ABBYY

Advances in Natural Language Processing

MLSS-2019
31.08.2019

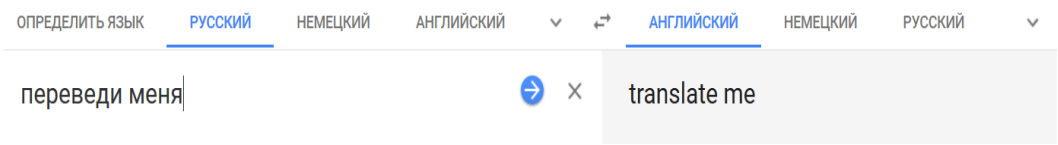
Ivan Smurov, ivan.smurov@abbyy.com

Overview

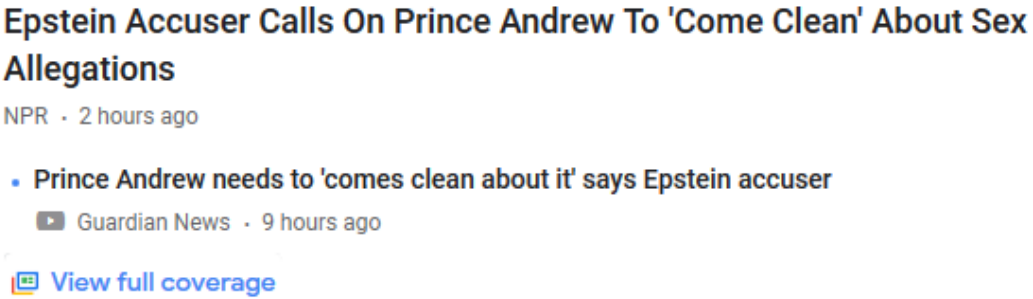
- Core NLP tasks
- NLP pipeline: embeddings, CharCNN, BiRNNs
- Seq2seq and Transformers
- Language model embeddings: ELMo, ULMFit, BERT

Core NLP tasks

- Machine translation



- Text classification
  - Spam/ham
  - Sentiment analysis (reviews)
  - Thematic text classification
- Text clustering

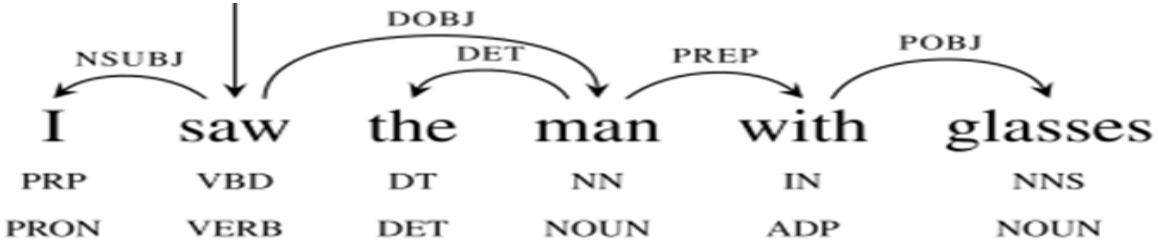


Core NLP tasks

- Named Entity Recognition(NER)

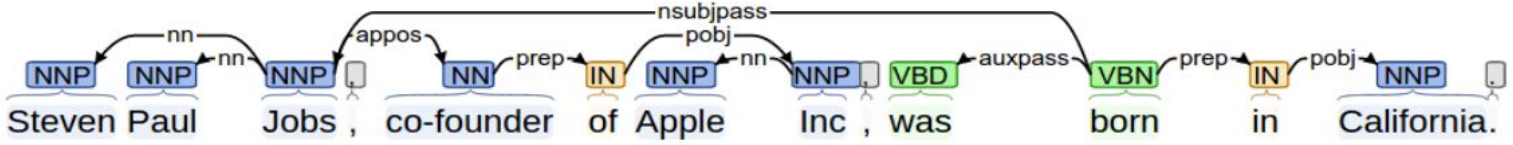
Kofi Atta Annan is a Ghanaian diplomat who served as the seventh Secretary General of the United Nations from January 1, 1997, to January 1, 2007, serving two five-year terms. Annan was the co-recipient of the Nobel Peace Prize in October 2001.

- Syntactic Parsing



Core NLP tasks

- Relation extraction (fact extraction)

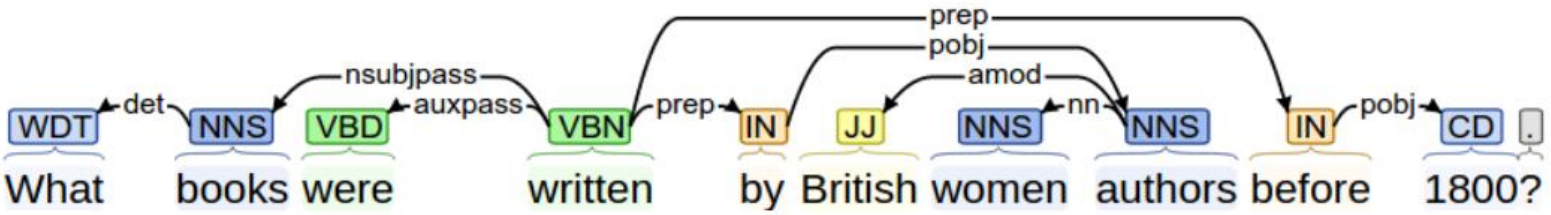


- Summarization

Article
(cnn) to allay possible concerns, boston prosecutors released video friday of the shooting of a police officer last month that resulted in the killing of the gunman. the officer wounded, john moynihan, is white. angelo west, the gunman shot to death by officers, was black. after the shooting, community leaders in the predominantly african-american neighborhood of (...)
Human-written summary
boston police officer john moynihan is released from the hospital. video shows that the man later shot dead by police in boston opened fire first. moynihan was shot in the face during a traffic stop.
Generated summary (See et al., 2017)
boston prosecutors released video friday of the shooting of a police officer last month. the gunman shot to death by officers , was black . one said the officers were forced to return fire. he was placed in a medically induced coma at a boston hospital.

