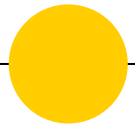
Neuronske mreže

Transformer model

Agenda

- *Attention Mechanism*
- Transformer model
- *BERT*
- Transformer *fine-tuning*
- Dodatno čitanje

Attention Mechanism



Attention Mechanism

- *Transfer learning* u *NLP*-u:
  - Da li je moguće napraviti univerzalan *Language model* koji bi se mogao koristiti u raznim *NLP* (*Natural Language Processing*) zadacima (kao što koristimo pretrenirane *VGGNet, ResNet, …*)?
  - Više pokušaja istraživanja u ovom smeru.

Attention Mechanism

- *RNN* - Primer klasifikacije sekvence (podsetnik):

Tokenizacija rečenice na reči

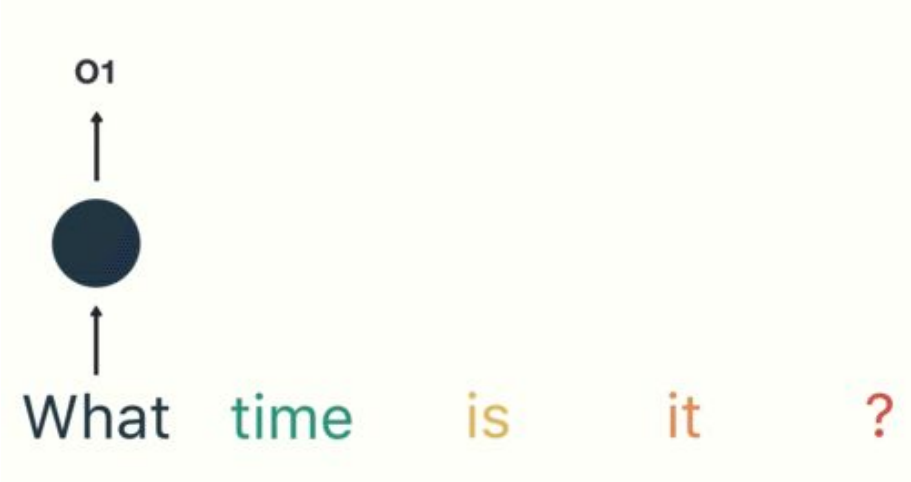
Attention Mechanism

- *RNN* - Primer klasifikacije sekvence (podsetnik):

Dovođenje prve reči na *RNN*. *RNN* kodira ulaz i vraća rezultat.

Attention Mechanism

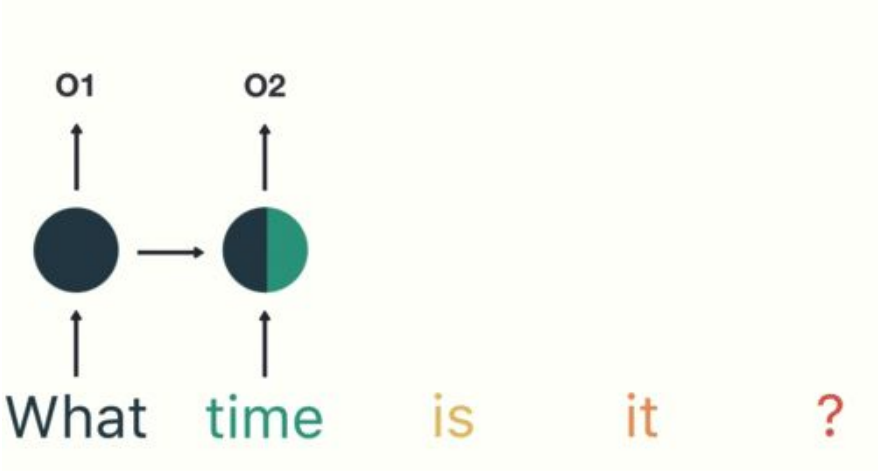
- *RNN* - Primer klasifikacije sekvence (podsetnik):



Dovođenje druge reči na *RNN*. Pored druge reči, dovodi se i skriveno stanje iz prethodnog koraka. *RNN* ima transformaciju i *“What”* i *“time”*.

Attention Mechanism

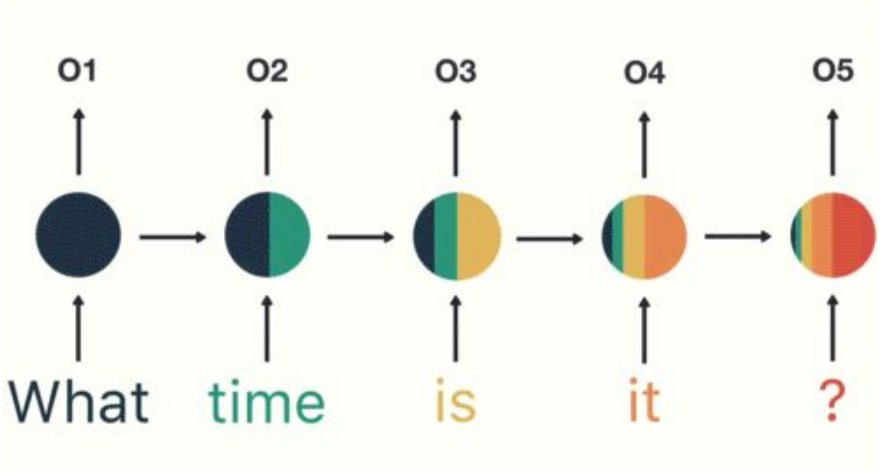
- Primer klasifikacije sekvence:



Proces se ponavlja do kraja sekvence. Rezultat je kodiranje cele ulazne sekvence reči.

Attention Mechanism

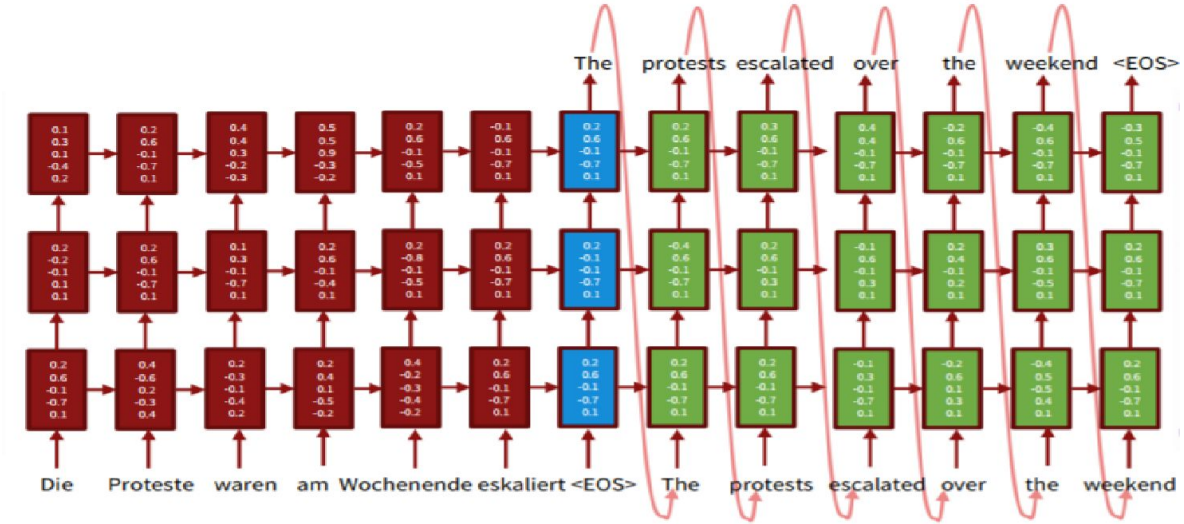
- *RNN* - Primer klasifikacije sekvence (podsetnik):



Pošto vektor **O5** sadrži kodirane informacije o celoj sekvenci, njega možemo poslati na neki klasifikator (recimo običan MLP) da bi izvršili klasifikaciju sekvence.

