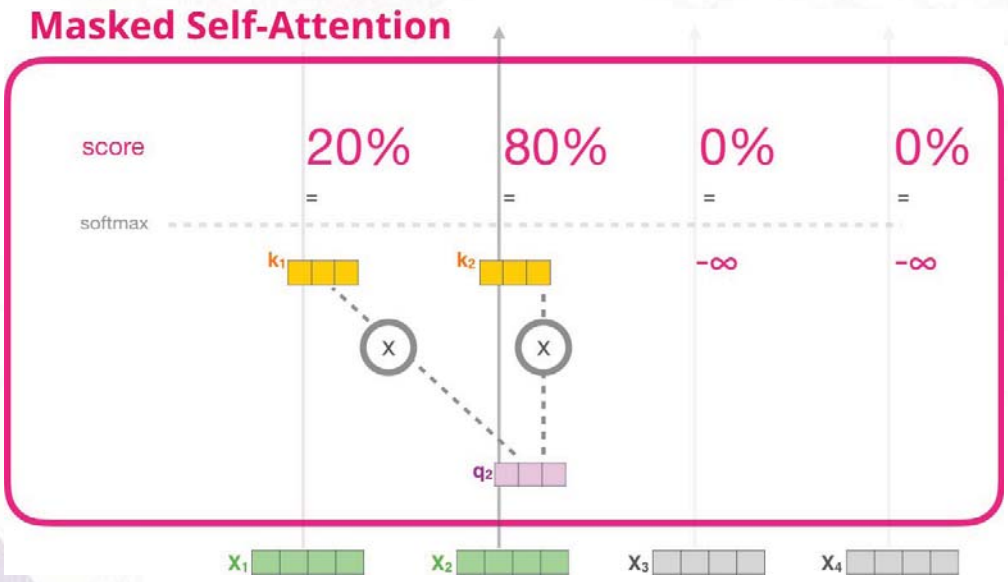
Transformer Details: Masked Multi-Head Attention

- Masked Multi-head Attention
  - Self attention layers in the decoder is only allowed to attend to earlier positions in the output sequence, which is done by masking future positions (setting them to –inf) before the softmax step in the self attention calculation.



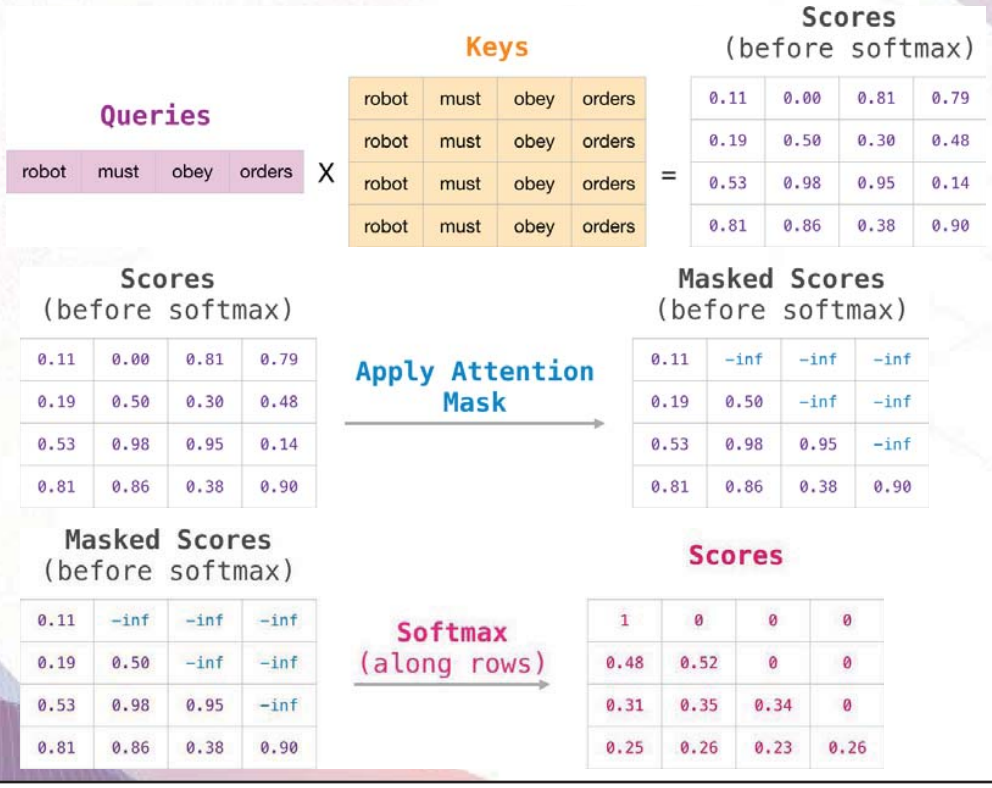


Transformer Details: Masked Multi-Head Attention



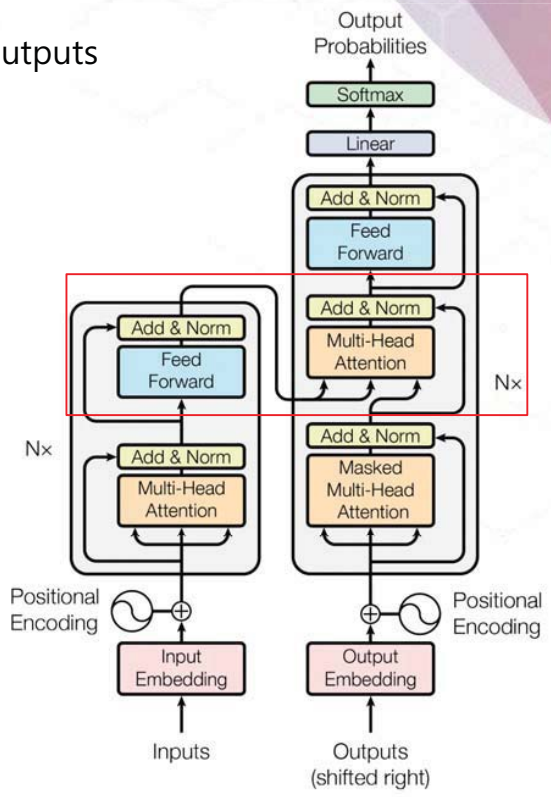
Transformer Details: Masked Multi-Head Attention