Core NLP tasks

- Question-answering Systems (QA)



- Conversational Systems, Chatbots
  - Amazon Alexa
  - Siri
  - Yandex Alice

NLP is simple, right?

Sale of chicken murder

Go back toward your behind

Deep fried baby

Meat muscle stupid bean sprouts

Why NLP is not that simple

ABBYY

- Language is highly ambiguous, there exist such phenomena like polysemy and homonymy
  - Polysemy: assembly, board, court
  - Homonymy: bear, yard, train
  - Press space bar to continue -> бар космический пресс продолжает работу
- There also exist sophisticated phenomena such as:
  - Coreference: John hit the ball. He was angry
  - Ellipsis: I ate the green apple and you, the red one.
- While humans can resolve such ambiguities by context easily enough, doing this automatically is extremely challenging

Towards NLP Pipeline

ABBYY

- Until recently (2-4 years ago depending on a specific task) deep learning did not perform significantly better than classical machine learning. Now however the gap between these two approaches is rapidly increasing.
- For most NLP tasks classical approaches have a task-specific architecture and feature set that cannot be used in other tasks.
- Neural network architectures for different tasks have a lot more in common, while the features used are in most cases from one feature set.
- Thus we can talk about a universal NLP pipeline

NLP pipeline first steps: segmentation and tokenization

ABBYY

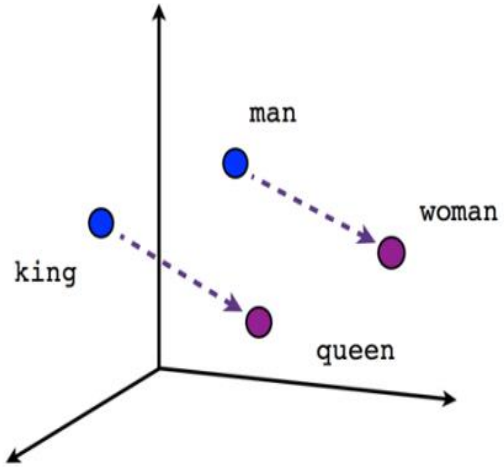
- For most NLP tasks we work with discrete unit or token (words) features.
- Token features should include some context-based information. In modern models context is usually sentence level.
- Many NLP tasks (e. g. machine translation) are solved on the sentence level. Others (e. g. chatbots) require extra sentence context.
- Thus first two steps of NLP pipeline are sentence segmentation and word tokenization.

Token features: embeddings

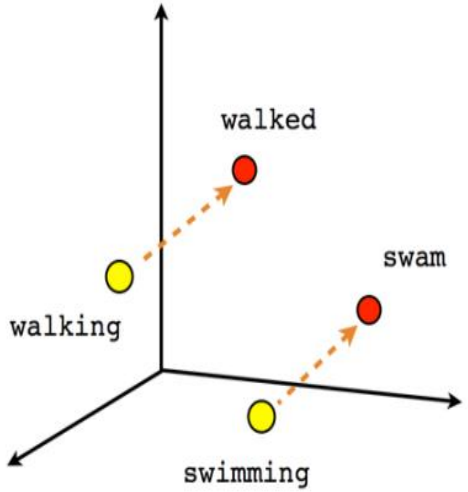
ABBYY

- An embedding is a mapping of discrete feature vector into dense vector of fixed dimension *h*.
  - Classic example is word embedding.
    Original vector: x = (0, 0, … , 1, 0, … 0) of length *v*, where *v* is the size of dictionary
    Resulting vector: x'= (0.2, 0.8, …, -15.9) of length *h*.
- Notable advantages:
  - Feature space dimension reduction.
  - Distributional properties i. e. similar elements have close embeddings.
- Can be trained from scratch but pretraining can boost performance significantly