Attention Mechanism

- *Sequence To Sequence* model:



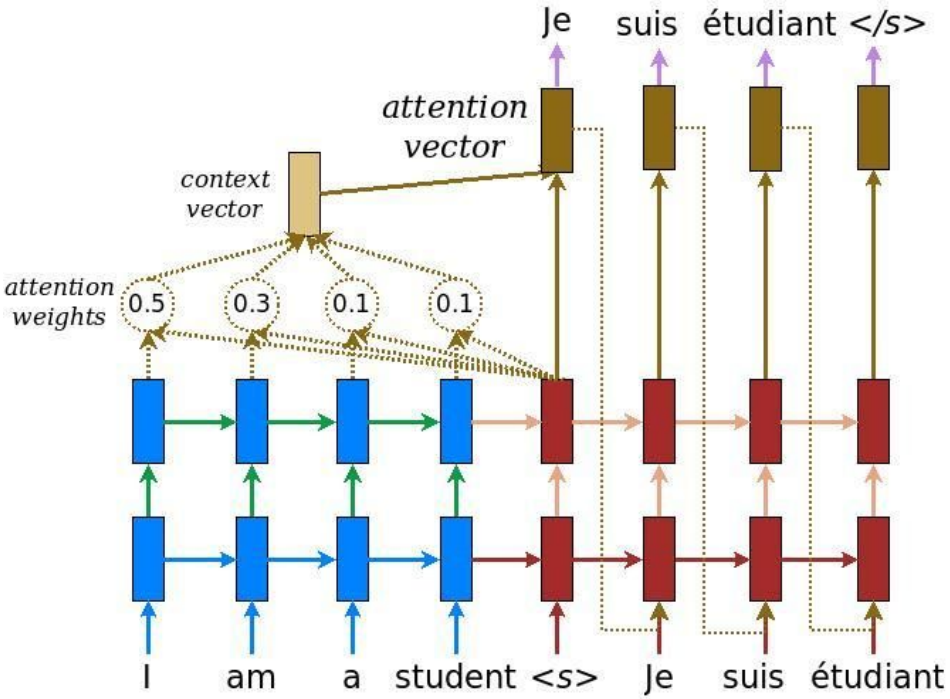
Izlaz enkodera se koristi kao ulaz u dekoder.

Da li je iz tog vektora lako izvući informaciju koja reč iz jezika A najviše utiče na reč iz jezika B (kod primera *NMT*)?

Ne. Dekoder nema jasnu sliku koja reč utiče na koju reč i generisaće kako je obučen.

Attention Mechanism

- *Attention Mechanism:*
  - Jednostavan mehanizam koji se ubacuje u prostor između enkodera i dekodera
  - Ovo dekoderu omogućava da uhvati više “globalnih” informacija, a ne samo poslednje skriveno stanje iz enkodera.



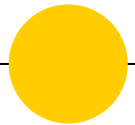
Attention Mechanism

- Kombinacija *RNN + Attention Mechanism* daje dobre rezultate:
  - Obično dosta bolje u poređenju sa običnom RNN u enkoder-dekoder arhitekturi
- Međutim, računanje *attention*-a je skupo zbog velikog broja težina koje se takođe podešavaju u procesu obučavanja *RNN* (koji je po sebi skup)

Attention Mechanism

- Da li nam je uopšte potrebna *RNN* komponenta, ili se može računati i samo *attention* na osnovu *embedding* vektora reči?

Transformer model



Transformer model

- Prvi put predstavljen u junu 2017. od strane *Google Machine Translation* tima
- Namenjen za *language understanding* i bazira se na upotrebi *attention* mehanizma, bez *RNN*

Transformer model

Transformer model

- Postigao najviši skor u mašinskom prevođenju (bolji i od *RNN* i *CNN* pristupa)
- Umesto da čita tekst sa leva na desno (i/ili obrnuto), kao što rade *RNN* kojima je nekada potrebno dosta reči da bi donele odluku, transformer to radi u mnogo manjem broju koraka

Transformer model

- Npr.: *I arrived at the bank after crossing the river.*
  - *RNN* bi bilo potrebno da pročita sve reči između reči *bank* i kraja rečenice, da bi shvatila da je u pitanju obala reke, a ne banka
  - Da bi izračunao reprezentaciju reči *bank*, transformer poredi sve reči sa rečju *bank* i računa *attention score*
    - Ovime se dobija uticaj svake reči iz konteksta na reč *bank* (*river* će imati visok *attention*)
    - Ovi *attention scores* se nakon toga koriste kao težine za *weighted average*, koji se prosleđuje *FC* mreži, koja nakon toga daje reprezentaciju reči *bank*.

Transformer model

- Kao i *RNN/LSTM*, transformer je arhitektura koja pretvara jednu sekvencu u drugu sekvencu uz pomoć dva dela (enkodera i dekodera), ali se razlikuje po tome što nema rekurentnih veza unutar sebe
- Da bi modelovao sekvencu reči (gde je redosled reči bitan), transformer koristi tzv. pozicioni enkoding (eng. *positional encoding*).

Transformer model

- Transformer modeli koriste tzv. *self-attention*, gde se reprezentacija reči uči na osnovu vrednosti *attention*-a u odnosu na druge reči u tom kontekstu
  - Na taj način se hvataju:
    - Zavisnosti od konteksta
    - Sličnost pojedinačnih reči
    - Međusobni uticaji
    - I slično.

Encoder

Decoder

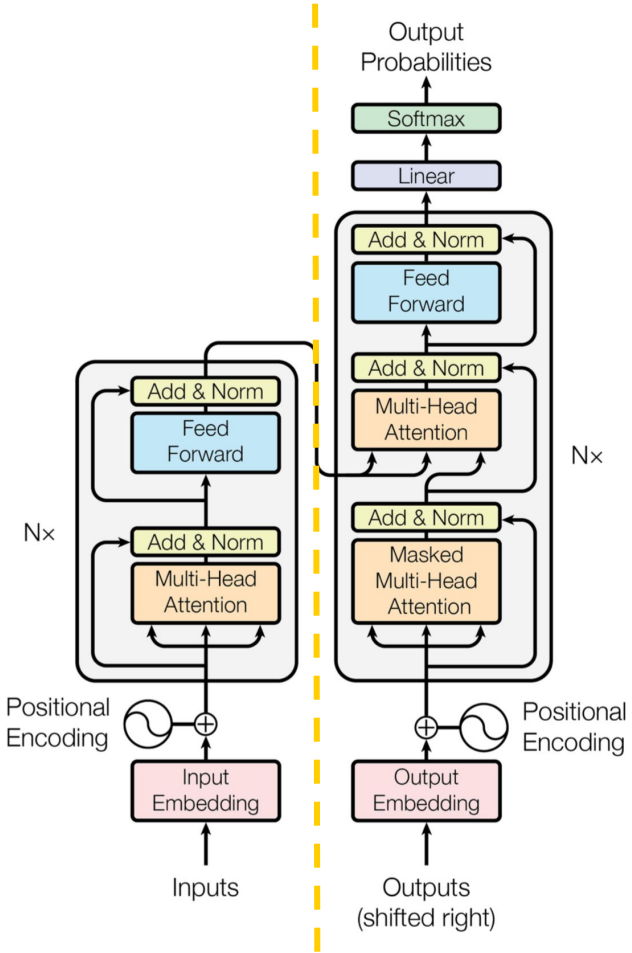
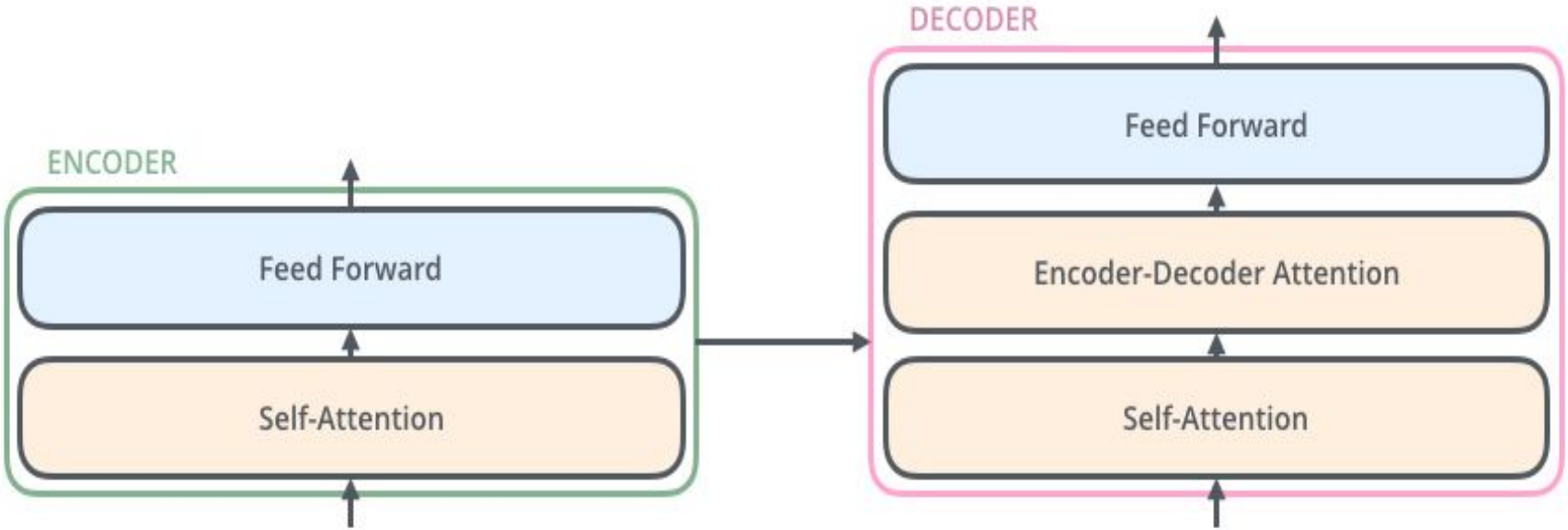


Figure 1: The Transformer - model architecture.