- Masked Multi-head Attention



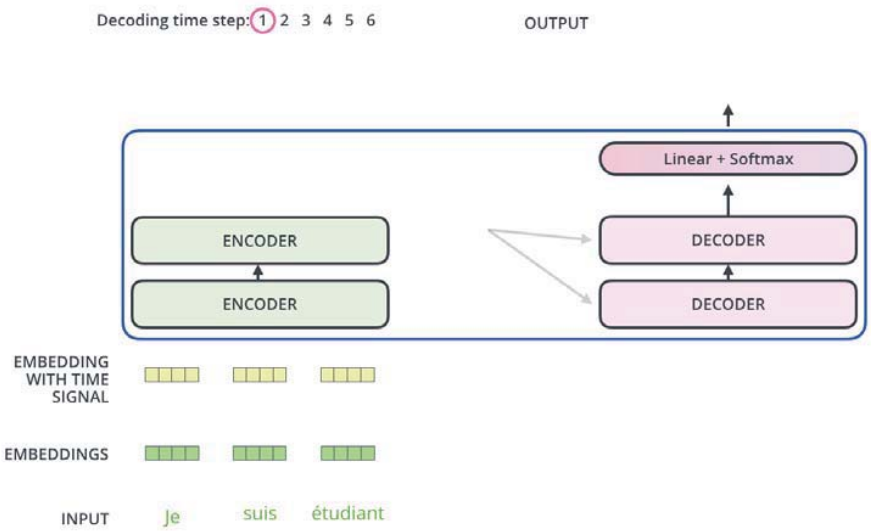
Transformer Details: Encoder-Decoder Multi-Head Attention

- Multi-Head Attention with Encoder Outputs



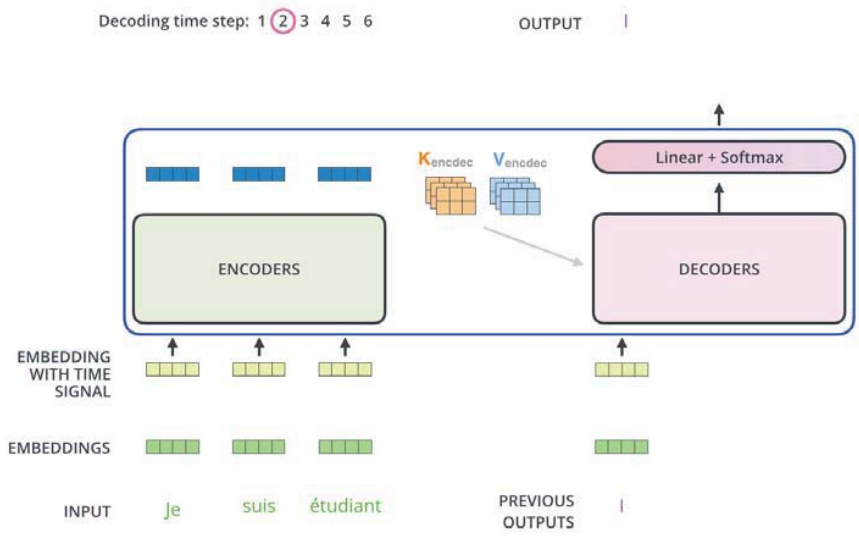
Transformer Details: Encoder-Decoder Multi-Head Attention

- The Decoder Side



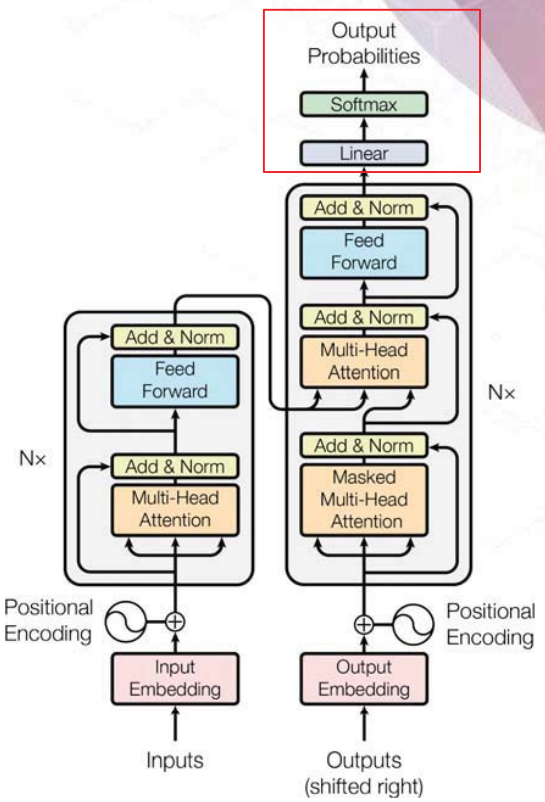
Transformer Details: Encoder-Decoder Multi-Head Attention