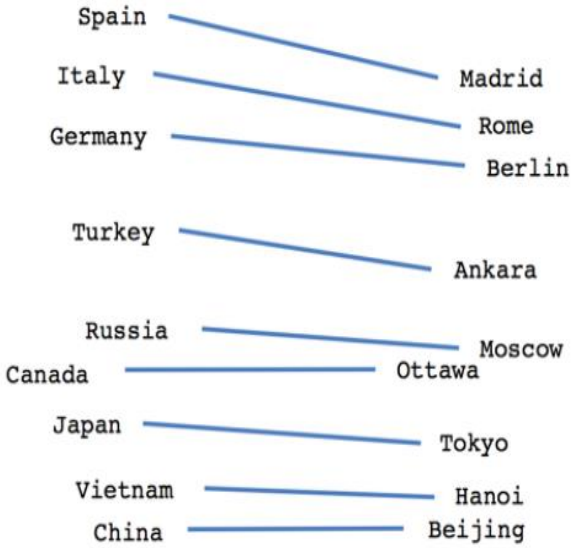
Embedding: vector magic



Male-Female



Verb tense



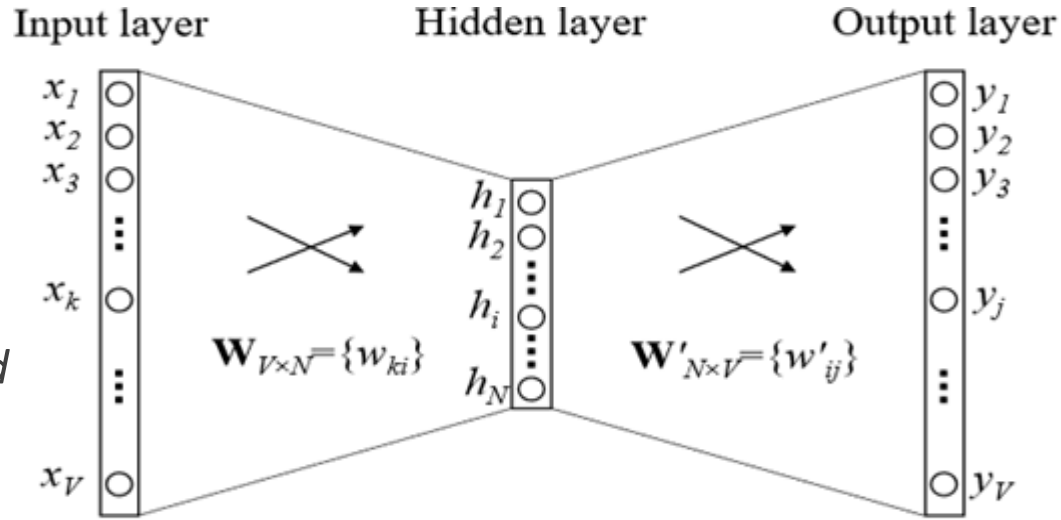
Country-Capital

Embeddings pretraining

- Let us train on the task of predicting next token given previous one
  - Input: one-hot encoding of the previous token
  - Output: probability distribution of the next token over vocabulary
  - Activation function of the last layer is softmax:

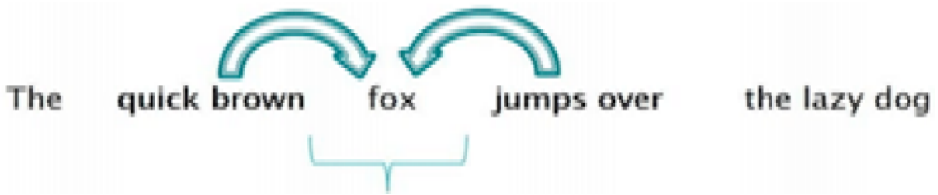
$$p(u|v) = \frac{\exp\langle \phi_u, \theta_v \rangle}{\sum_{u' \in W} \exp\langle \phi_{u'}, \theta_v \rangle}$$

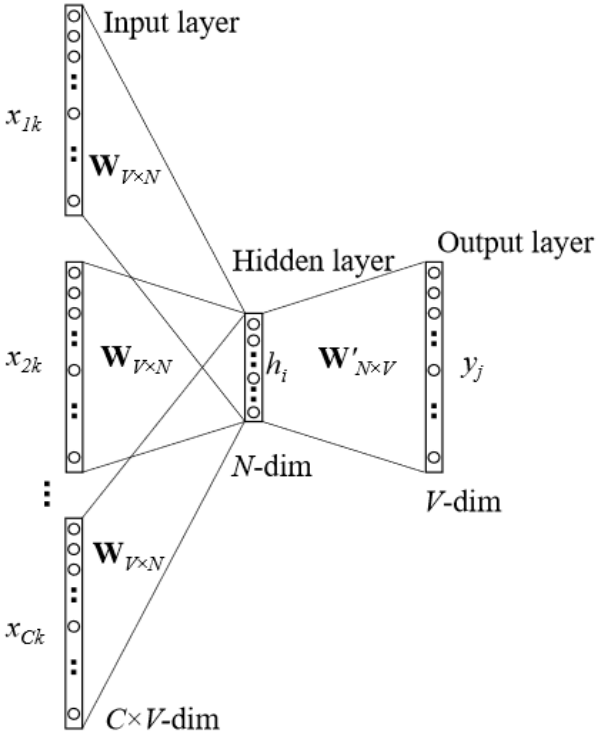
  - W or W’ are embedding matrices
- Model is extremely simple and we can train on *unlabeled* data. Thus corpora used for training are large (Wiki++)



Continuous bag of words

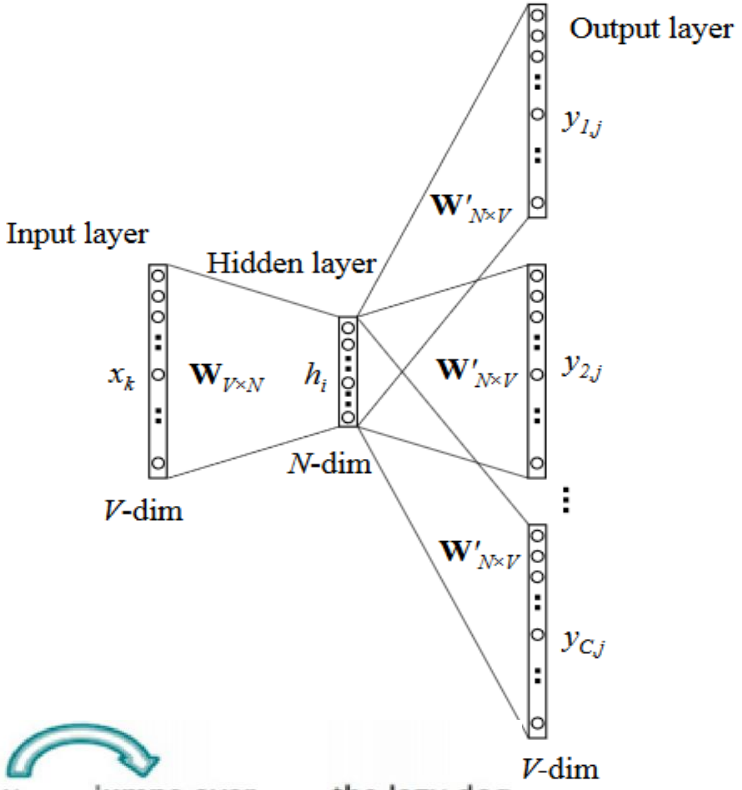
- Let us use for predictions several-token contexts instead of one previous token
- For simplicity we will use weights 1 for all tokens present in context and 0 for all not present.
- Order of tokens in context is not taken into account
- This model resembles bag of words,
  thus the name: CBOW – continuous bag of words

