

# Moderní jazykové modely

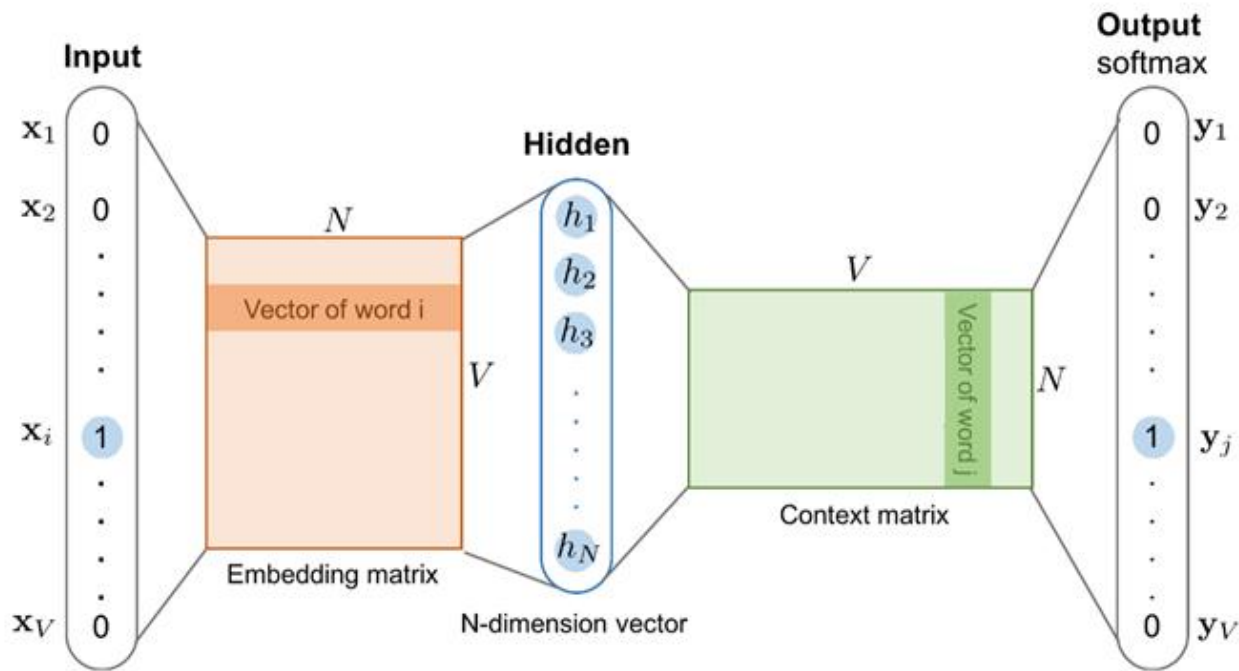
*František Kynych*  
5. 12. 2024 | MVD

# Moderní jazykové modely

- Založené na neuronových sítích
- Předchozí přednášky
  - Word2Vec, GloVe
- Tato přednáška se zaměřuje na komplexnější architektury
  - Rekurentní neuronové sítě
  - Transformer

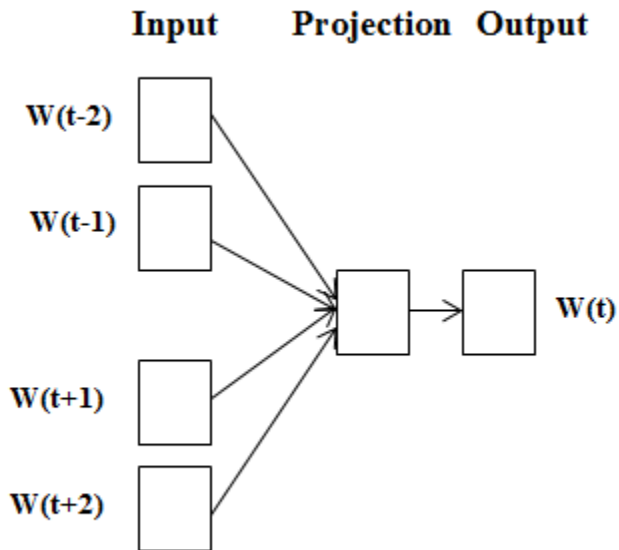
# Opakování z předchozích přednášek

# Word2Vec

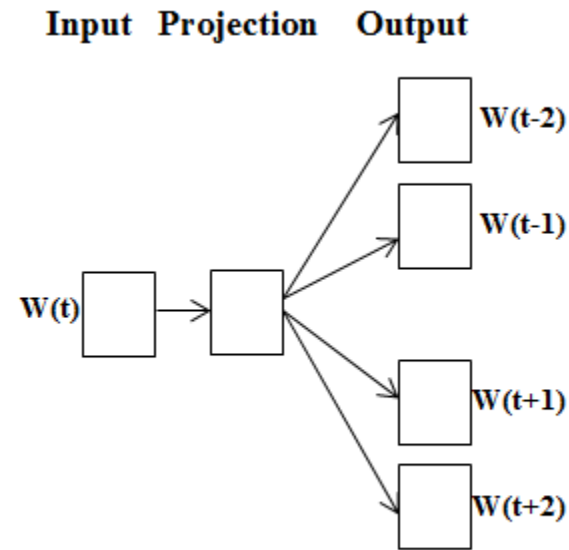


<https://towardsdatascience.com/word2vec-made-easy-139a31a4b8ae>

# Přístup: CBOW vs Skip-gram



**CBOW**



**Skip-gram**

# Problémy předchozích přístupů

- Použití omezeného kontextu okolo slova
- Každé slovo má pouze jeden embedding
  - Embedding se nemění v závislosti na kontextu, ve kterém je použit