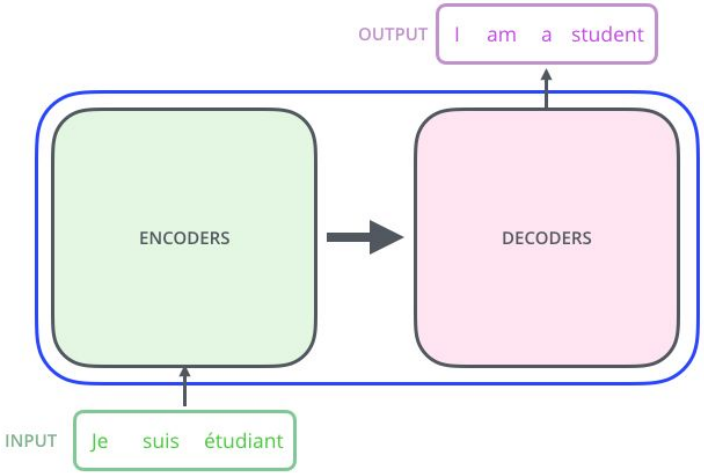

DECODER
Feed Forward
Encoder-Decoder Attention
Self-Attention
ENCODER
Feed Forward
Self-Attention

Transformer model

- Primer mašinskog prevođenja:
  - Sa najvišeg nivoa apstrakcije, model bi izgledao ovako:

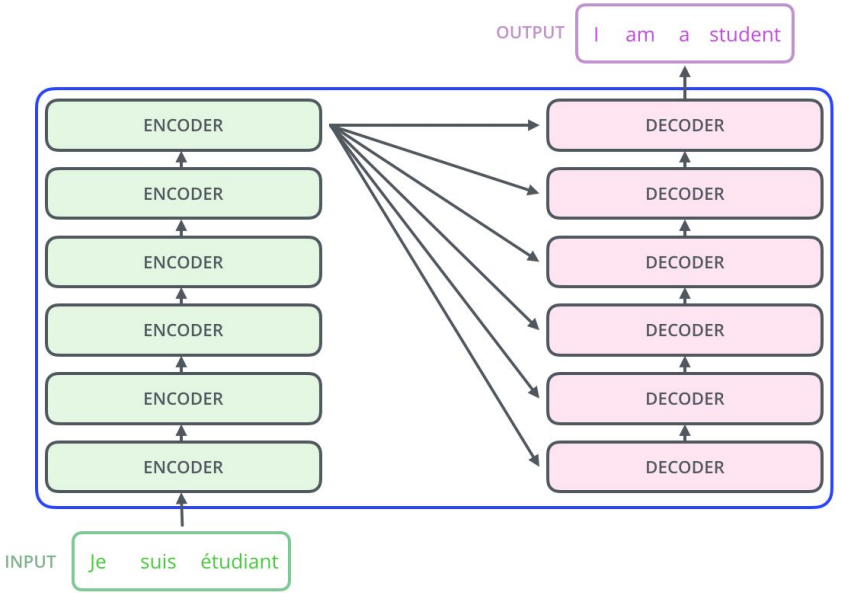
Transformer model

- Primer mašinskog prevođenja:
  - Ulaskom dublje, vidimo da je to enkoder-dekoder arhitektura koja nam je već poznata



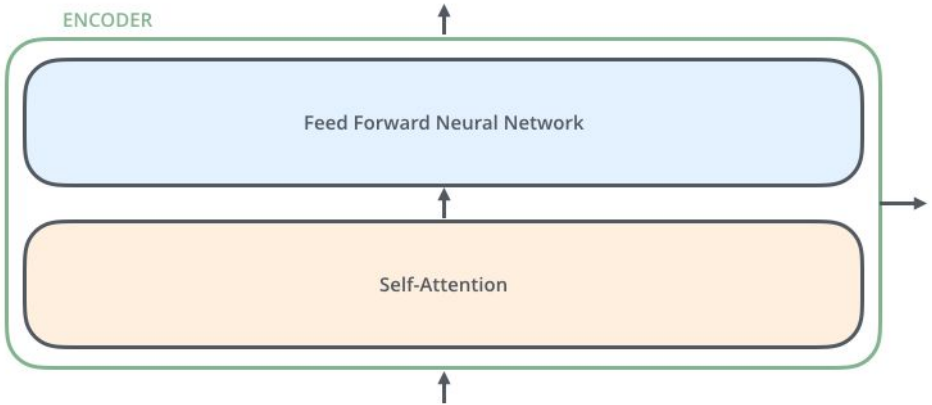
Transformer model

- Primer mašinskog prevođenja:
  - Ulaskom dublje, vidimo da je to enkoder-dekoder arhitektura koja nam je već poznata



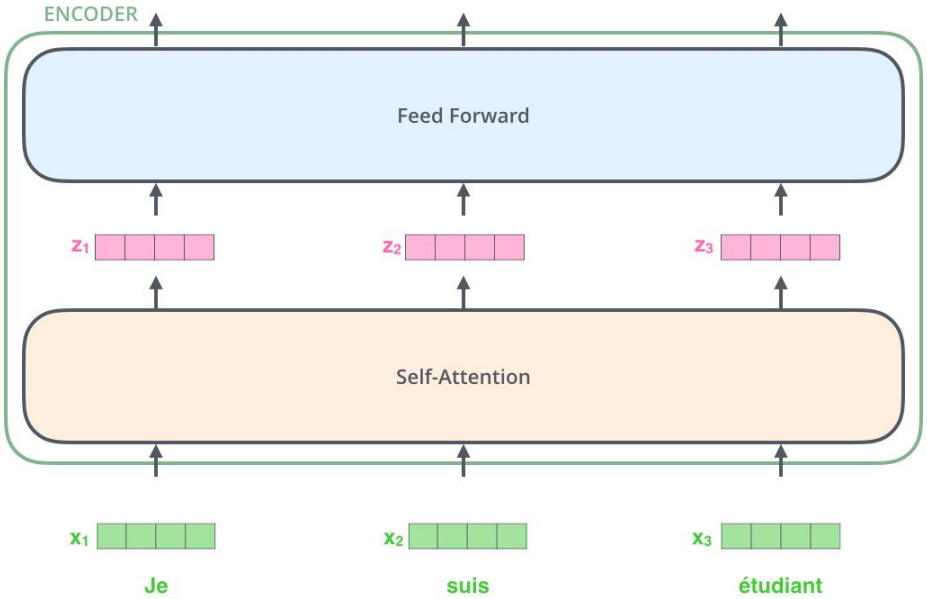
Transformer model

- Primer mašinskog prevođenja:
  - Od čega se sastoji enkoder?