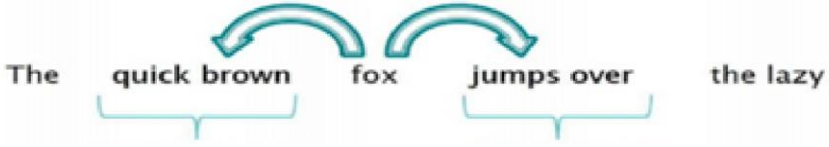




Skip-gram model

- Let us consider a dual task: predict context given word
- Model is called skip-gram
- On downstream tasks works better than CBOW
- Output vector has the dimension V (size of vocabulary) thus computing softmax is computationally expensive. Many techniques can be used for optimization. The most popular: negative sampling





Negative sampling

- Key idea: instead of predicting the output distribution we intend to differentiate each positive sample from k random samples.
- Positive samples are pairs of tokens and their contexts (w, c) occurred in the training data.
- Negative samples are sampled as follows $(w, c) \sim p_{words}(w)\frac{p_{contexts}(c)^{3/4}}{Z}$
- Loss formulation: $\arg\max_{\theta} \sum_{(w,c)\in D} \log \sigma(v_c \cdot v_w) + \sum_{(w,c)\in D'} \log \sigma(-v_c \cdot v_w)$
- Trained by SGD
- More computationally effective than computing full softmax while having leading to similar results

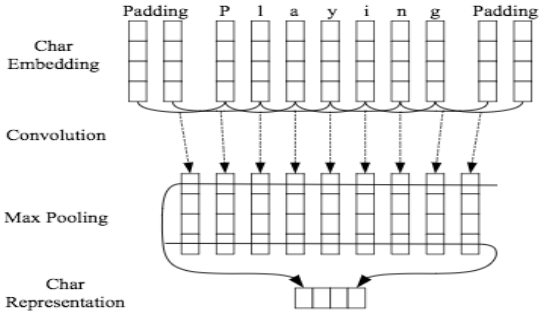
Word2vec: summary and problems

ABBYY

- Word embeddings successfully solve 2 main problems of simple vector models:
  - Embedding dim can be manageable: usually around 100-1000
  - Embeddings can represent similarity of tokens (synonymy etc.)
- Vanilla word2vec have several unsolved problems:
  - Embeddings are built using fixed dictionary. Vectors on out of vocabulary words are not defined. There are several traditional approaches to treat OOV:
    - One vector for all OOV
    - OOV depends on grammatical characteristics: e. g. a separate vector for OOV nouns, verbs, ect.
    - OOV vector is computed as a mean for all embeddings in its context
  - Embeddings depend only on the graphical form of the token i. e. embedding for “Train” is same in both contexts “Train, dev, test” and “Train arrives at 10:15”

NLP pipline: token features

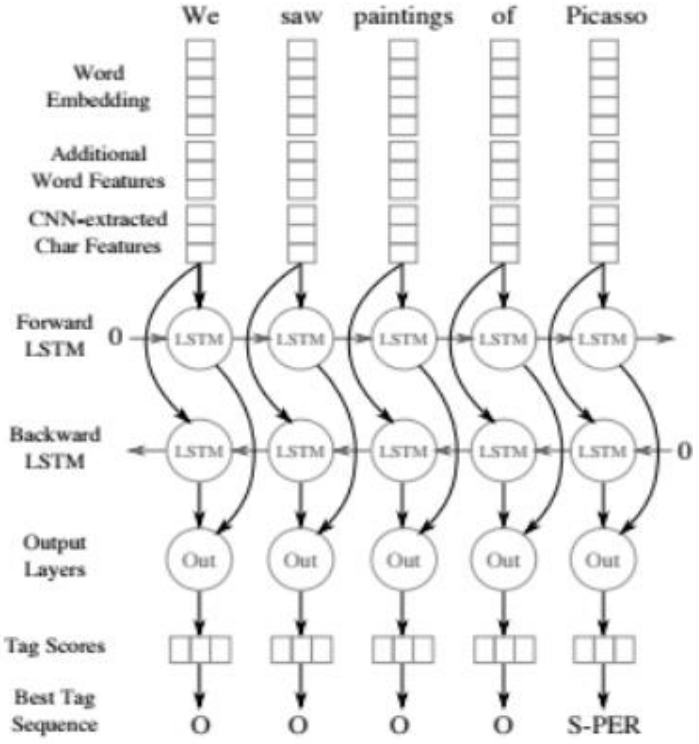
- Usually token features can be split into 3 groups:
  - Word embeddings. In academia embeddings pretrained on large corpus are usually tuned on smaller corpus. In practice having constant pretrained embeddings yield similar results
  - Char-level features: char embeddings for each token are fed into CNN (or RNN) of limited dim. The result is concatenated with other token features
  - Additional token features: POS-tags (or their embeddings), capitalization etc.



- Method first introduced in its whole in :
  Lample el al (2016) Neural Architectures for Named Entity Recognition and
  Ma and Hovy (2016) End-to-end Sequence Labeling via Bi-direcitonal LSTM-CNNs-CRF

Modern approaches: Embeddings + CharCNN + BLSTM

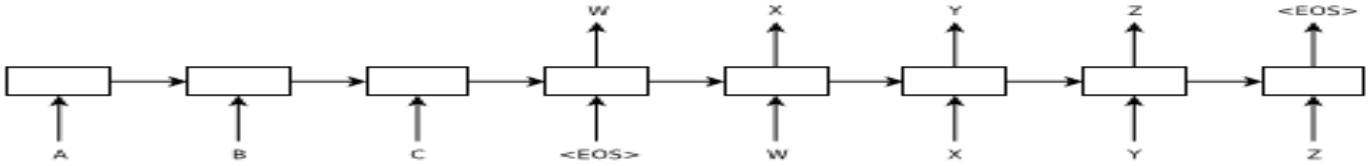
- Almost any NLP task can be solved with the following architecture:
- We compute *context-independent* feature for each token (embeddings, CharCNN, additional features)
- This vectors are fed into Bidirectional RNN in order to compute *context-dependent features* for each token
- Top layer is task specific: for example in NER popular top layer is CRF



BLSTM structure for tagging named entities.