Př. 1: Apple (společnost) vs apple (jablko)

Př. 2: Elmo (postava z pořadu Sezame, otevři se) vs Elmo (jazykový model)

Př. 3: Výslovnost slova read v závislosti na použitém čase (minulý, přítomný čas)



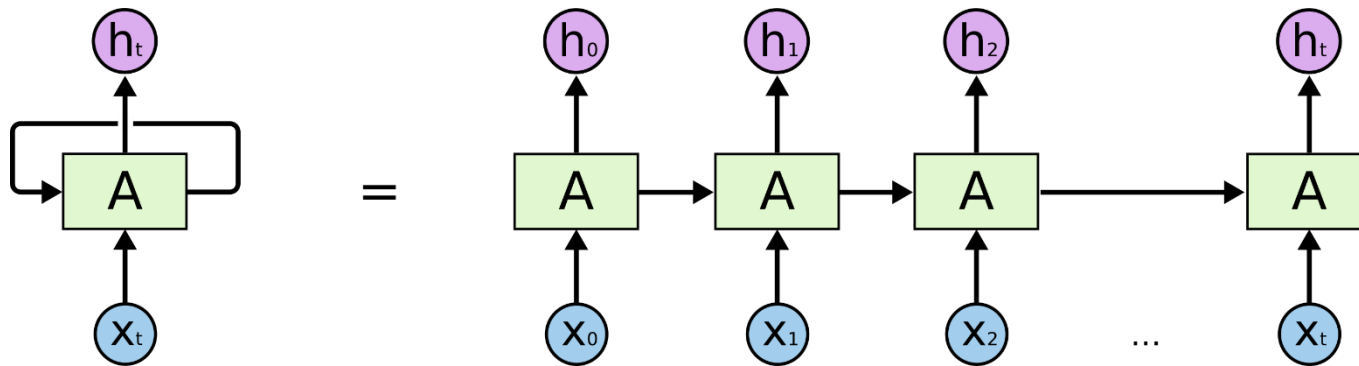
TECHNICKÁ UNIVERZITA V LIBERCI  
Fakulta mechatroniky, informatiky  
a mezioborových studií ■

# Část I.: ELMo



# Rekurentní neuronové sítě

- Opakování z předmětu ANS
- Kromě standardního výstupu obsahují i skrytý stav
  - Skrytý stav je rekurencí průběžně aktualizován

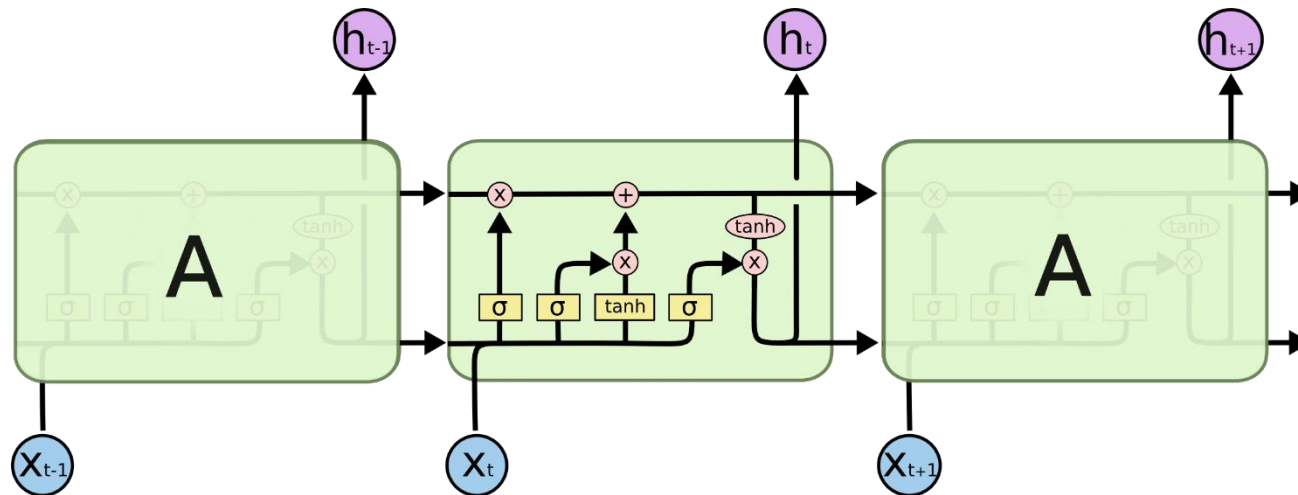


<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>



# LSTM

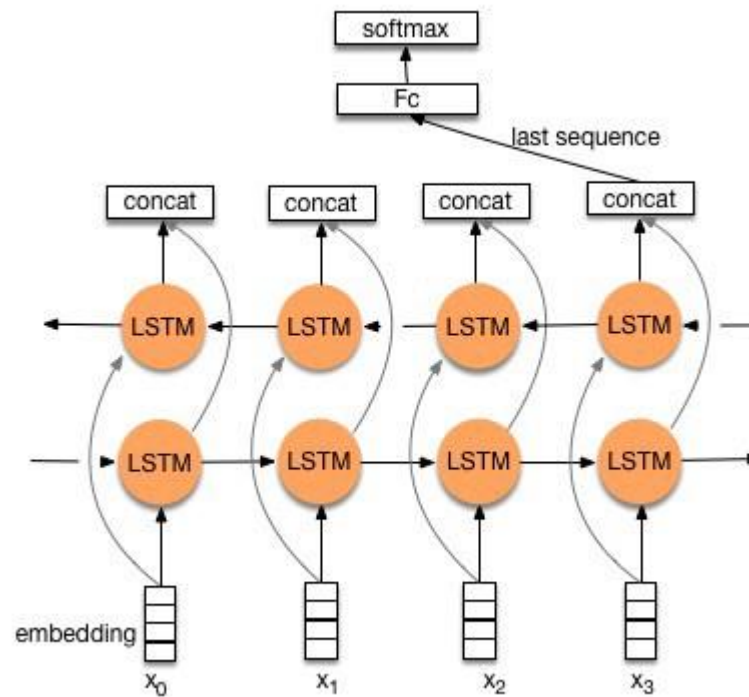
- Long Short-Term Memory
- Zachycují i dlouhodobé závislosti
  - Zároveň řeší problém zanikajícího gradientu