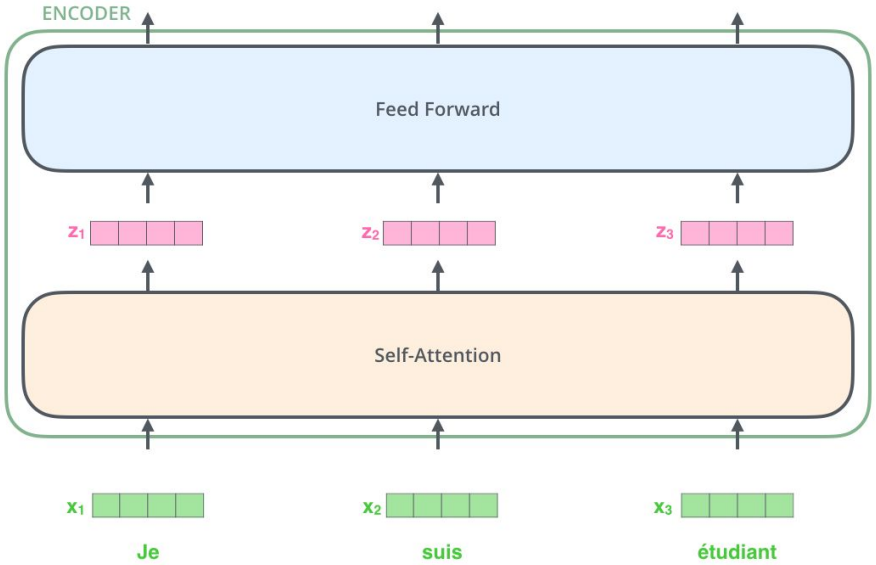
Transformer model

- Primer mašinskog prevođenja:
  - Od čega se sastoji enkoder?



Transformer model

Feed forward sloj se trenira da bi se napravio *bias* ka konkretnom modelu i da problem ne bude generičko računanje *self-attention* vektora za svaku reč.



Enkoder u prvom sloju mreže prima vektore reči (*word embedding*), a svi ostali primaju izlaz iz prethodnog enkodera u steku. Enkoderi ne dele težine.

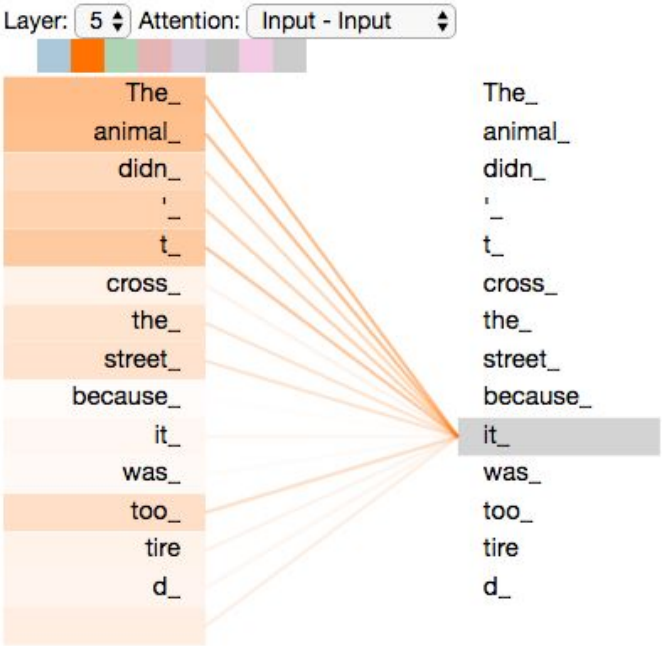
Enkoder se trenira za predefinisanu dužinu ulaznih rečenica (fiksan broj reči).

Svaka reč ima svoju putanju kroz enkoder.

Transformer model

- *Self-attention* predstavlja težinsku kombinaciju (eng. *Weighted combination)* svih *word embedding*-a u datom kontekstu, uključujući i reči koje se pojavljuju pre i reči koje se pojavljuju posle u rečenici, a kao težine se uzimaju *attention* koeficijenti.

Transformer model



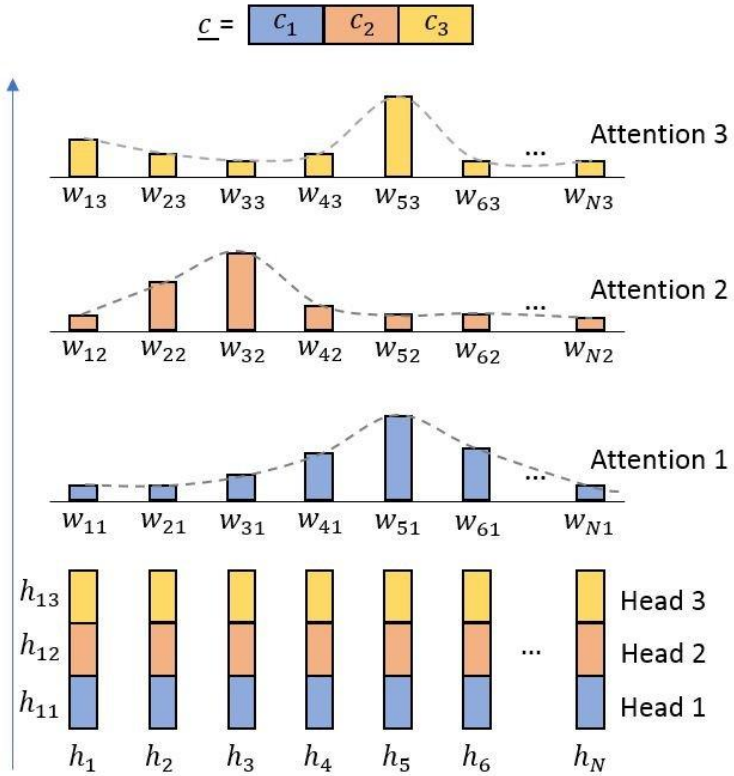
Intuitivno, omogućava modelu da predstavi datu reč tako što će gledati kako je sam kontekst formira i kako druge reči utiču na nju.

Kada se napravi više slojeva ovakvih *softmax attention* vektora (koji se množe sa *word embedding*-om) dobija se jako moćna hijerarhijska reprezentacija međusobnih veza i uticaja unutar teksta, pa model ima vrlo moćan način razumevanja istog.

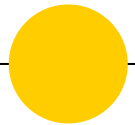
Transformer model

Inicijalni rad o transformer modelima uvodi i pojam *multi-headed attention* mehanizma:

- Ideja jeste da se računanje *attention*-a dešava više puta, tako da model može da nauči različite semantički interesantne informacije o tekstu kroz različite kanale/glave (eng. *heads*) kao što su recimo *short-term* ili *long-term* zavisnosti u tekstu.
- Ovo računanje će se obavljati u paraleli za svaku glavu odvojeno, a nakon toga se ovi vektori konkateniraju u jednu reprezentaciju. Tako se dobija rezultat kao da je u pitanju jedna glava, ali je ovaj put reprezentacija mnogo bogatija.
- Uvođenje *multi-headed attention* mehanizma značajno povećava matricu težina, pa će model imati i mnogo više parametara.

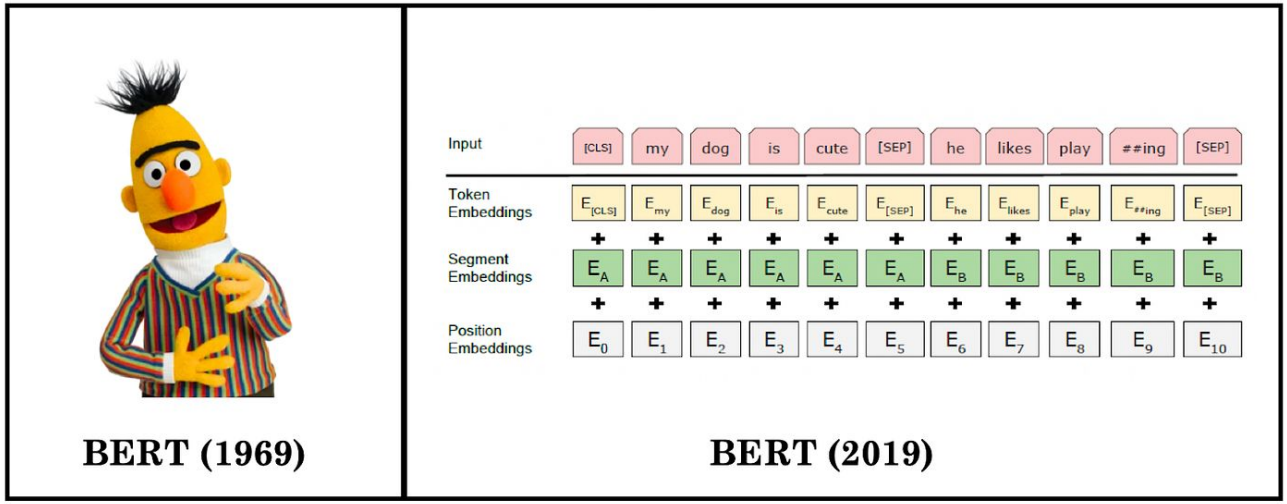


BERT



BERT

- Objavljen 11.10.2018. od strane *Google AI Language* tima



BERT

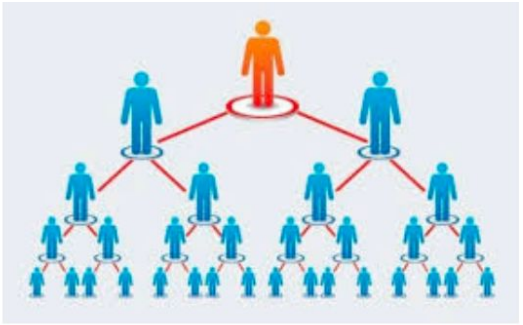
- *Bidirectional Encoder Representation from Transformers*
- *State-of-the-art (SOTA) language model for NLP*
- Izazvao lavinu u primeni MU na *NLP*, pošto je postigao *SOTA* performanse u velikom broju *NLP* problema, uključujući i *Question Answering* gde radi bolje od čoveka.