- The Decoder Side



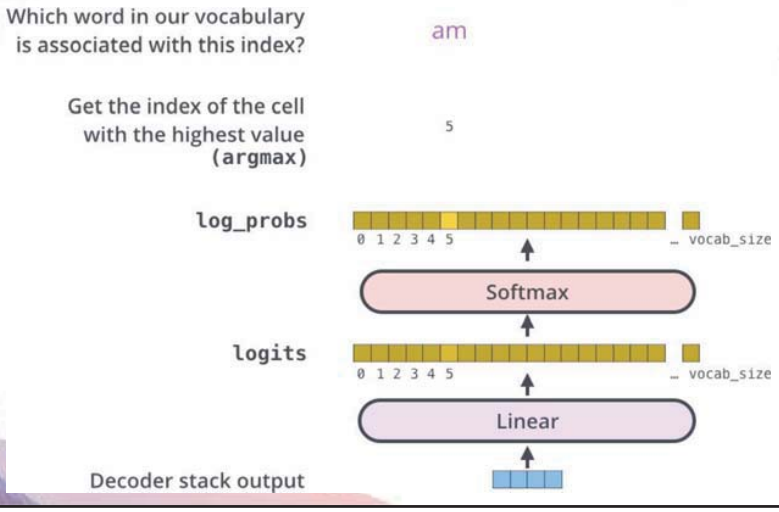
Transformer Details: Final Layer

- The Final Linear and Softmax Layer



Transformer Details: Final Layer

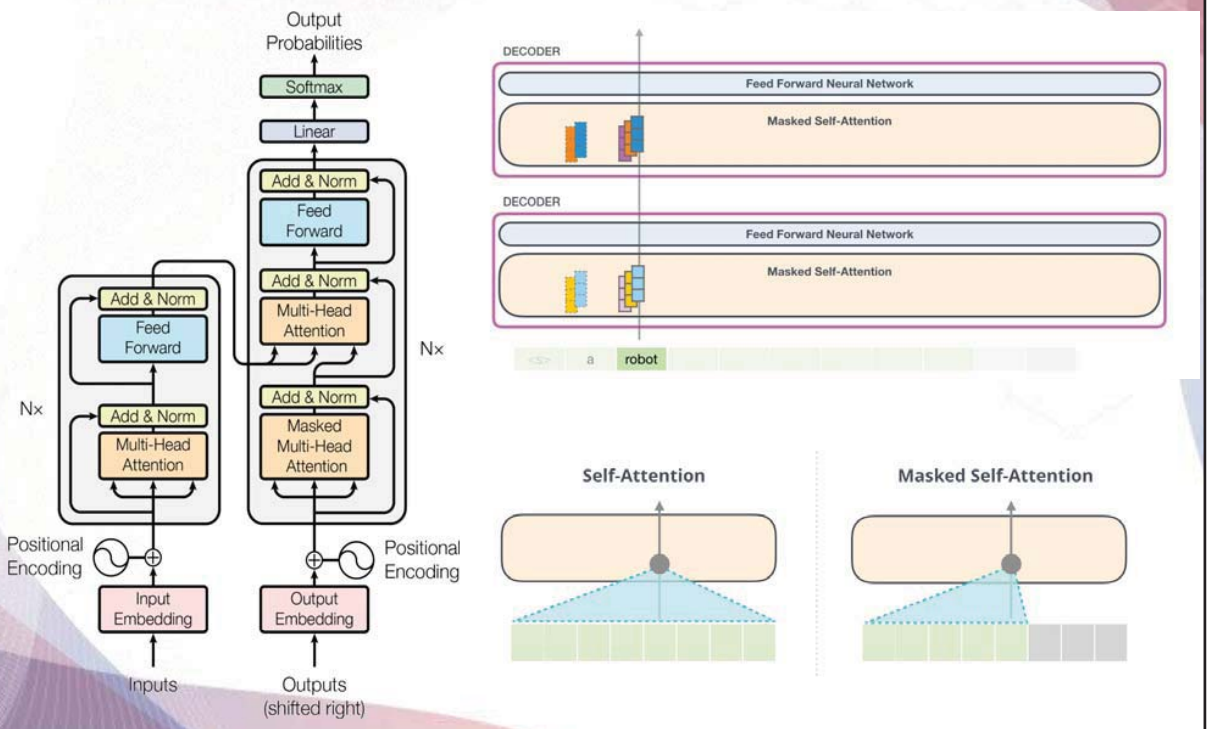
- The Final Linear and Softmax Layer
  - Linear layer: a simple fully connected neural network that projects the vector produced by the stack of decoders into a much larger vector called a logits vector
  - Softmax layer: turns those scores into probability
  - The cell with the highest probability is chosen, the word associated with it is produced as the output of this time step



Transformer-based Pretrained Language Models

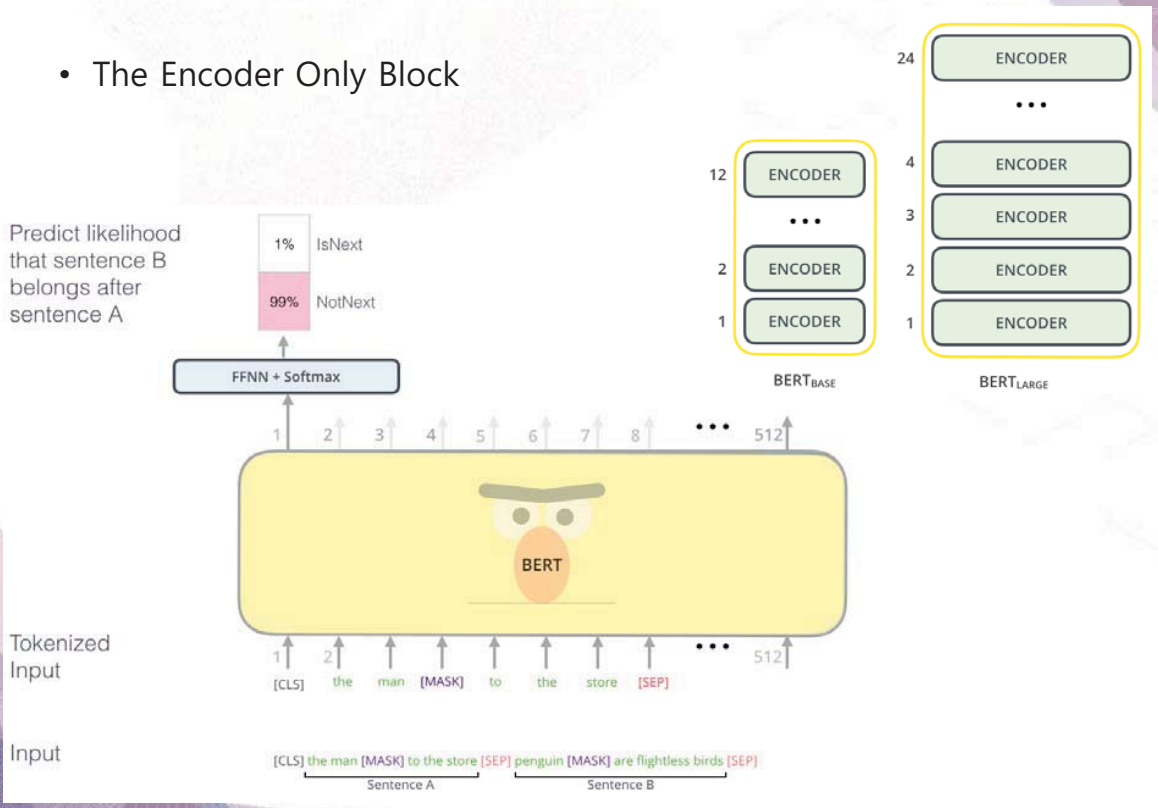
GPT: Generative Pre-Training of a Language Model