<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# Bidirectional LSTM

- Vstupní sekvence je procházena z obou směrů



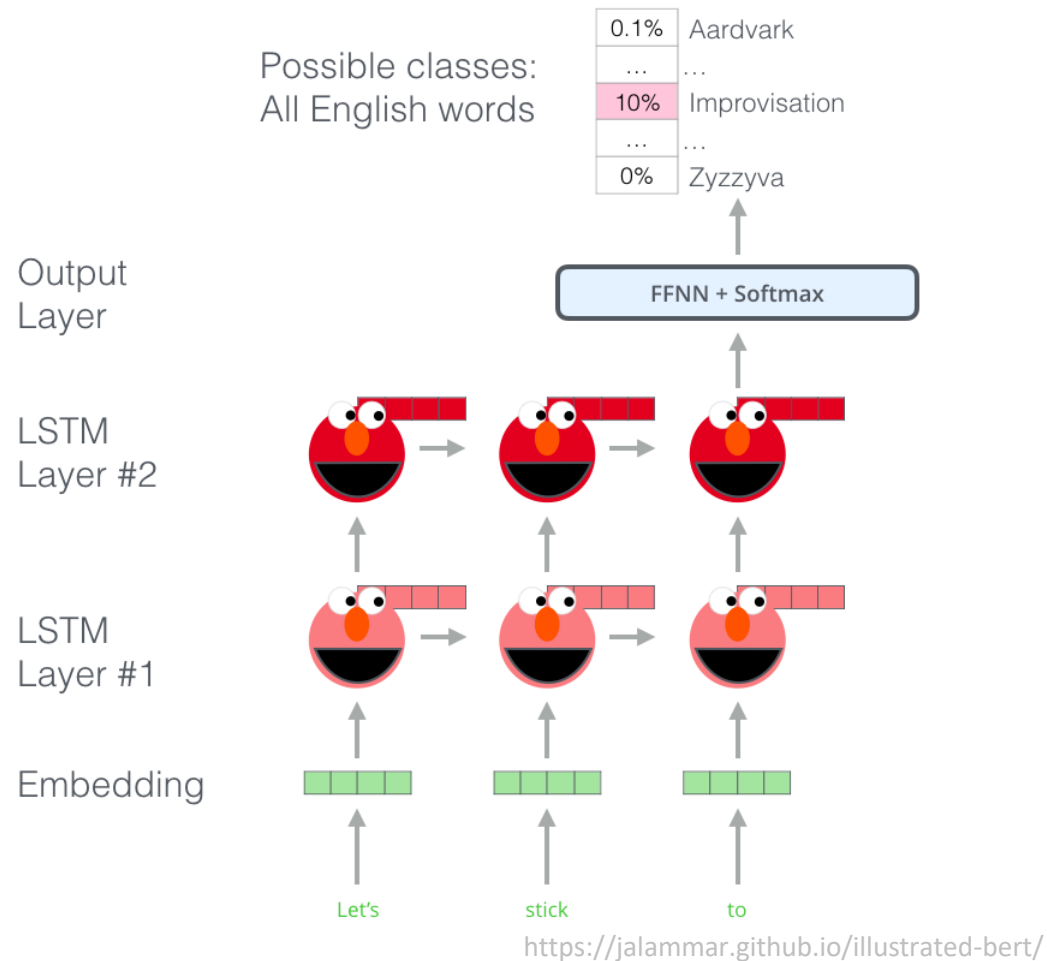
[http://doc.paddlepaddle.org/develop/doc/\\_images/bi\\_lstm.jpg](http://doc.paddlepaddle.org/develop/doc/_images/bi_lstm.jpg)

# ELMo: Contextual language embedding

- Embeddings from Language Models (2018, AllenAI)
- Používá bidirectional LSTM pro tvorbu embeddingů
- Embedding je vytvořen v závislosti na kontextu slova
- Model je založen na znacích
  - Lze vytvořit embedding pro slovo, které není ve slovníku