BERT

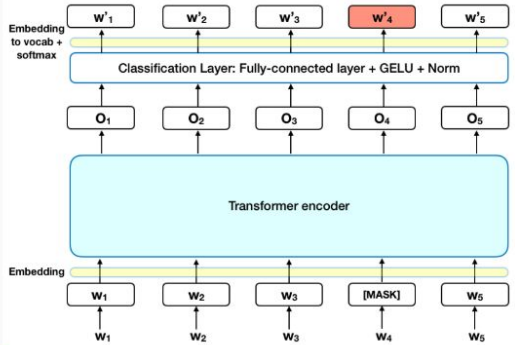
- Predložen za *language model* nad kojim se može raditi *fine-tuning* za razne primene u domenu *NLP*
- Inicijalni *BERT* modeli su trenirani korišćenjem dve tehnike (trenirani su za dva zadatka istovremeno):
  - Maskiranje reči
  - Predikcija da li jedna rečenica sledi iza druge

BERT

- Maskiranje reči:
  - *MLM*



Multi Level Marketing



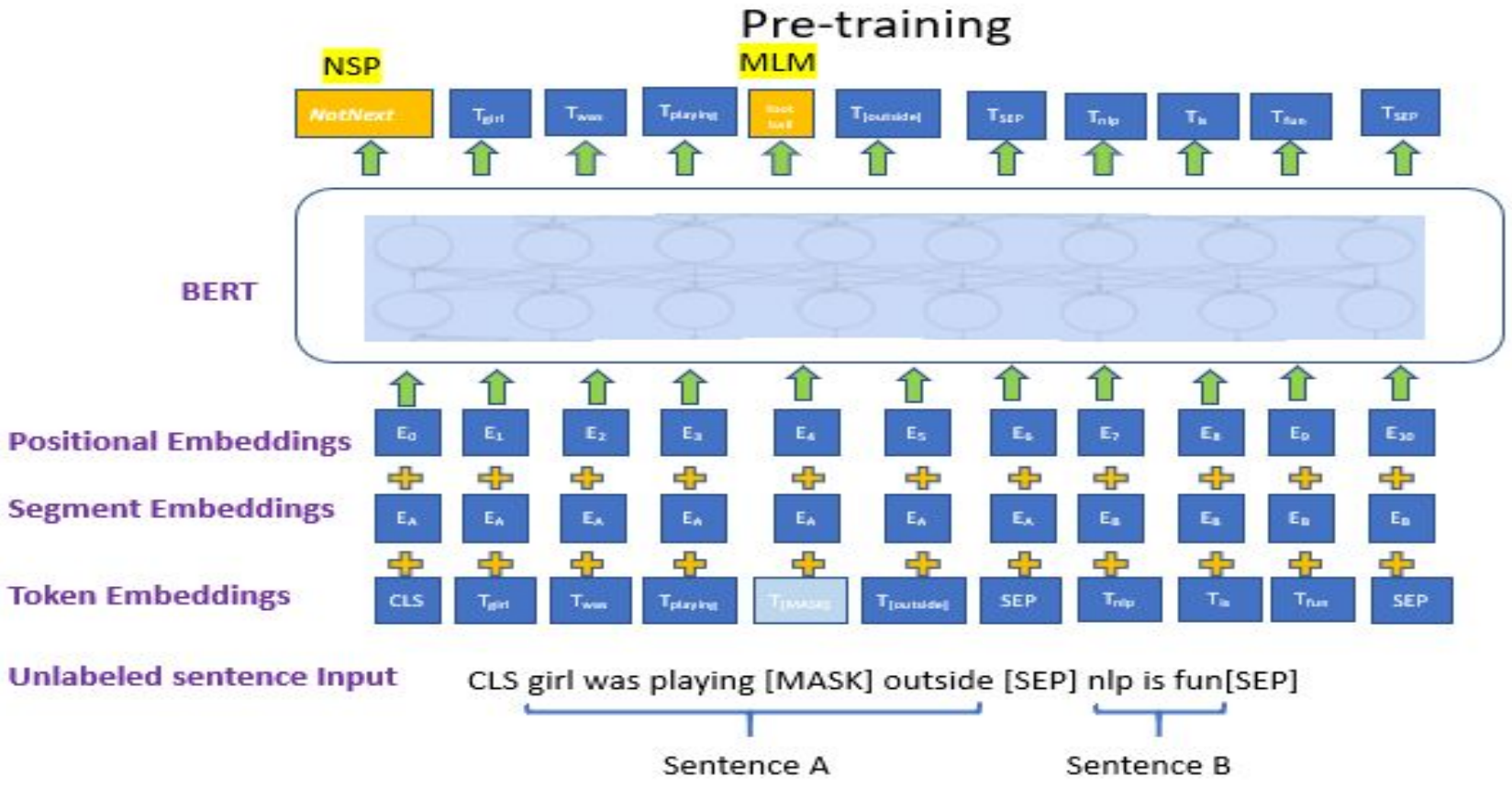
Masked Language Model

BERT

- Maskiranje reči:
  - *MLM - Masked Language Modeling*
  - Ideja je da model pogodi koja se reč nalazi pod maskom
  - Za ovo je potrebno duboko razumevanje konteksta i model koji ovo može je jako moćan u razumevanju jezika

BERT

- Predikcija da li jedna rečenica sledi iza druge:
  - *NSP - Next Sentence Prediction*
  - Model treba da predvidi da li rečenica B sledi iza rečenice A, ili su njih dve potpuno nezavisne (iz različitih konteksta)


Pre-training
NSP
MLM
NotNext
BERT
Positional Embeddings
Segment Embeddings
Token Embeddings
CLS
SEP
Unlabeled sentence Input
CLS girl was playing [MASK] outside [SEP] nlp is fun[SEP]
Sentence A
Sentence B

BERT

- Treniranje tehnikom maskiranja:
  - Maskira se oko 15% nasumično izabranih tokena sa ulaza:
    - U 80% od ovih 15% primeraka se reč menja tokenom *[MASK]*
    - U 10% od ovih 15% primeraka se reč menja drugim nasumičnim tokenom
      - Ovo će naterati model da generiše kontekstualni *embedding* za sve tokene u sekvenci, a ne samo za *[MASK]*
    - U ostalih 10% slučajeva se reč ni ne menja
      - Ideja iza ovoga je da se model *bias*-uje ka samoj reči koju pogađa. Sama reč se koristi za pogađanje same sebe na izlazu, čime se stvara *bias* ka tačno toj reči i poboljšavaju se performanse modela.

BERT

- Treniranje tehnikom maskiranja:
  - Prednost ove procedure je u tome što enkoder ne zna koje reči će od njega biti tražene za predikciju ili koje su reči zamenjene nasumičnim tokenima, pa je nateran da čuva distribuirane kontekstualne *embedding*-e svih ulaznih tokena
  - Takođe, pošto se zamena nasumičnih tokena događa samo u 1.5% svih tokena (10% od 15%), ovo ne narušava moć modela da razume jezik.