- The Decoder-Only Block

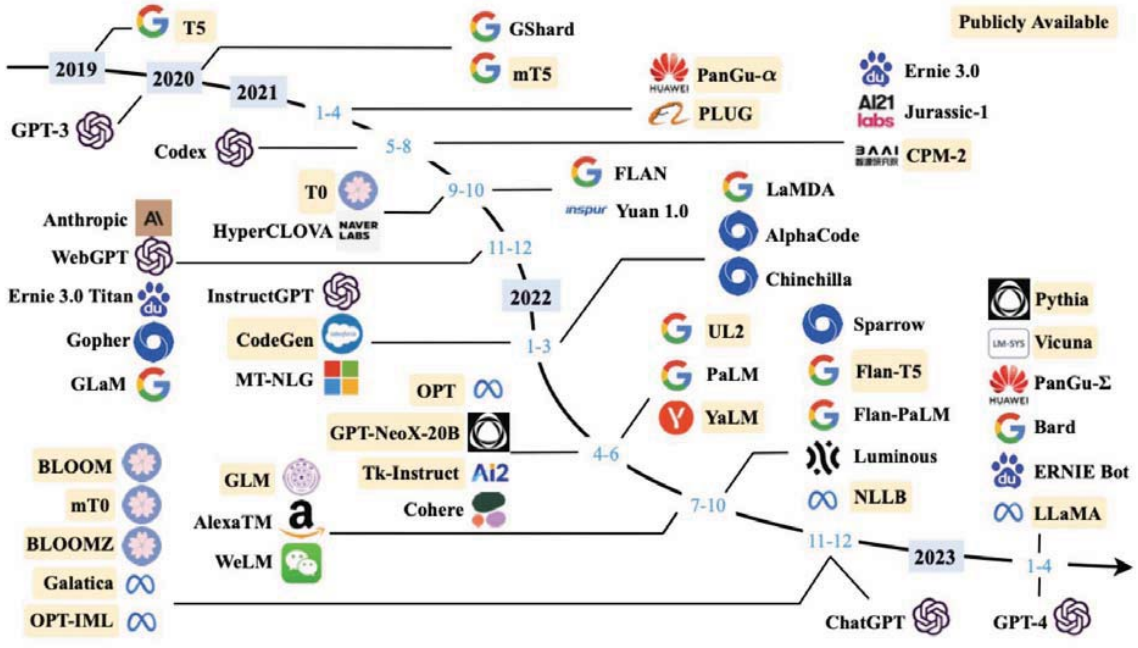


BERT: Bidirectional Encoder Representations from Transformer

- The Encoder Only Block

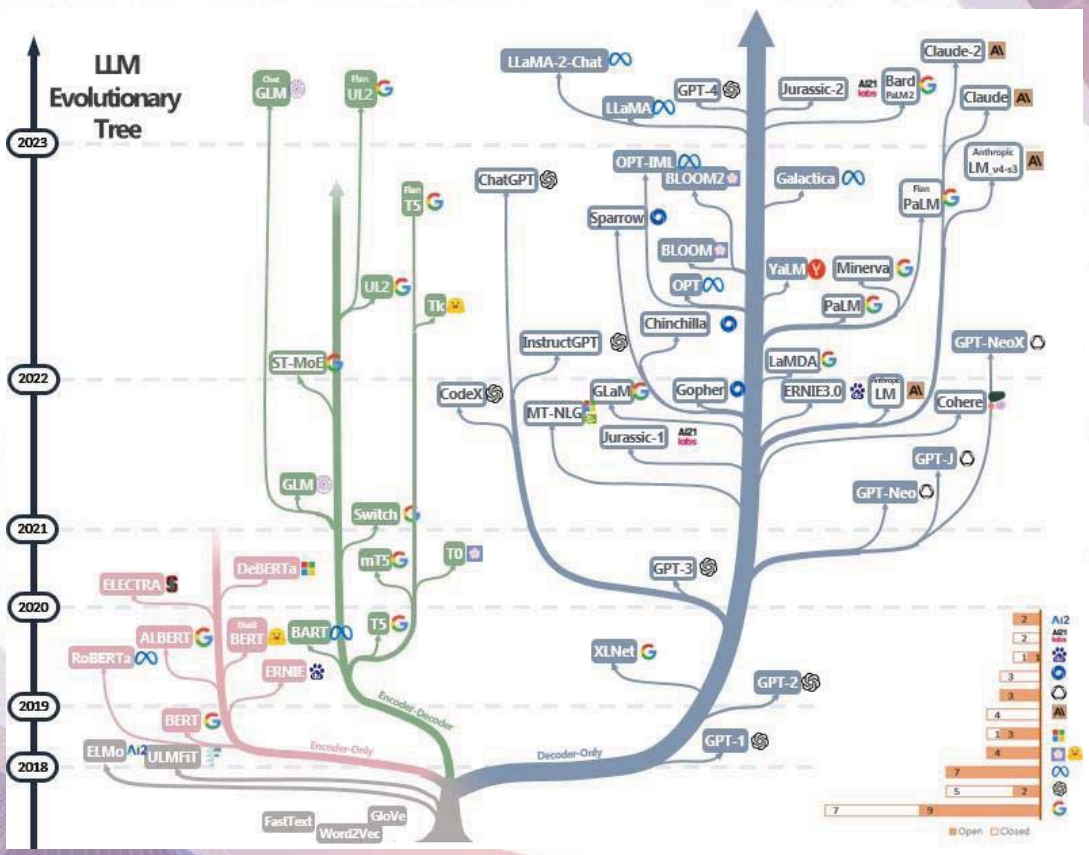


Major Large Language Model Timeline



https://brunch.co.kr/@brunchgpjz/49

Evolutional Tree of LLM



Transformers in Biomedical Domain

- BioBERT (Bioinformatics, 2019), BioGPT (Briefings in Bioinformatics, 2022)