=> Místo použití slovníku (např. Word2Vec) vytváříme vektory za běhu

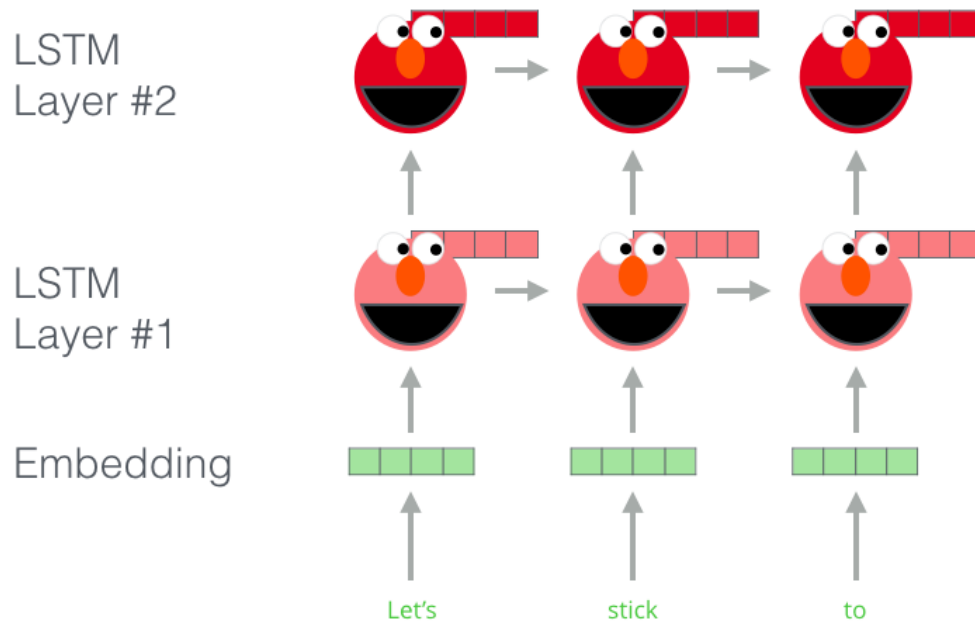
# ELMo



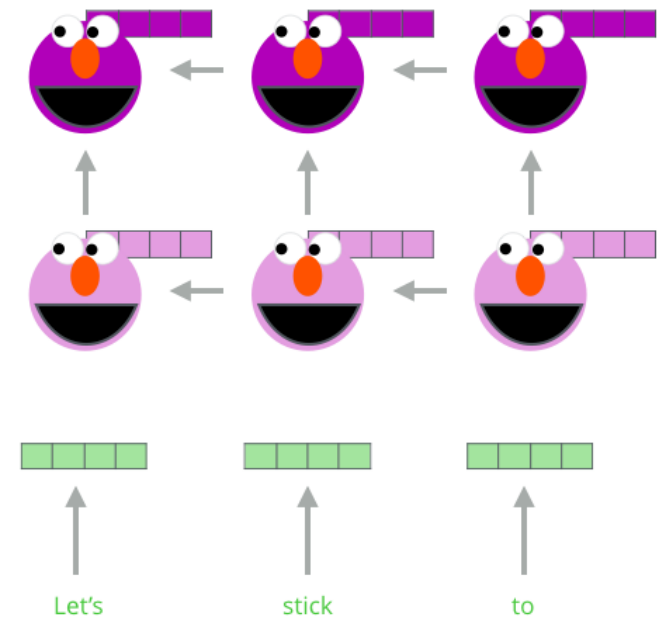
# ELMo

## Embedding of “stick” in “Let’s stick to” - Step #1

Forward Language Model



Backward Language Model



<https://jalammar.github.io/illustrated-bert/>

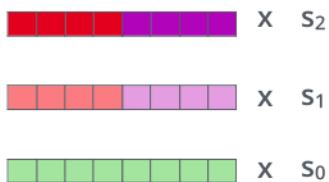
# ELMo

## Embedding of “stick” in “Let’s stick to” - Step #2

1- Concatenate hidden layers



2- Multiply each vector by a weight based on the task

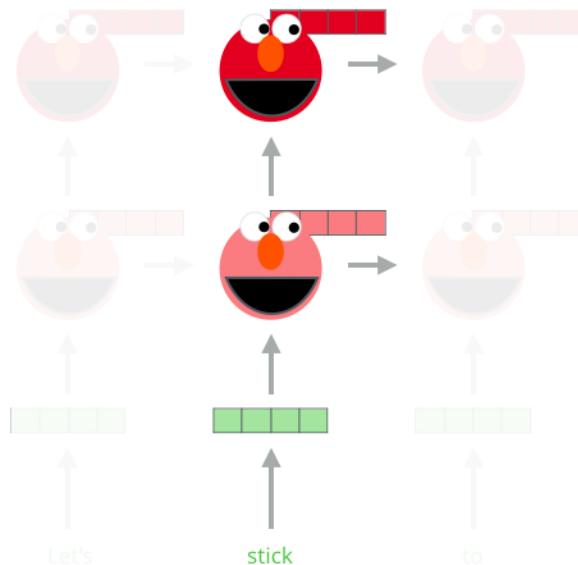


3- Sum the (now weighted) vectors

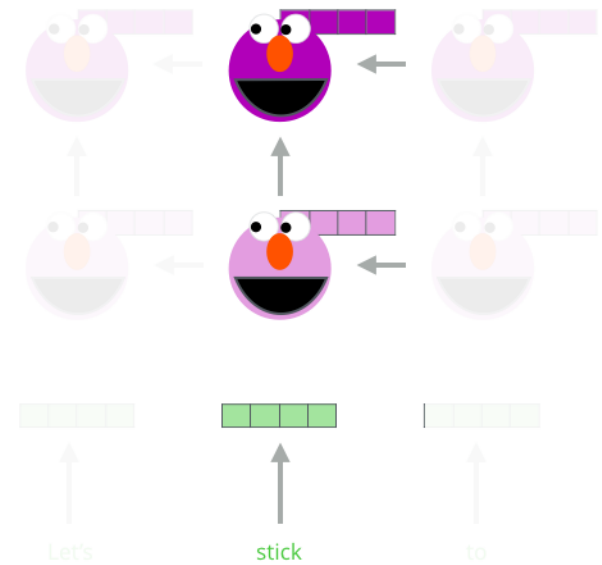


ELMo embedding of “stick” for this task in this context

Forward Language Model



Backward Language Model



<https://jalammr.github.io/illustrated-bert/>

# ELMo

| TASK  | PREVIOUS SOTA        |                  | OUR<br>BASELINE | ELMo +<br>BASELINE | INCREASE<br>(ABSOLUTE/<br>RELATIVE) |
|-------|----------------------|------------------|-----------------|--------------------|-------------------------------------|
| SQuAD | Liu et al. (2017)    | 84.4             | 81.1            | 85.8               | 4.7 / 24.9%                         |