BERT

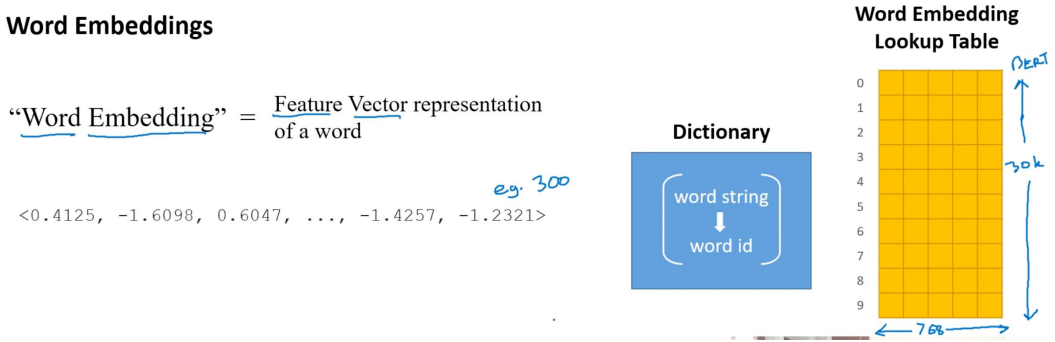
- *Embedding* ulaznih reči:
  - *BERT* enkodira ulaznu sekvencu kroz tri tipa *embedding*-a koje kombinuje:
    - *Word embedding*
      - Za svaku reč se dobija njen indeks u rečniku
    - *Sentence embedding*
      - Za svaku reč se dodaje indeks rečenice kojoj ona pripada
    - *Positional embedding*
      - Za svaku reč se dodaje njena relativna pozicija u odnosu na početak rečenice.

BERT

- *Word embedding*:
  - Rečnik je ograničen na 30k reči
    - Korišćen je *WordPiece* model najčešćih reči i prefiksa/sufiksa/slova
  - Zbog toga se nepostojeće reči suzbijaju na n-grame ili na pojedinačna slova ako n-grami ne postoje
    - Ove reči u sekvenci tokena imaju prefiks *“##”*

BERT

- *Word embedding*:



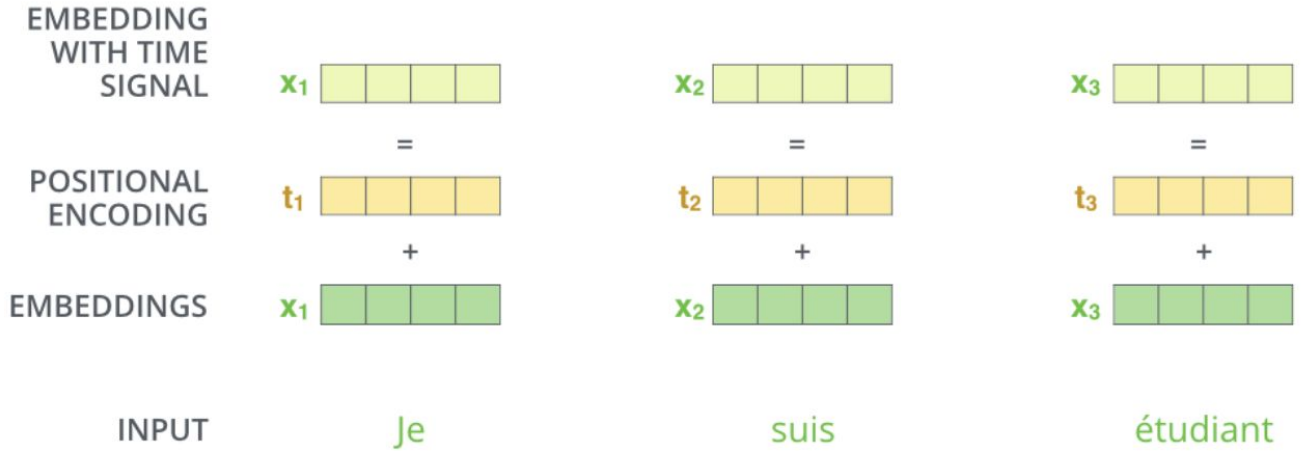
  - Ukoliko reč *"embeddings"* ne postoji u rečniku, *BERT tokenizer* će je predstaviti kao *['em', '##bed', '##ding', '##s']*

BERT

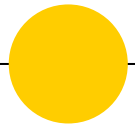
- *Positional embedding*:
  - Omogućava unošenje informacija o redosledu reči unutar sekvence
  - *RNN* je ovo radio automatski, ali se proces nije mogao paralelizovati, dok u ovom slučaju može i to je jedna od najvećih prednosti transformer modela u odnosu na *RNN*

BERT

- *Positional embedding*:



Transformer *fine-tuning*



Transformer *fine-tuning*

- Transformer modeli predstavljaju prve prave jezičke modele (eng. *Language models*) koji su pandan pretreniranim *CNN* (*VGG, ResNet,* …) u oblasti računarske vizije


@mat_MTO
ME, A RECENT GRAD WITH NO NLP EXPERIENCE
BERT
LEARNING FUNDAMENTAL TECHNIQUES THAT DONT REQUIRE EXPENSIVE HARDWARE

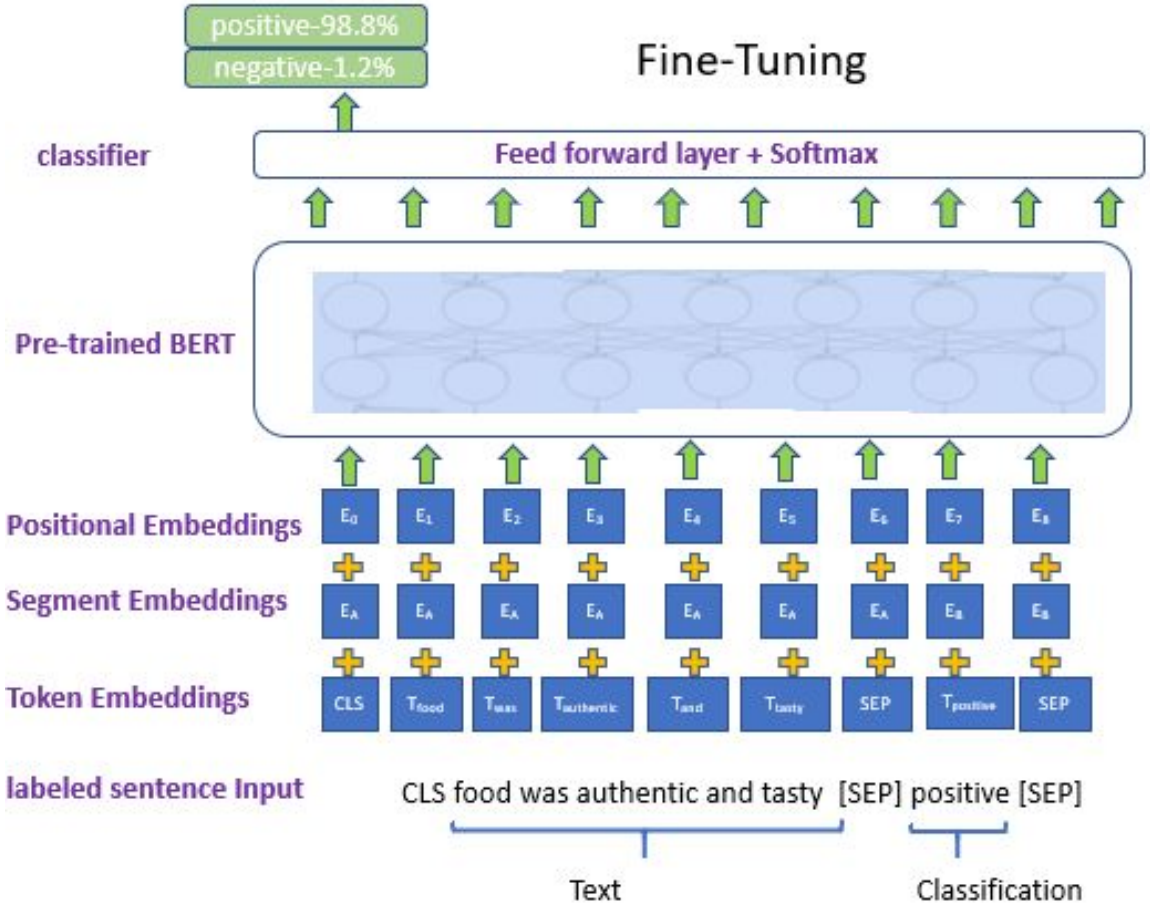

how many layers of NLP are you on
GloVe
like,, maybe word2vec or right now. my dude
this was supposed to be legi
you are like a little baby
watch this
FINE TUNE BERT