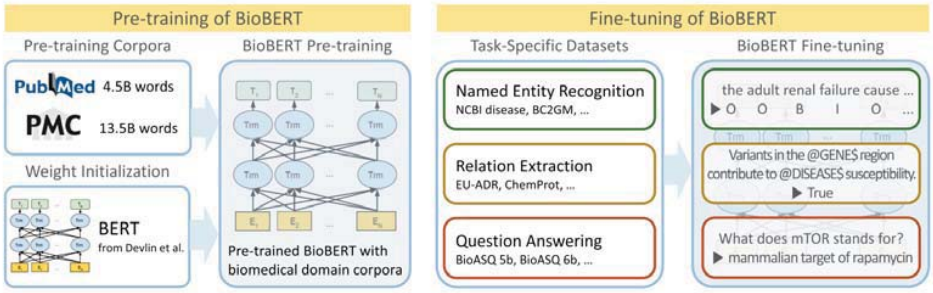
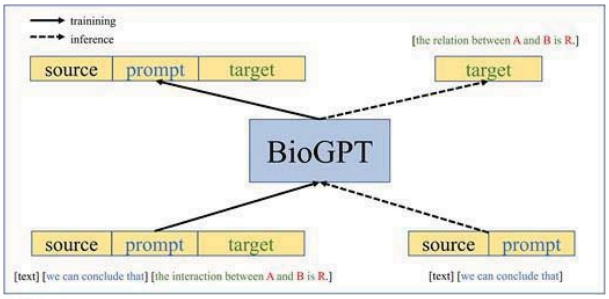
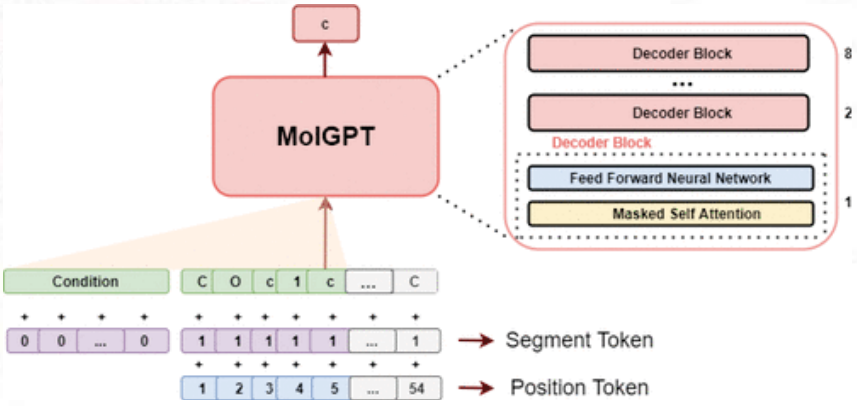


Fig. 1. Overview of the pre-training and fine-tuning of BioBERT



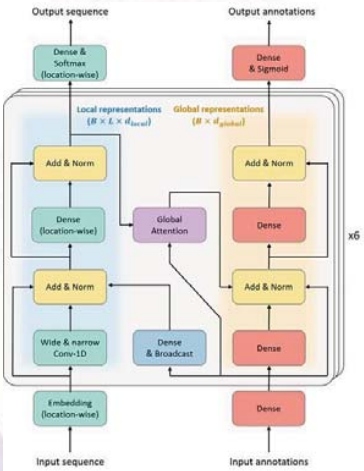
Transformers in Biomedical Domain

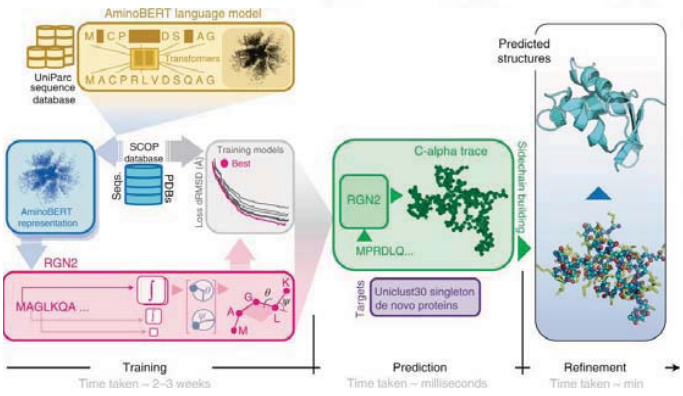
- MolGPT (JCIM 2021), ChemBERTa (NeurIPS 2020)



Transformers in Biomedical Domain

- ProteinBERT (Bioinformatics, 2022), RGN2 (Nature Biotechnology, 2022)





MolTrans: Molecular Interaction Transformer for drug-target interaction prediction

Kexin Huang, Cao Xiao, Lucas M Glass, Jimeng Sun

Bioinformatics (2021)

Introduction

◦ Motivation

- 신약 개발은 긴 연구 기간 대비 높은 연구 비용 및 낮은 성공률을 보임
- in-silico에서 Drug Target Interaction(DTI) 예측은 약물 설계 단계까지의 시간적, 비용적 소모를 크게 절감할 수 있음

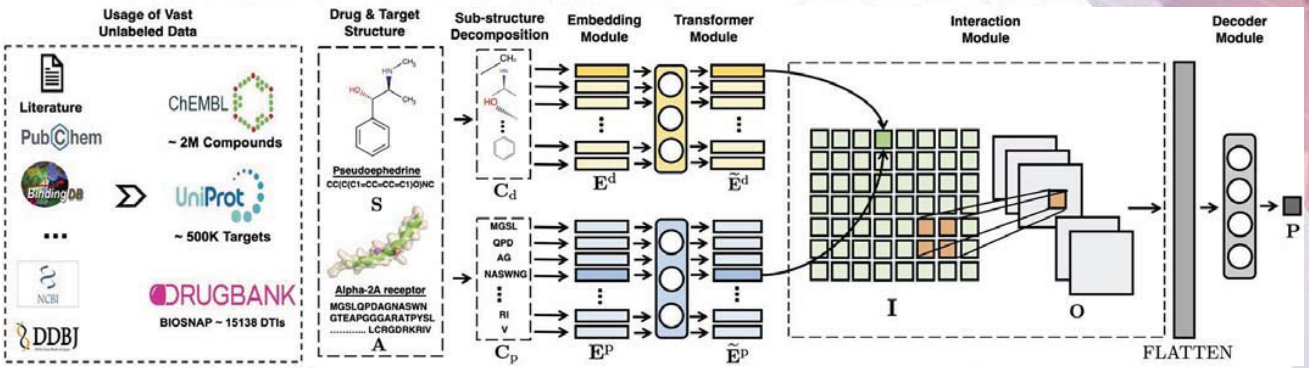
◦ Challenges

- 기존 연구에서는 약물과 단백질의 전체 분자 구조가 학습에 활용됨
  -> 어떤 substructure가 DTI에 관여하는지 해석하기 어려움
- DTI에 대한 unlabeled biomedical data는 학습에 활용되지 않음
  -> Interaction의 가능성이 있는 substructure가 무시됨