Seq2seq without attention mechanism

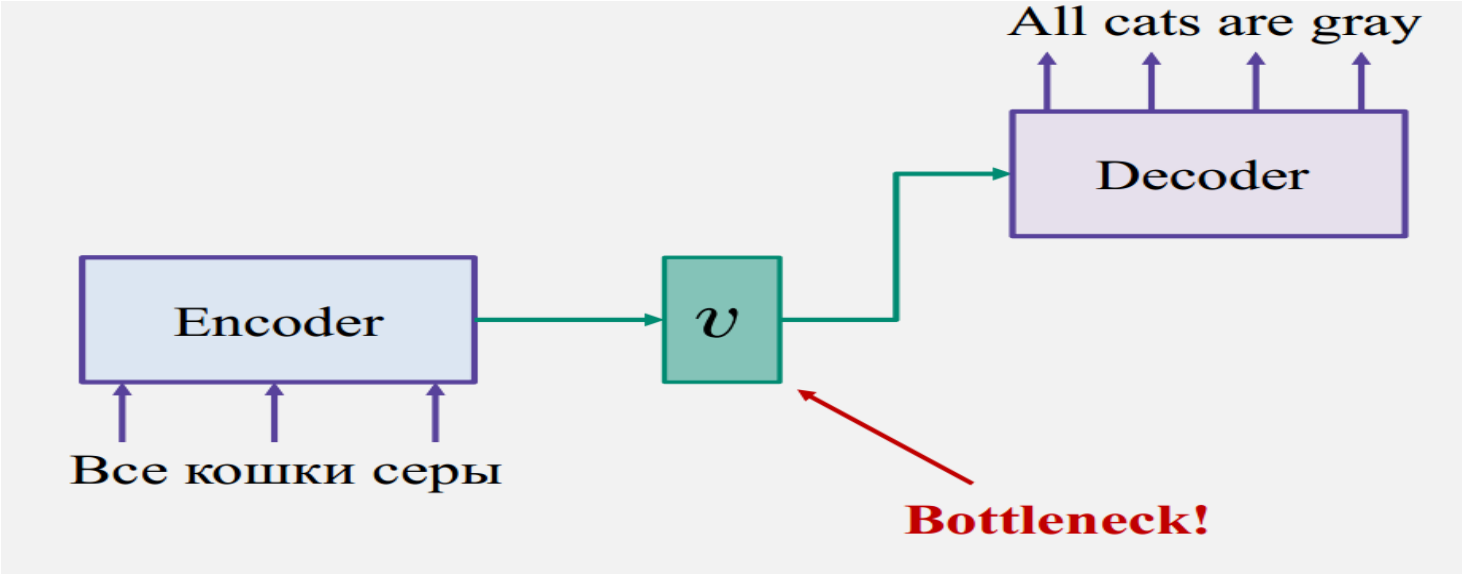
- Main idea: most NLP tasks (e. g. MT) can be treated as follows: given input sequence generate output sequence (possibly of different length)
- 2 main parts of neural net – encoder and decoder (both consisting of recurrent layers). Encoder processes input and generates vector *c* with info of all sequence.
- Decoder generates output sequence until EOS is generated. Input consists of *c,* previous decoder state *h* and output generated on the previous step



- *Sutskever et al 2014 – "Sequence to sequence learning with neural networks"*

Vanilla seq2seq disadvantages

- The whole input is compressed into one vector.
- Thus the representation quality decreases for longer input sequences

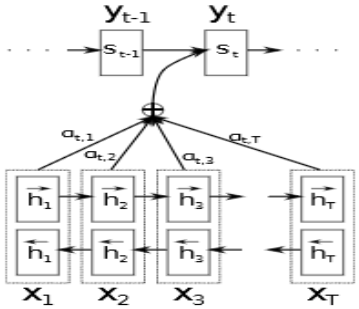


Attention mechanism

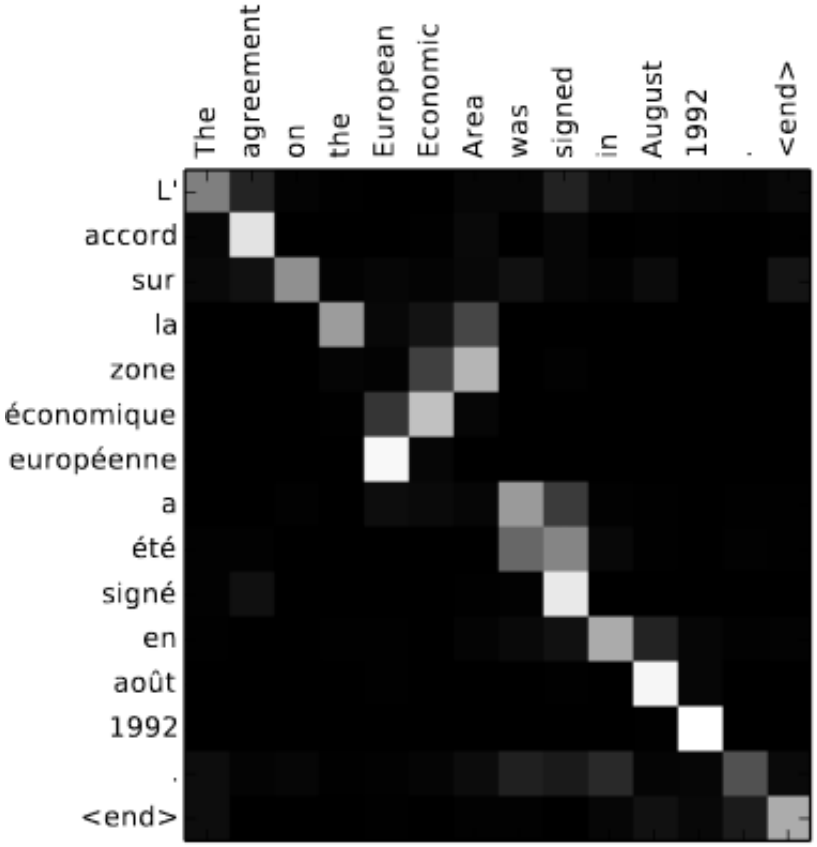
- Decoder states – s_i, encoder states – h_j.