Transformer *fine-tuning*

- Pošto transformer modeli imaju milione/milijarde parametara, njihovo treniranje bi oduzimalo jako puno resursa (što zbog broja parametara, što zbog masivnosti skupova podataka za treniranje)
- Zbog toga se koristi *fine-tuning*, gde se uzme pretrenirani model, a onda se dotrenira samo dekoder deo

Transformer *fine-tuning*

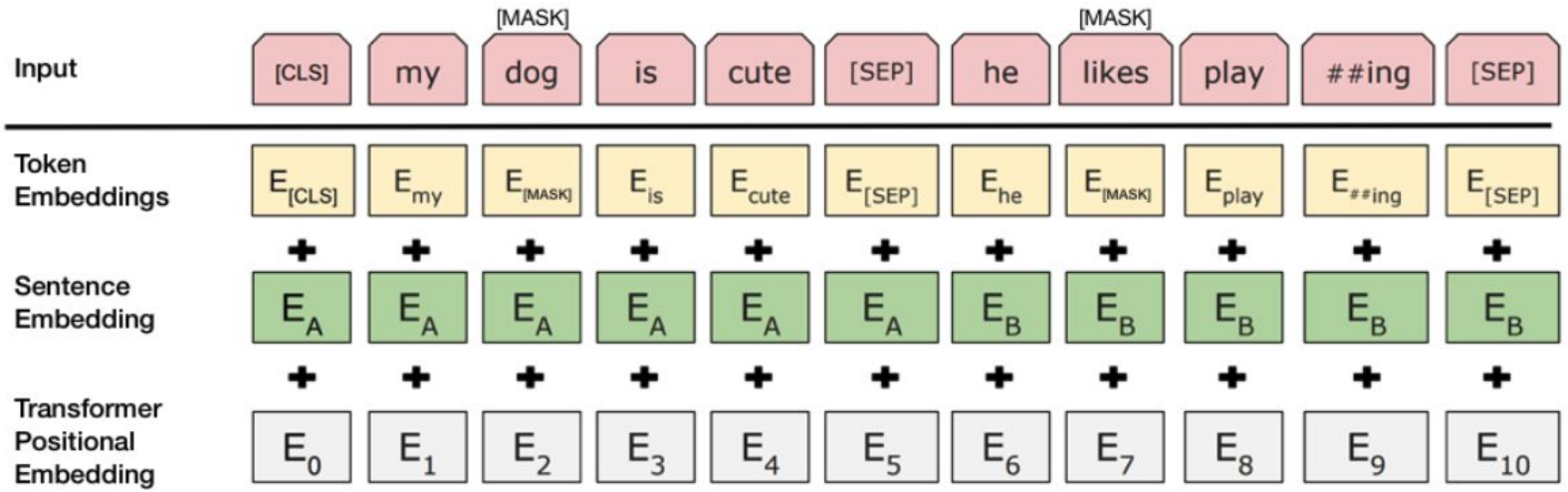
- Resursi:
  - *HuggingFace*
    - Zvanična dokumentacija
    - *GitHub* repozitorijum
    - Treniranje i *fine-tuning*
    - Modeli
  - *DeepPavlov*
    - *Wrapper* biblioteka za transformer i druge modele
    - Podrška za *BERT*
    - Demo


positive-98.8%
negative-1.2%
Fine-Tuning
classifier
Feed forward layer + Softmax
Pre-trained BERT
Positional Embeddings
Segment Embeddings
Token Embeddings
CLS
SEP
labeled sentence Input
CLS food was authentic and tasty [SEP] positive [SEP]
Text
Classification

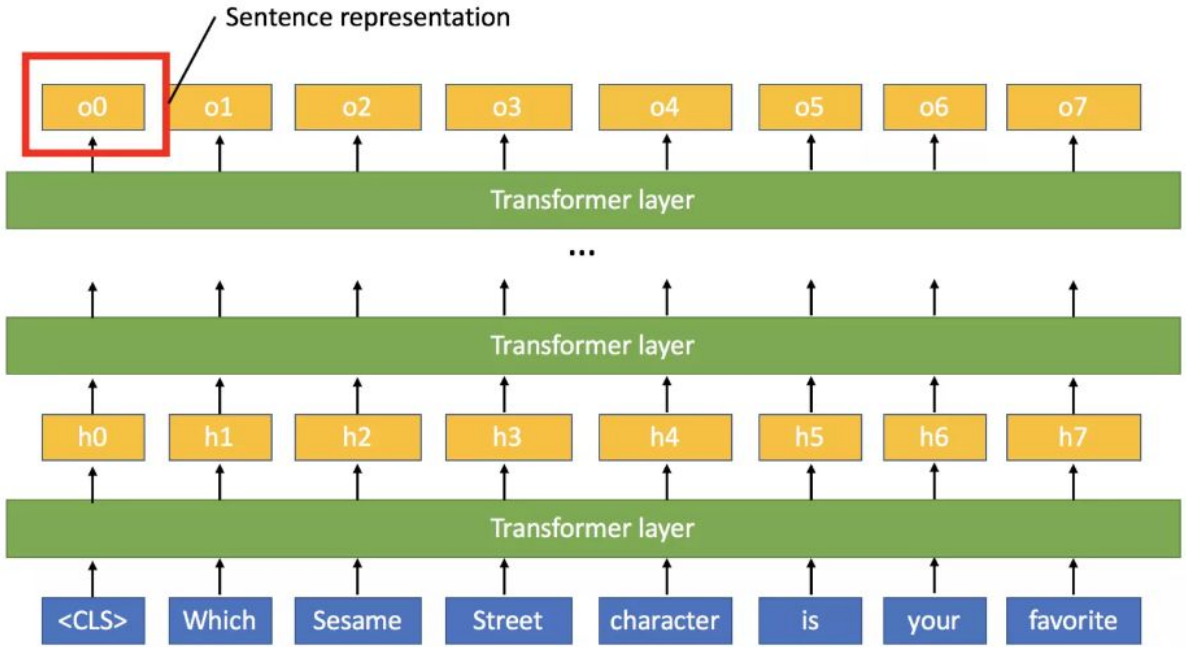
Transformer *fine-tuning*

- *BERT fine-tuning*:
  - A šta da radimo sa prvim tokenom iz sekvence *[CLS]* na ulazu?
    - Šta predstavlja njegov *embedding* na izlazu transformera?