◦ Contribution

- 약물과 단백질의 substructure간 interaction을 학습 및 해석 할 수 있는 Transformer 기반 DTI prediction 모델 제시
- Interaction의 가능성이 있는 substructure의 수를 최대화하기 위한 substructure mining 기법 제시

Overview of the MolTrans



◦ Notations

Compound sequence $S = \{S_1, \ldots, S_n\}$ (n : drugs)

S_i : drug representation by the SMILES

e.g. CC(=O)OC1=CC=CC=C1C(=O)OC2=CC=CC(=C2)CO[N+](=O)[O-]

Protein sequence $\mathrm{A} = \{A_1, \ldots, A_m\}$ (m : proteins)

A_i : protein representation by a sequence of protein tokens

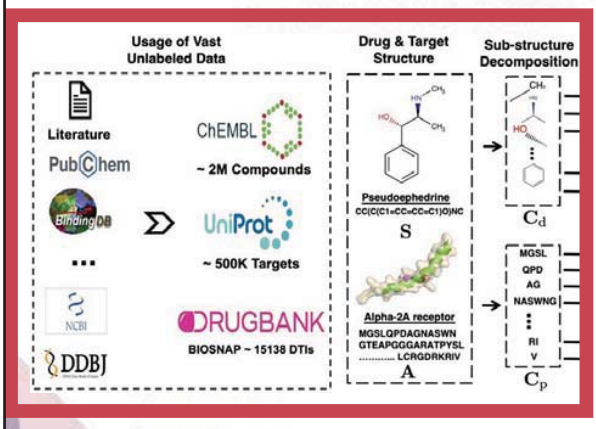
e.g. MNLLCCCCCSNMAPNQRVTRKWELFAGRNK...

◦ Problem definition

Binary classification task $\mathcal{F} : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$

Methods (1) - Tokenization

◦ Frequent Consecutive Sub-sequence (FCS) mining for Substructure Decomposition



◦ **Frequent Consecutive Sub-sequence (FCS) mining:** To extract high-quality sub-structures of drugs and proteins from millions of drugs and proteins sequences

◦ Drug, protein을 substructure로 분해

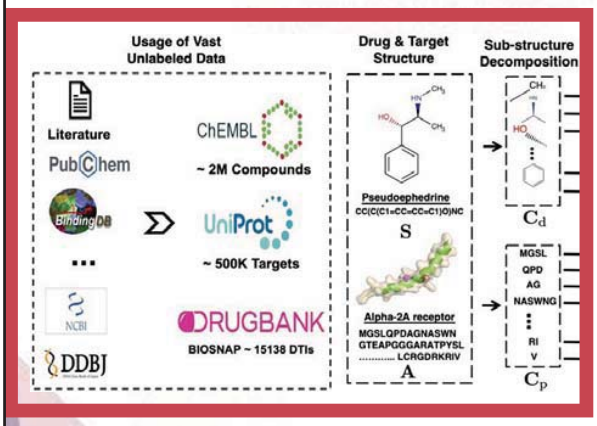
◦ ChEMBL, UniProt에 등록된 데이터로부터 빈도가 높은 substructure vocab 구성

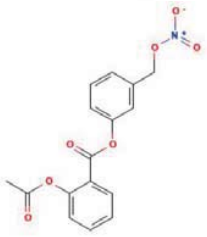
ChEMBL ~2M SMILES strings

UniProt ~500K target sequence

Methods (1) - Tokenization

◦ Frequent Consecutive Sub-sequence (FCS) mining for Substructure Decomposition





Nitroaspirin

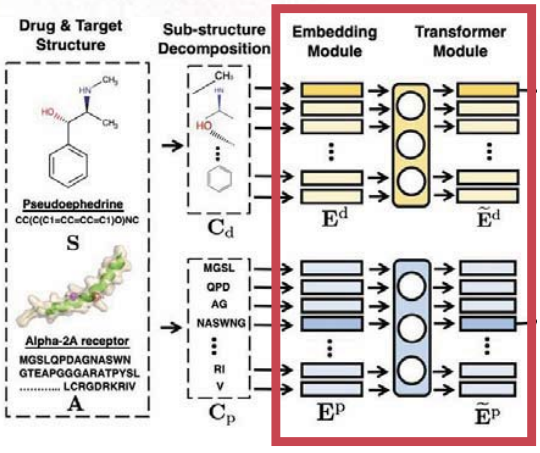
CC(=O)OC1=CC=CC=C1C(=O)OC2=CC=CC(=C2)CO[N+](=O)[O-]

CC(=O)OC1
=CC=CC
=C1
C(=O)OC
2=CC=CC
(=C2)C
O[N+](=O)[O-]

$C_d/C_p = \{C_1, \ldots, C_m\}$: drug/protein의 substructure sequence

Methods (2) – Transformer Encoder

◦ Augmented Transformer Embedding Module



◦ To capture relations among each sub-structure in the input, leverage Transformer's self-attention mechanism. The resulting sub-structural embedding is better because it is contextual by taking account into the **complex chemical relationships among the neighboring sub-structures**.