| SNLI  | Chen et al. (2017)   | 88.6             | 88.0            | 88.7 $\pm$ 0.17    | 0.7 / 5.8%                          |
| SRL   | He et al. (2017)     | 81.7             | 81.4            | 84.6               | 3.2 / 17.2%                         |
| Coref | Lee et al. (2017)    | 67.2             | 67.2            | 70.4               | 3.2 / 9.8%                          |
| NER   | Peters et al. (2017) | 91.93 $\pm$ 0.19 | 90.15           | 92.22 $\pm$ 0.10   | 2.06 / 21%                          |
| SST-5 | McCann et al. (2017) | 53.7             | 51.4            | 54.7 $\pm$ 0.5     | 3.3 / 6.8%                          |

<https://arxiv.org/pdf/1802.05365.pdf>



TECHNICKÁ UNIVERZITA V LIBERCI  
Fakulta mechatroniky, informatiky  
a mezioborových studií ■

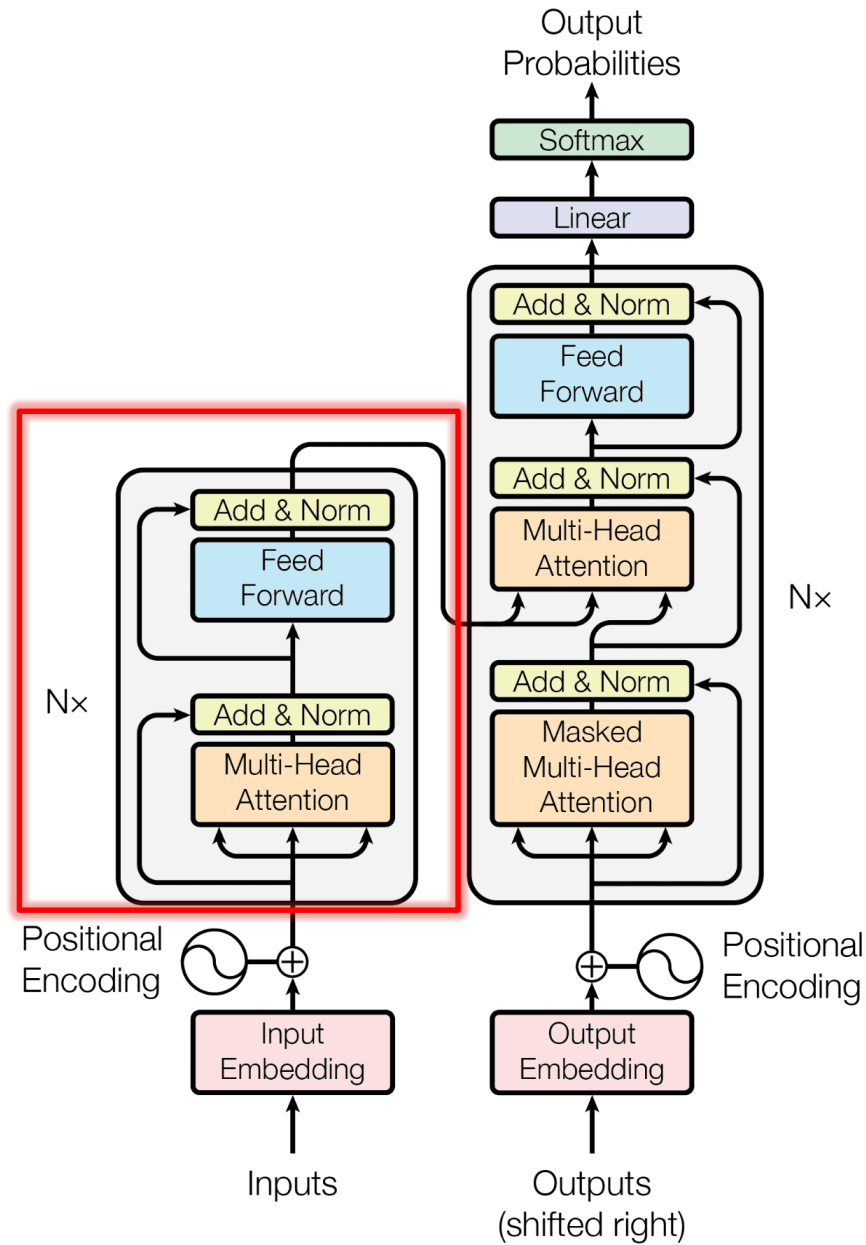
## Část II.: BERT





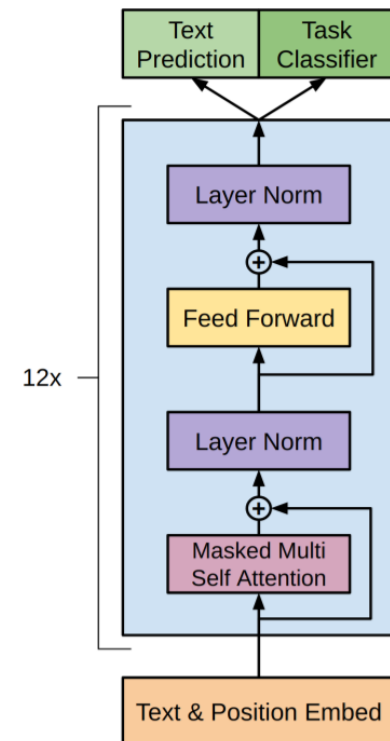
# Transformer

- Opakování z předmětu ANS
- Založeno na attention mechanismu
- Vylepšení oproti RNN
  - Odstranění rekurencí
    - => Paralelní zpracování vstupu  
(+positional encoding)
- Encoder-Decoder architektura
- **Encoder**
  - Obsahuje důležité self-attention vrstvy  $\text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$
  - Zpracovává vstupní informaci a jeho výstup obsahuje vektorovou reprezentaci vstupu