Transformer *fine-tuning*



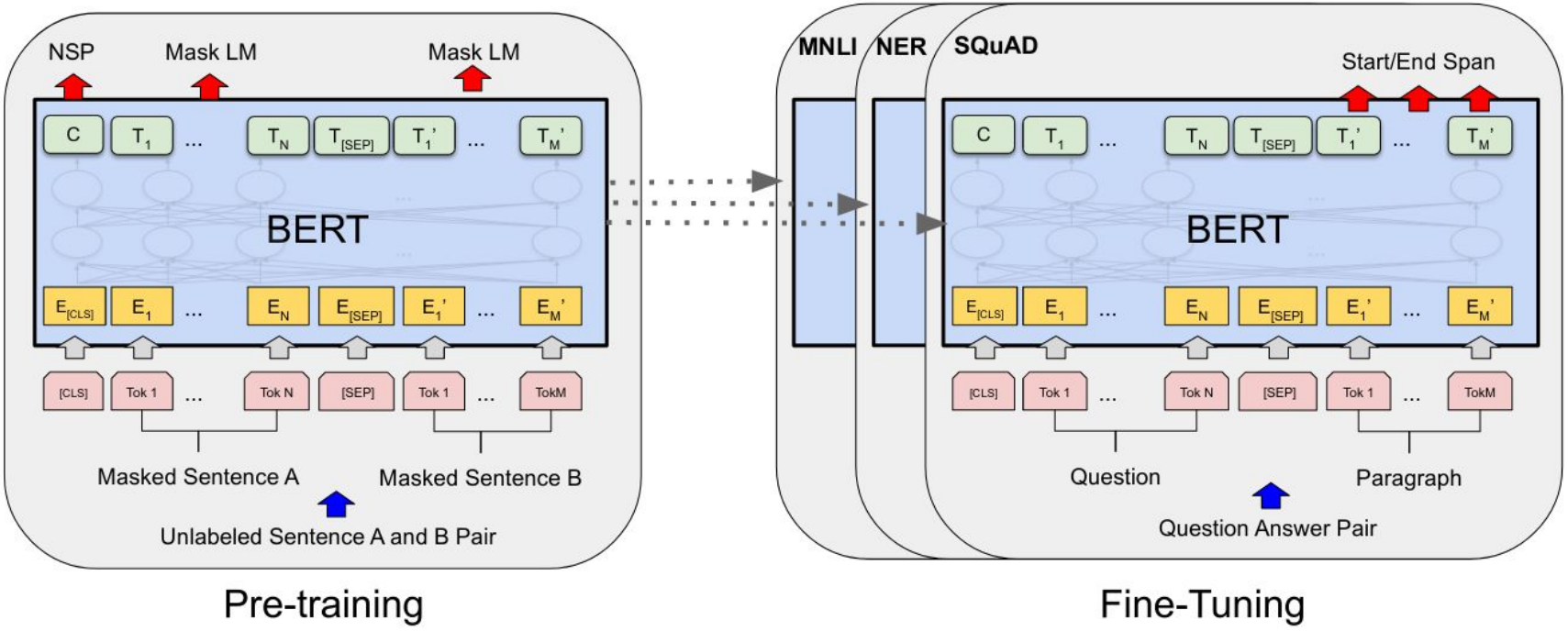
Transformer *fine-tuning*

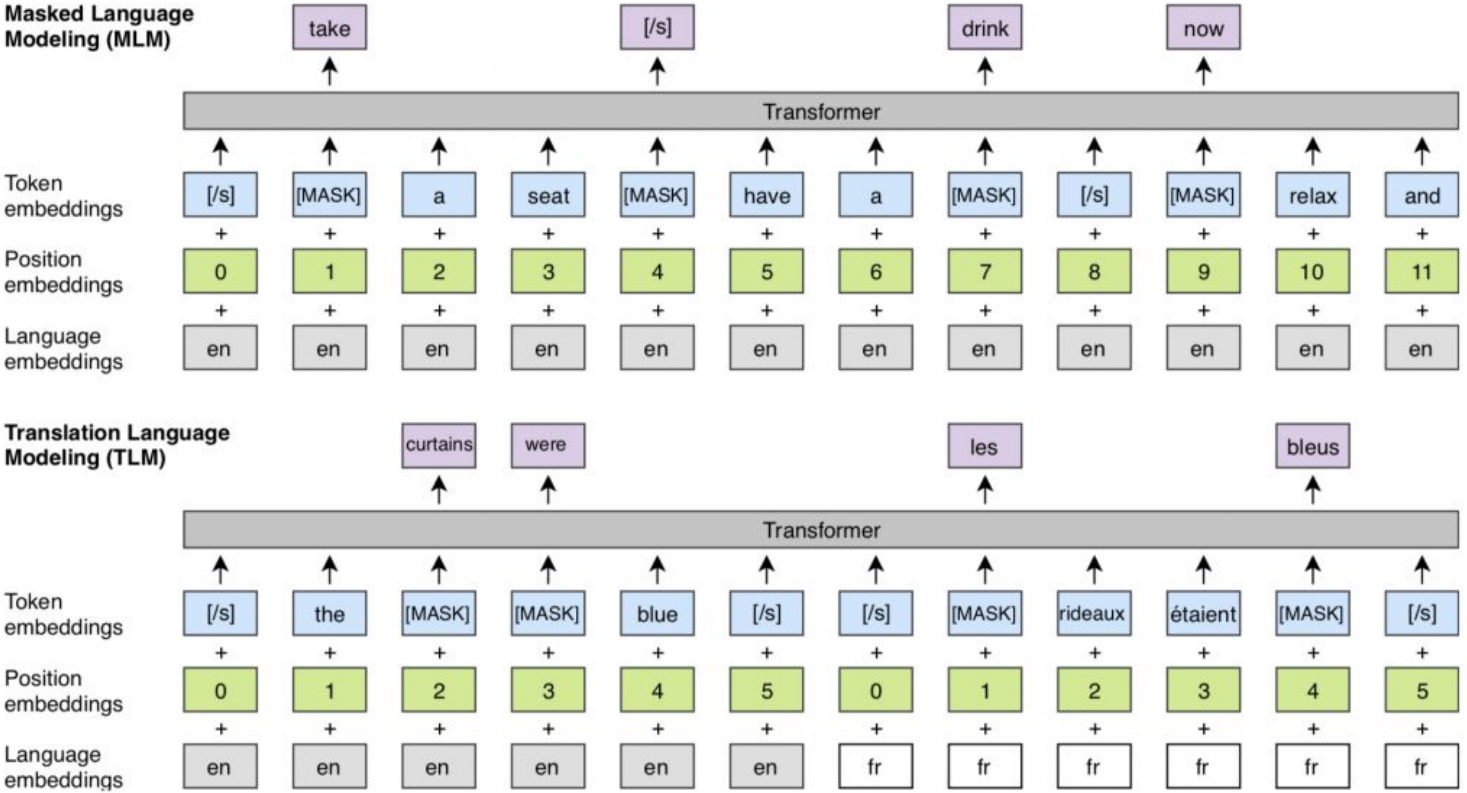


Često se u *fine-tuning* procedurama za klasifikaciju teksta koristi samo ovaj prvi *embedding* vektor.

U tom slučaju se on klasifikuje, a ne ceo izlaz.

Ovaj vektor sadrži *attention* kombinacije cele sekvence i jako je bogat semantikom.


NSP
Mask LM
Mask LM
C
T1
...
TN
T[SEP]
T1'
...
TM'
BERT
E[CLS]
E1
...
EN
E[SEP]
E1'
...
EM'
[CLS]
Tok 1
...
Tok N
[SEP]
Tok 1
...
TokM
Masked Sentence A
Masked Sentence B
Unlabeled Sentence A and B Pair
Pre-training
MNLI
NER
SQuAD
Start/End Span
Question
Paragraph
Question Answer Pair
Fine-Tuning


Masked Language Modeling (MLM)
take
[/s]
drink
now
Transformer
Token embeddings
[/s] [MASK] a seat [MASK] have a [MASK] [/s] [MASK] relax and
Position embeddings
0 1 2 3 4 5 6 7 8 9 10 11
Language embeddings
en en en en en en en en en en en en
Translation Language Modeling (TLM)
curtains
were
les
bleus
Transformer
Token embeddings
[/s] the [MASK] [MASK] blue [/s] [/s] [MASK] rideaux étaient [MASK] [/s]
Position embeddings
0 1 2 3 4 5 0 1 2 3 4 5
Language embeddings
en en en en en en fr fr fr fr fr fr

Transformer *fine-tuning*

- *BERT vs ELMo:*

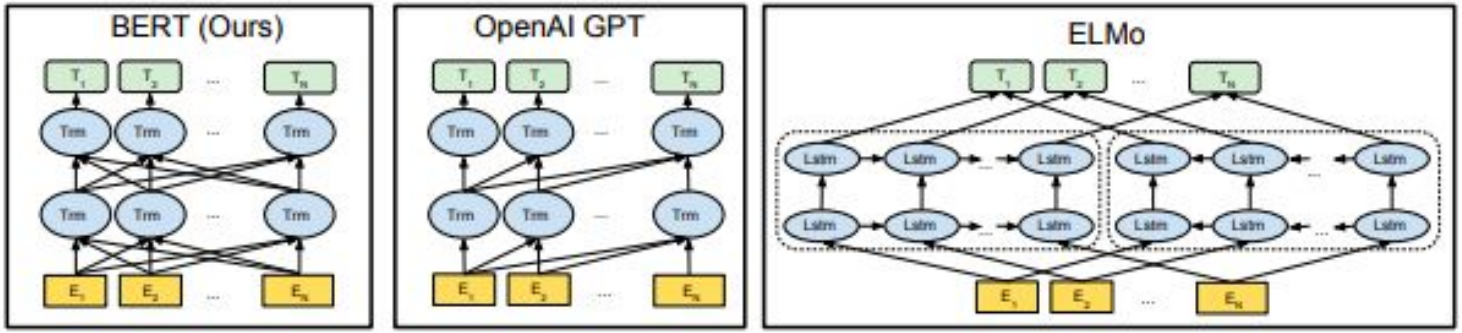
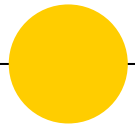


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

Transformer *fine-tuning*

- Primeri:
  - Primer klasifikacije teksta uz BERT
  - BERT fine-tuning

Dodatno čitanje



Dodatno čitanje

- *Attention Is All You Need*
- *A Gentle Introduction to Positional Encoding in Transformer Models*
- *The Illustrated Transformer*
- *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
- *BERT Explained: State of the art language model for NLP*

Hvala na pažnji!

Pitanja?