Methods (2) – Transformer Encoder

◦ Augmented Transformer Embedding Module

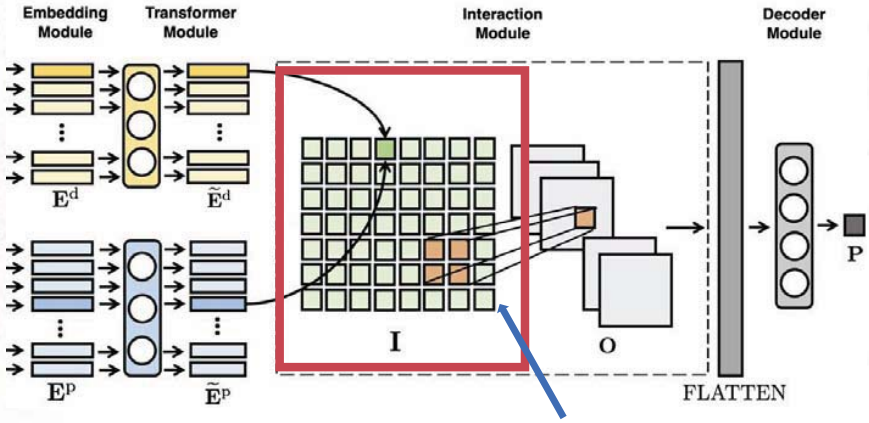
- C_p, C_d 를 두 Matrix $M^p \in \{0,1\}^{k \times \Theta_p}$ / $M^d \in \{0,1\}^{l \times \Theta_d}$ 로 변환
  - k, l : drug / protein에 대한 substructure의 전체 크기 (= Vocabulary set $\mathbb{V}$ 크기)
  - Θ_p, Θ_d : drug / protein에 대한 substructure sequence의 최대 길이 (=max sentence length)
  - M_i^p, M_j^d : protein / drug sequence의 i, j번째 substructure 에 대한 index (one-hot vector)

- $E^p_{cont_i} = W^p_{cont} M^p_i$, $E^d_{cont_j} = W^d_{cont} M^d_j$: content embedding for protein / drug
- $E^p_{pos_i} = W^p_{pos} \mathbb{I}^p_i$, $E^d_{pos_j} = W^d_{pos} \mathbb{I}^d_j$: positional embedding for protein / drug
  - $\mathbb{I}^p_i \in \mathbb{R}^{\Theta_p}, \mathbb{I}^d_j \in \mathbb{R}^{\Theta_d}$: single one-hot vector (i / j 번째 position = 1)

- $\boldsymbol{E^p_i = E^p_{cont_i} + E^p_{pos_i}}$, $\boldsymbol{E^d_j = E^d_{cont_j} + E^d_{pos_j}}$
- $\boldsymbol{\tilde{E}^p = \mathbf{Transformer}_{\mathbf{Protein}}(E^p)}$, $\boldsymbol{\tilde{E}^d = \mathbf{Transformer}_{\mathbf{Drug}}(E^d)}$

Methods (3) - CNN

◦ Interaction Module – pairwise interaction



$$\boldsymbol{I_{i,j} = F(\tilde{E}^p_i, \tilde{E}^d_j)}$$

Each position corresponds to an interaction intensity between a drug and a target sub-structure

◦ Motivated by the fact that DTI happens in sub-structural level

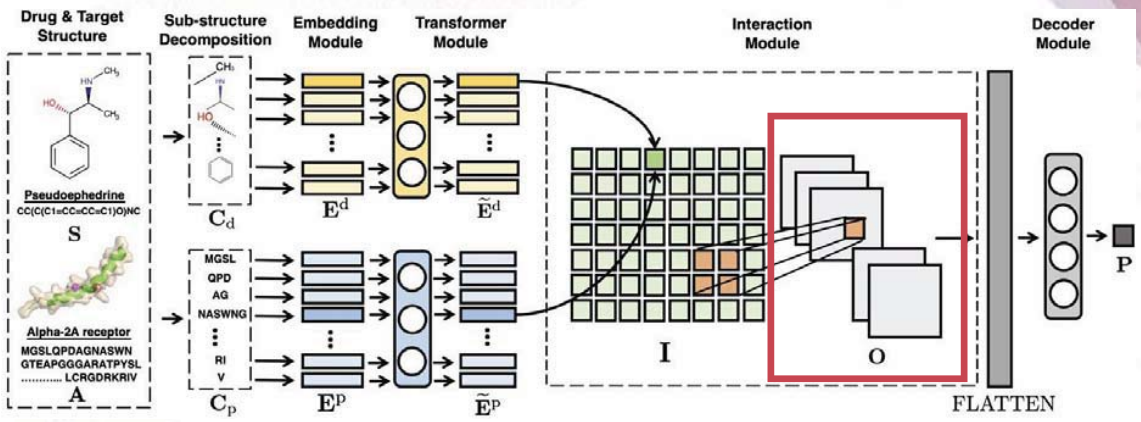
◦ 각 substructure i, j 쌍에 대한 interaction 측정

◦ $\boldsymbol{I_{i,j} = F(\tilde{E}^p_i, \tilde{E}^d_j)}$: protein, drug의 각 substructure i, j 에 대해 function F 적용

  ◦ Function F is can be any function as sum, average, dot product

Methods (3) - CNN

◦ Interaction Module – neighborhood interaction



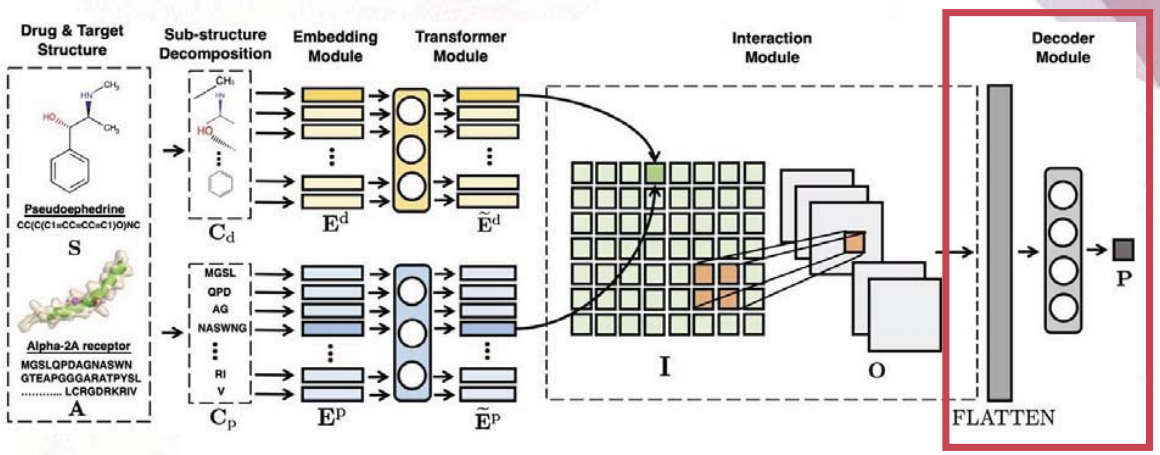
◦ $\boldsymbol{O} = \boldsymbol{CNN}(\boldsymbol{I})$