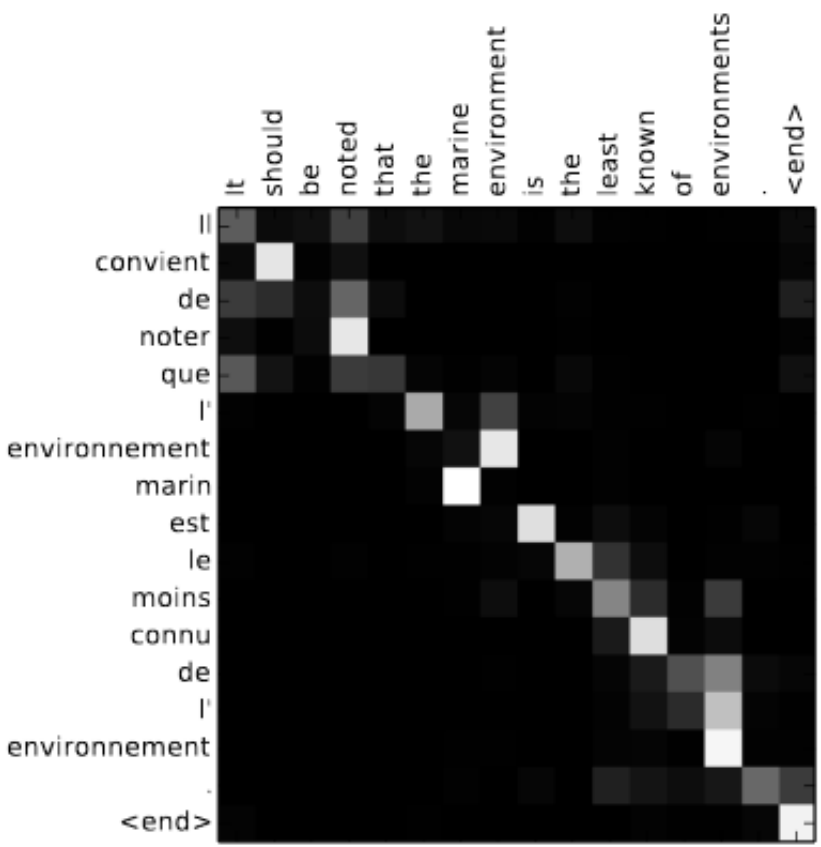
$$s_i = f(s_{i-1}, y_{i-1}, c_i). \quad c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \quad e_{ij} = a(s_{i-1}, h_j) \; e_{ij} = v_a^\top \tanh(W_a s_{i-1} + U_a h_j),$$

- Let us introduce attention c_i – soft alignment between output at specific step and the elements of input that influence output most at that step
- Almost no decrease in the quality of translation on longer sentences
- *Bahdanau et al 2014 "Neural Machine Translation by Jointly Learning to Align and Translate"*



Soft alignment





Global and local attention

- Attention is calculated after decoder. Thus architecture becomes less sophisticated and stacking decoder layers can be realized in a natural way

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

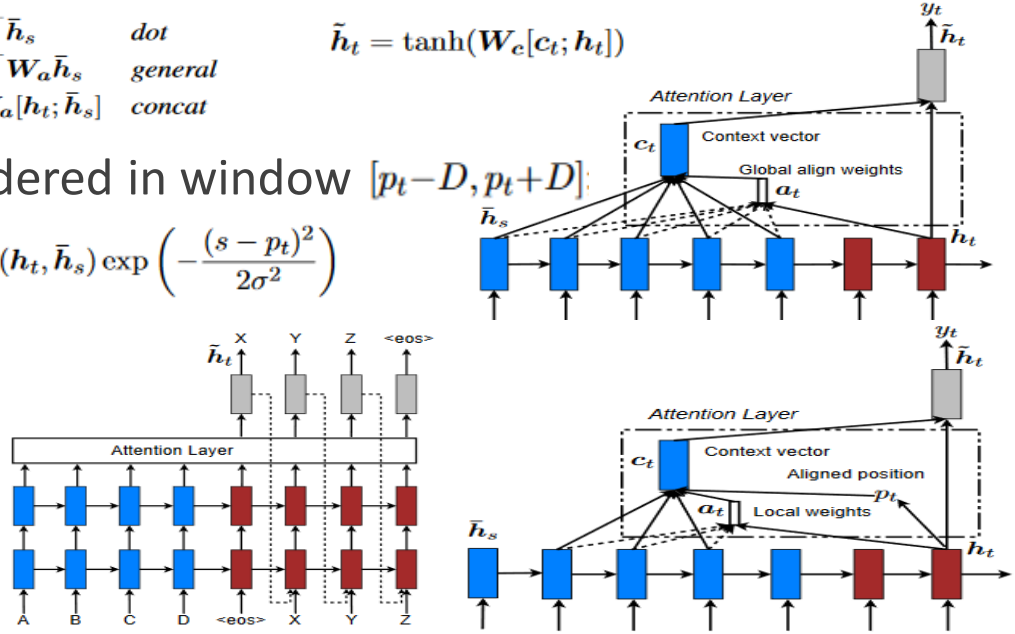
$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{W}_a[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s] & \textit{concat} \end{cases}$$

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_c[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