# GPT

- Generative Pre-Training
- Pre-training
  - Snaží se predikovat následující slovo
    - Trénováno jako jazykový model
  - Fine-tuning pro jednotlivé aplikace
- Z Transformeru využívá pouze dekodéry



# BERT

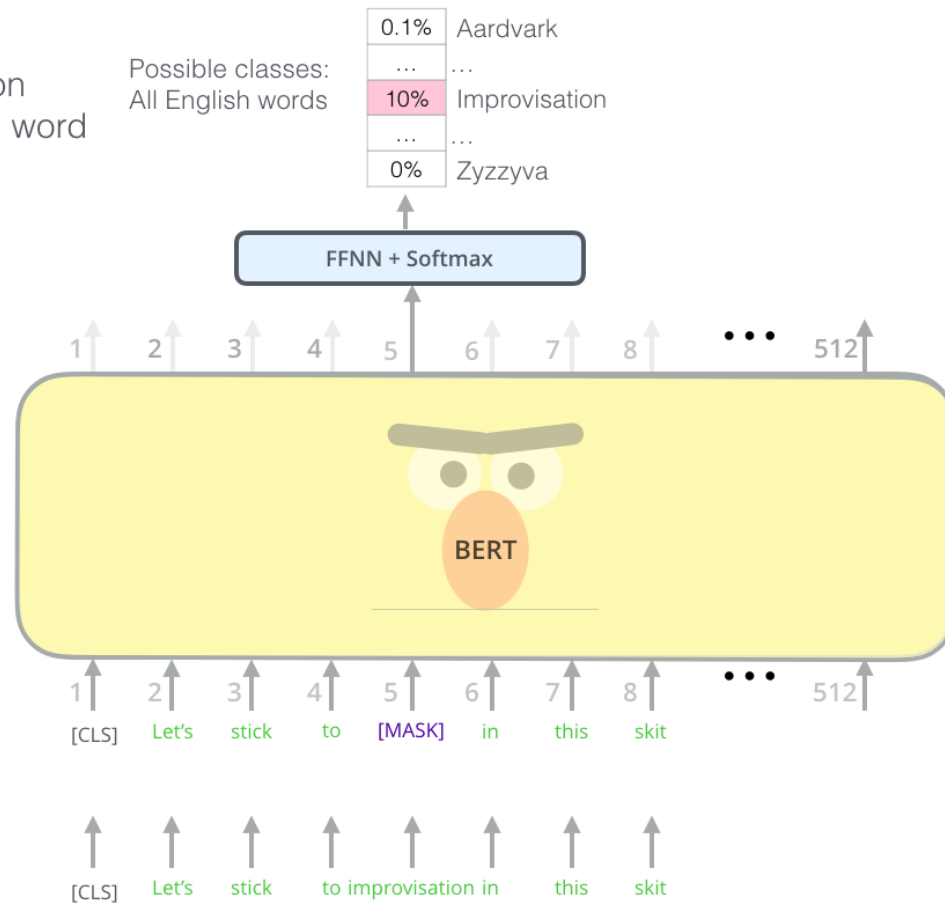
- Bidirectional Encoder Representations from Transformers
- Z Transformeru využívá pouze kodéry (encoder)
  - Neobsahuje Masked self attention
    - Predikce slova při trénování není založena pouze na předchozích, ale i na budoucích slovech
    - Lze vytvořit slovní embedding závislý na kontextu (podobně jako u ELMo modelu)
- Pre-training
  - Jazykový model se učí dvě úlohy
    - Predikce maskovaného slova ve větě
    - Predikce další věty (binární klasifikace)
      - Na vstupu jsou dvě věty a cílem je určit, zda druhá věta následuje za první

# BERT

- Predikce maskovaného slova ve větě
  - 15 % slov je při trénování náhodně maskováno
  - Postup:
    - V 80 % případů je slovo nahrazeno MASK tokenem
    - V 10 % případů je nahrazeno náhodným jiným tokenem
    - V 10 % případů je ponecháno beze změny
  - Mask může být více v jedné větě