◦ Nearby sub-structure of proteins and drugs also influence each other in triggering the interactions

◦ substructure의 neighborhood interaction 반영

Methods (4) - Decoder

◦ Interaction Prediction Module



◦ By flattening the CNN output, generate an embedding for the DTI pair

◦ The embedding is fed into a decoder for prediction

◦ $\boldsymbol{P} = \sigma(\boldsymbol{W_O} FLATTEN(\boldsymbol{O}) + \boldsymbol{b_O}))$: interaction probability

◦ $\boldsymbol{Loss} = \boldsymbol{Y} log(\boldsymbol{P}) + (1 - \boldsymbol{Y}) \log(1 - \boldsymbol{P})$

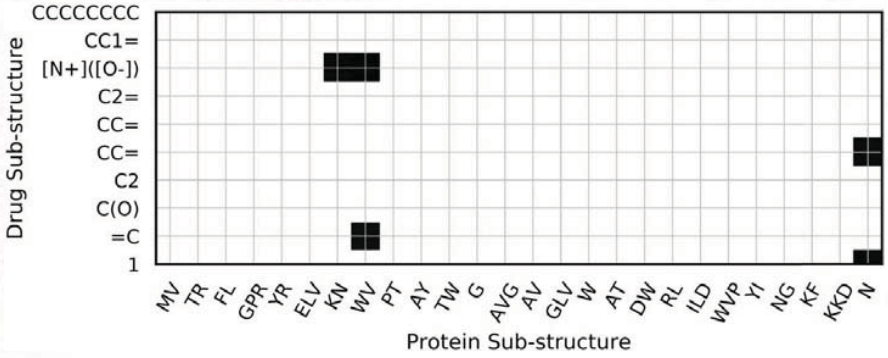
Results (1)

MolTrans has up to **25%** increase over best performing 2017 baseline

Method	ROC-AUC	PR-AUC	Sensitivity	Specificity	Threshold
Dataset 1: BIOSNAP					
LR	0.846 ± 0.004	0.850 ± 0.011	0.755 ± 0.039	0.800 ± 0.018	0.434
DNN	0.849 ± 0.003	0.855 ± 0.010	0.776 ± 0.040	0.838 ± 0.024	0.499
GNN-CPI	0.879 ± 0.007	0.890 ± 0.004	0.780 ± 0.014	0.819 ± 0.012	0.349
DeepDTI	0.876 ± 0.005	0.876 ± 0.006	**0.789±0.027**	0.845 ± 0.017	0.347
DeepDTA	0.876 ± 0.005	0.883 ± 0.006	0.781 ± 0.015	0.824 ± 0.012	0.466
DeepConv-DTI	0.883 ± 0.002	0.889 ± 0.005	0.770 ± 0.023	0.832 ± 0.016	0.441
MolTrans	**0.895 ± 0.002**	**0.901 ± 0.004**	0.775 ± 0.032	**0.851 ± 0.014**	0.431
Dataset 2: DAVIS					
LR	0.835 ± 0.010	0.232 ± 0.023	0.699 ± 0.051	0.842 ± 0.033	0.399
DNN	0.864 ± 0.009	0.258 ± 0.024	0.764 ± 0.045	0.860 ± 0.038	0.489
GNN-CPI	0.840 ± 0.012	0.269 ± 0.020	0.696 ± 0.047	0.842 ± 0.039	0.487
DeepDTI	0.861 ± 0.002	0.231 ± 0.006	0.751 ± 0.015	0.853 ± 0.012	0.387
DeepDTA	0.880 ± 0.007	0.302 ± 0.044	0.764 ± 0.045	0.865 ± 0.020	0.482
DeepConv-DTI	0.884 ± 0.008	0.299 ± 0.039	0.754 ± 0.040	**0.880 ± 0.024**	0.438
MolTrans	**0.907 ± 0.002**	**0.404 ± 0.016**	**0.800 ± 0.022**	0.876 ± 0.013	0.447
Dataset 3: BindingDB					
LR	0.887 ± 0.002	0.557 ± 0.015	0.741 ± 0.013	0.896 ± 0.011	0.394
DNN	0.908 ± 0.003	0.613 ± 0.015	0.769 ± 0.028	0.914 ± 0.021	0.371
GNN-CPI	0.900 ± 0.004	0.578 ± 0.015	0.754 ± 0.015	0.903 ± 0.011	0.406
DeepDTI	0.844 ± 0.002	0.429 ± 0.005	0.651 ± 0.024	0.895 ± 0.023	0.060
DeepDTA	0.913 ± 0.003	0.622 ± 0.012	0.780 ± 0.035	**0.915 ± 0.016**	0.305
DeepConv-DTI	0.908 ± 0.004	0.611 ± 0.015	0.781 ± 0.015	0.905 ± 0.013	0.318
MolTrans	**0.914 ± 0.001**	**0.622 ± 0.007**	**0.797 ± 0.005**	0.896 ± 0.007	0.355

Results (2)

◦ 2-nonyl n-oxide약물과 cytochrome b-c1단백질의 substructure간 interaction map



◦ $\boldsymbol{I = dot\ product}(\tilde{E}^p, \tilde{E}^d)$로 구성된 interaction map value중 일정 threshold 넘는 부분 highlight

◦ Nitrogen oxide group [N+](O-])와 KNWV이 높은 interaction value를 가짐

◦ Nitrogen oxide group이 cytochrome을 억제할 수 있음이 증명된 이전 연구 결과와 일치함

(JW Lightbown, FL Jackson. "Inhibition of cytochrome systems of heart muscle and certain bacteria by the antagonists of dihydrostreptomycin: 2-alkyl-4-hydroxyquinoline N-oxides." (1956))

Practice

◦ Colab Link: https://shorturl.at/hvwGQ

References

- Vaswani, Ashish, et al. Attention is all you need." Advances in neural information processing systems 30 (2017).
- Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).
- Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).
- Coursera, Convolutional neural Networks, DeepLearning.AI
- Alammar, J (2018). The Illustrated Transformer [Blog post]. Retrieved from https://jalammar.github.io/illustrated-transformer/
- Huang, Xiao, et al. "MolTrans: Molecular Interaction Transformer for drug-target interaction prediction." Bioinformatics (2021).