- Local attention: attention is considered in window $[p_t - D, p_t + D]$

$$p_t = S \cdot \text{sigmoid}(\boldsymbol{v}_p^\top \tanh(\boldsymbol{W}_p \boldsymbol{h}_t)) \qquad \boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right)$$

- Attention history can be utilized
- *Luong et al 2015“Effective Approaches to Attention-based Neural Machine Translation.”*



Transformer overview

- Main idea: we can train encoder-decoder structures without using RNNs with the help of attention mechanism.
- Training on comparable data yields similar results to RNN seq2seq.
- Using dense layers instead of RNNs is much more suitable for parallelism.
- Since without RNNs there is no information about relative positions of input and output tokens, we have to use positional embeddings:

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

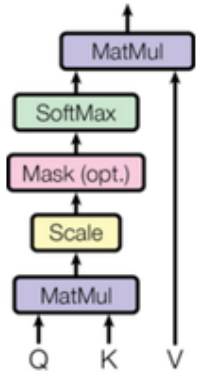
- *Vaswani et al (2017) Attention is all you need*

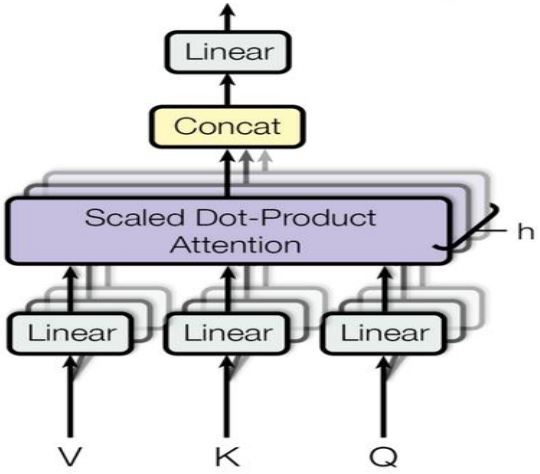
Scaled dot product attention and Multi-head attention

- Let us redefine dot product attention used in seq2seq as follows: Let us have 3 matrices: Q (queries), K (keys) and V (values). $\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$
- In classical attention Q corresponds to decoder states matrix, K = V to encoder states matrix.
- $\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$ follows:

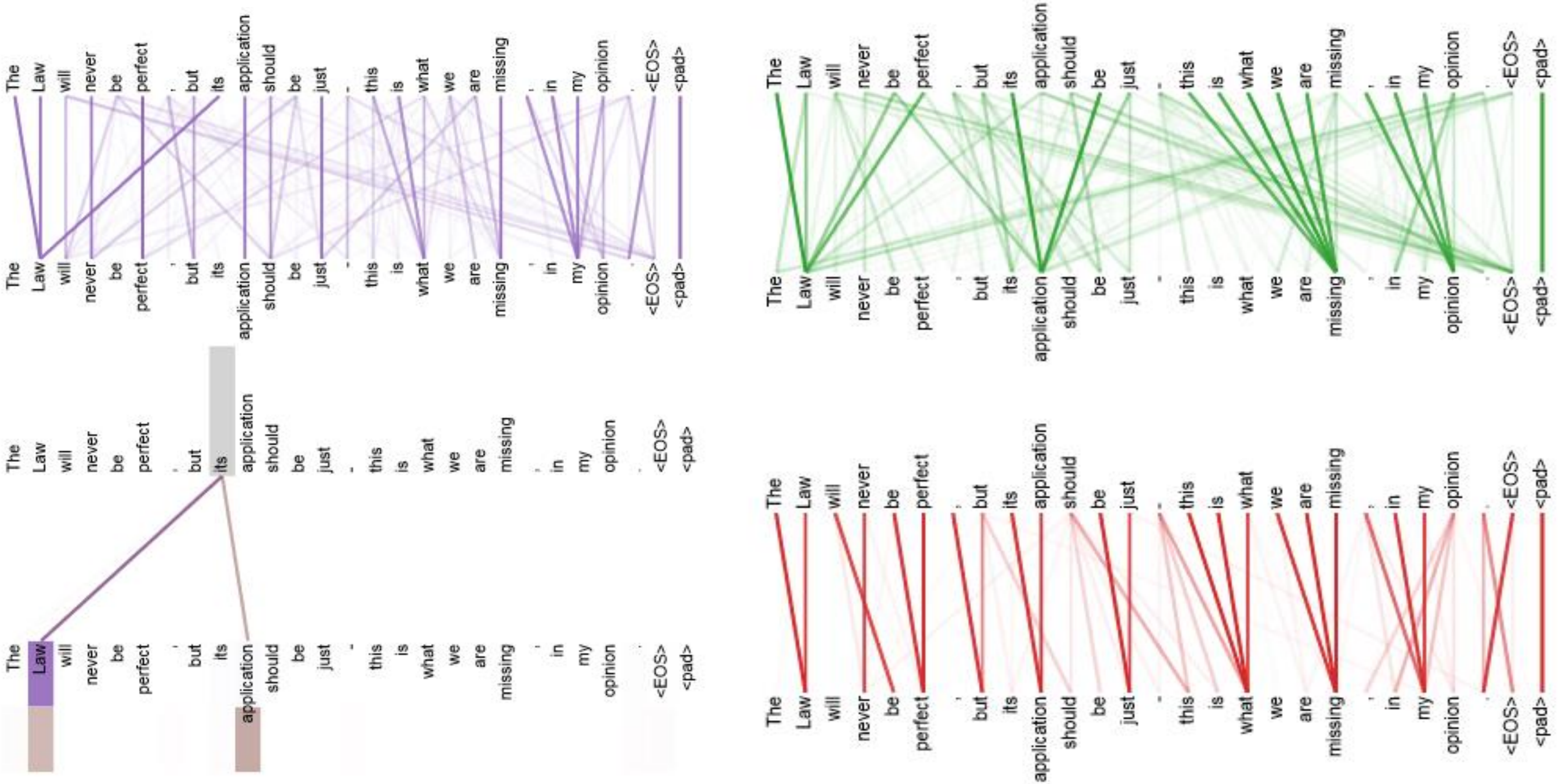
  $\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.