# BERT

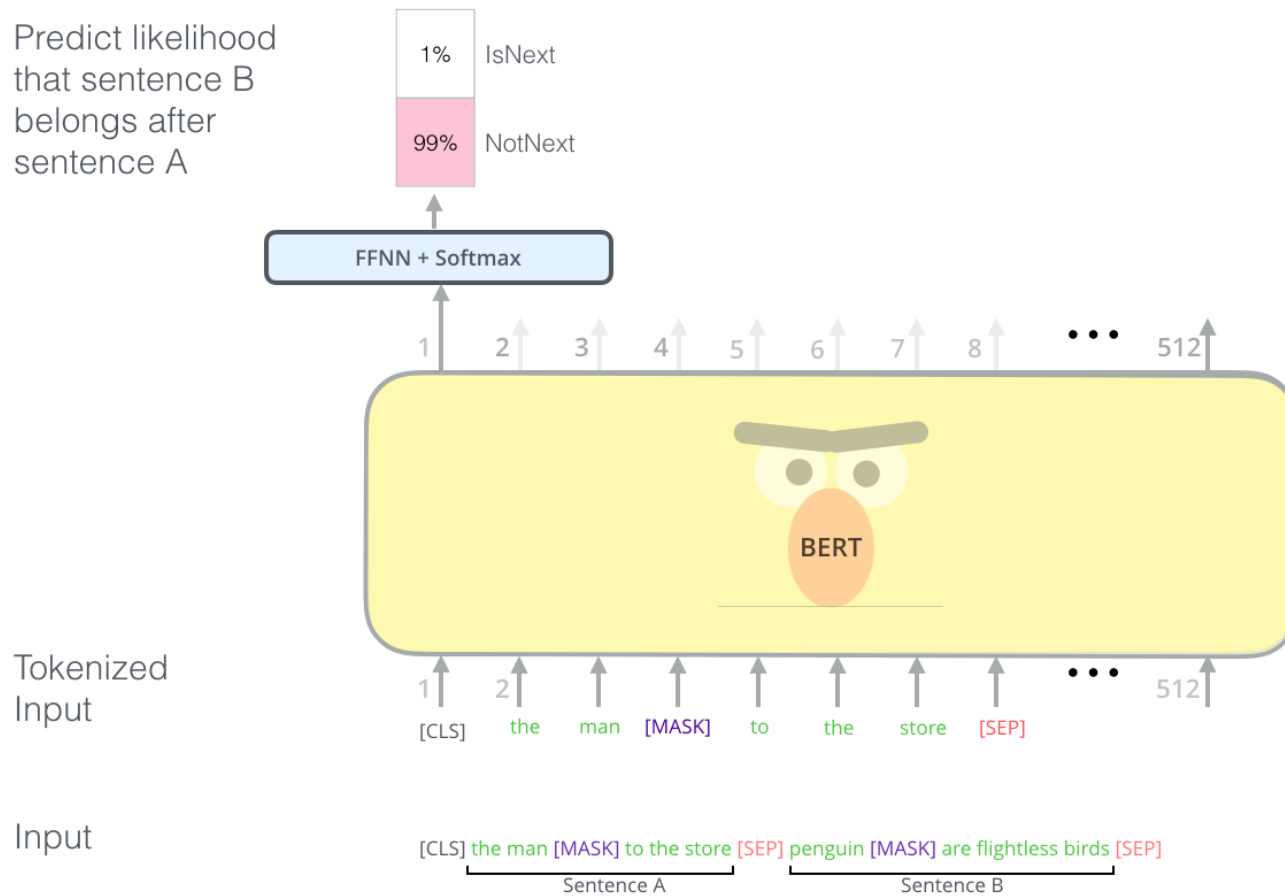
Use the output of the masked word's position to predict the masked word



<https://jalammar.github.io/illustrated-bert/>

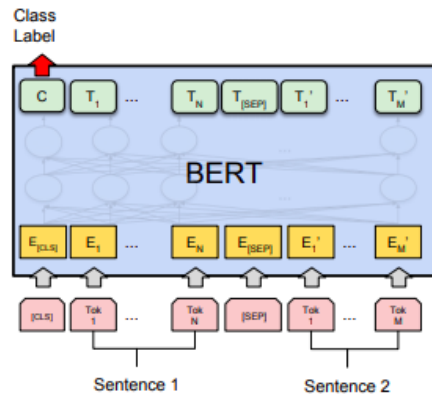
# BERT

Predict likelihood  
that sentence B  
belongs after  
sentence A

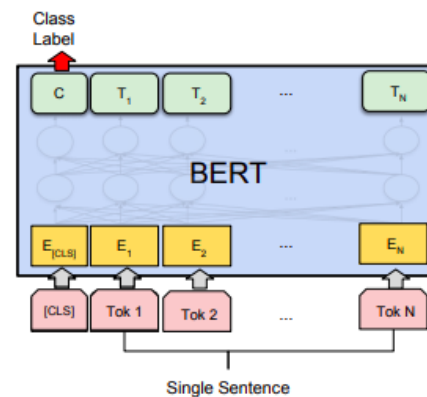


<https://jalammar.github.io/illustrated-bert/>

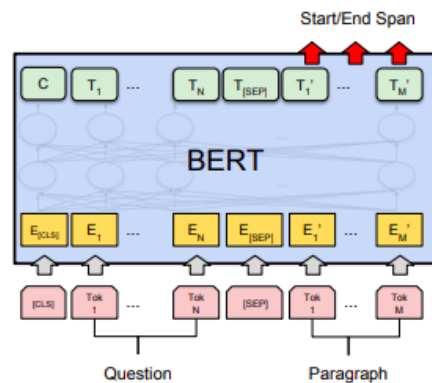
# BERT



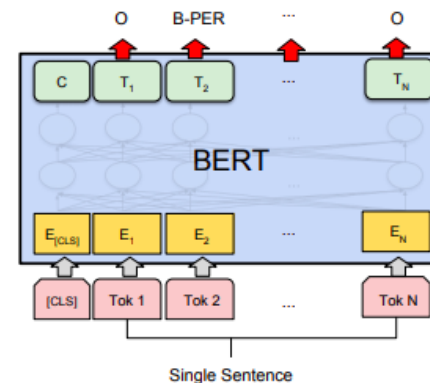
(a) Sentence Pair Classification Tasks:  
MNLI, QQP, QNLI, STS-B, MRPC,  
RTE, SWAG



(b) Single Sentence Classification Tasks:  
SST-2, CoLA



(c) Question Answering Tasks:  
SQuAD v1.1

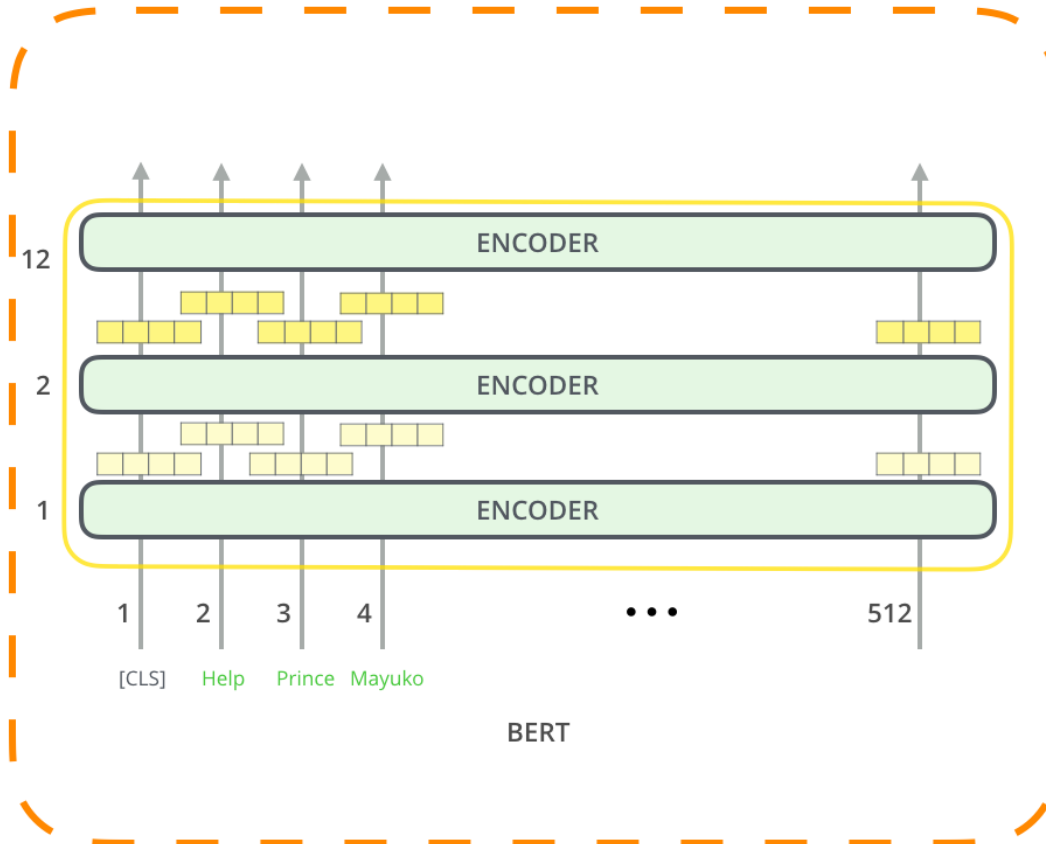


(d) Single Sentence Tagging Tasks:  
CoNLL-2003 NER

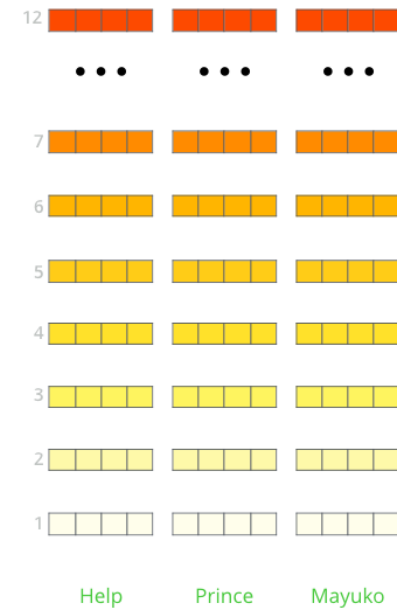


# BERT

## Generate Contextualized Embeddings



The output of each encoder layer along each token's path can be used as a feature representing that token.


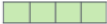





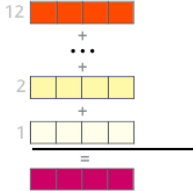


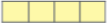

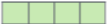
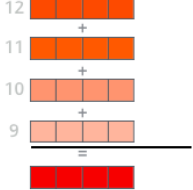



But which one should we use?

<https://jalammar.github.io/illustrated-bert/>

# BERT

What is the best contextualized embedding for “**Help**” in that context?  
 For named-entity recognition task CoNLL-2003 NER

|  |  | Dev F1 Score |
|--|--|--------------|
| 12  | First Layer Embedding        | 91.0         |
| ...  |  |              |
| 7   | Last Hidden Layer            | 94.9         |
| 6   |  |              |
| 5   | Sum All 12 Layers  | 95.5         |
| 4   |                              |              |
| 3   | Second-to-Last Hidden Layer  | 95.6         |
| 2   |  |              |
| 1  | Sum Last Four Hidden   | 95.9         |
|   |                             |              |
| Help   | Concat Last Four Hidden  | 96.1         |
|  |                            |              |

<https://jalammar.github.io/illustrated-bert/>



# Část III.: HuggingFace



# HuggingFace

- Opensource databáze předtrénovaných modelů založených na Transformer architektuře
  - 14 000+ modelů
- Také poskytuje různé datasety a různé nástroje pro usnadnění manipulace s daty
  - U datasetů jsou často odkazy na již natrénované modely
- Poskytována Python knihovna transformers

=> Online ukázka

# Užitečná literatura / kurzy

- Články jednotlivých modelů
  - [ELMo](#)
  - [Transformer](#)
  - [GPT](#)
  - [BERT](#)
- [HuggingFace](#)
- [Coursera NLP specializace](#)