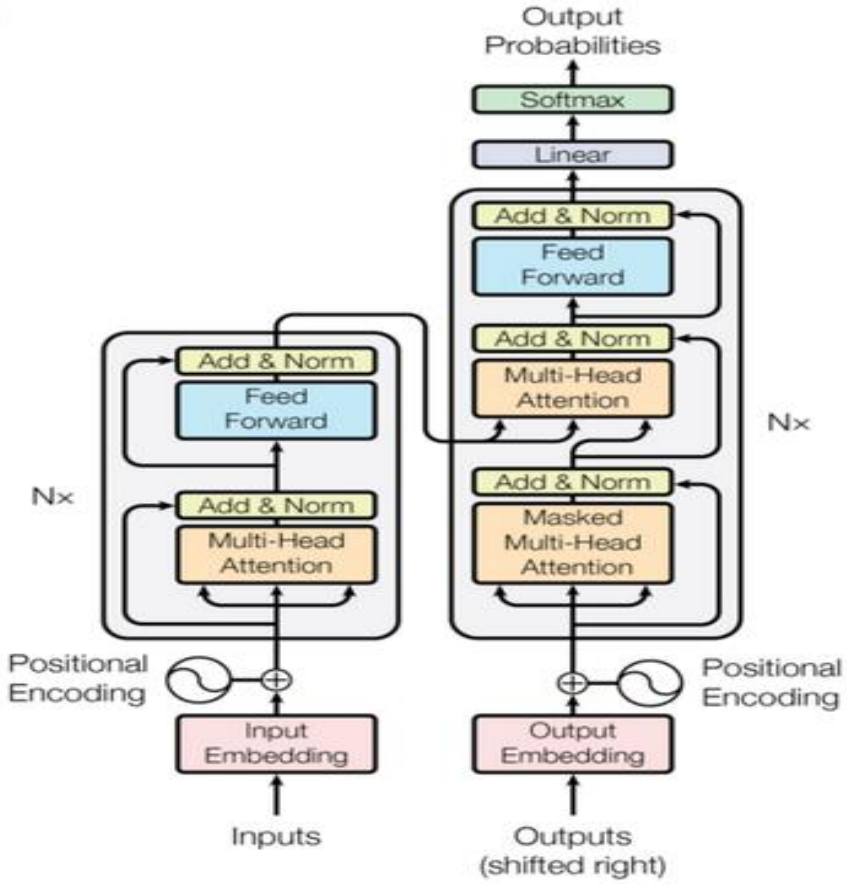


Attention on different heads



Transformer encoder and decoder

- A good implementation of transformer with necessary comments can be found here: **The Annotated Transformer**



Language modeling

- Language modeling is an NLP task of predicting the probability of the next token in sequence given all previous tokens $P(w_n|w_1, \ldots, w_{n-1})$
- E. g. what sequence is more probable
  - London is the capital of Great
    - Britain
    - Depression
- Let us compare language models with word2vec training task. Two key differences:
  - Language models take into account the order of tokens in context
  - Word2vec uses a fixed size window as a context, LMs the whole previous text.
- This also allows to compute the probability of the whole sequence $P(w_1, \ldots, w_n)$

Embeddings from Language models- ELMo

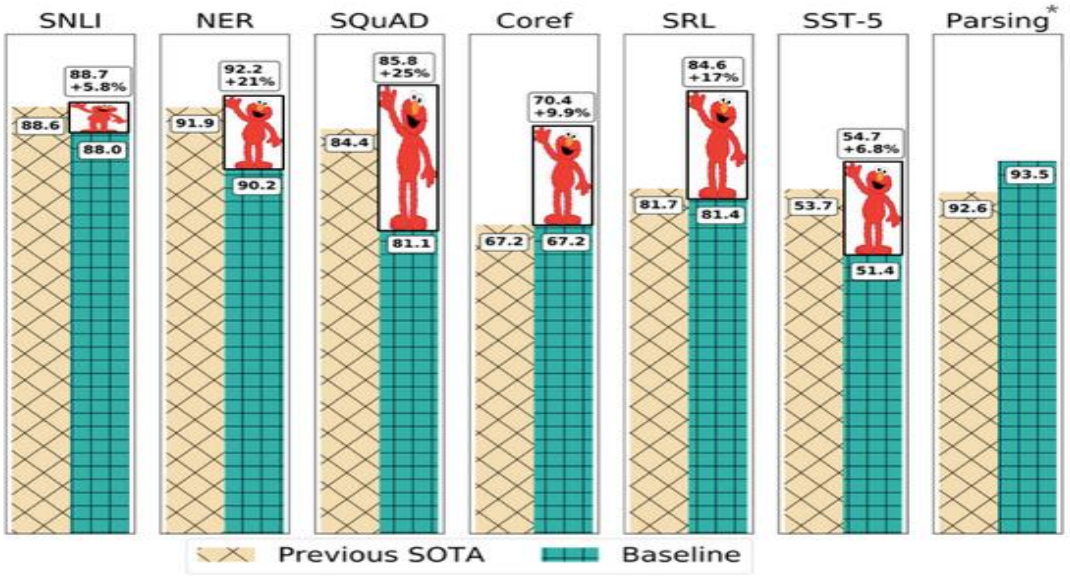
ABBYY

- Language model is pretrained on large corpus
- Language model architecture follows NLP pipeline introduced earlier
  - Token features are CharCNN (word-level embeddings are not used)
  - CharCNN are fed into two stacked Bidirectional RNNs
  - Top layers: Dense with relu followed by Dense with softmax
- Pretrained LM can be used in downstream tasks as follows:
  - The target sentence is fed into LM
  - Token representation is defined as follows: task-specific weighted sum of CharCNN and output of both BiRNNs

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}.$$

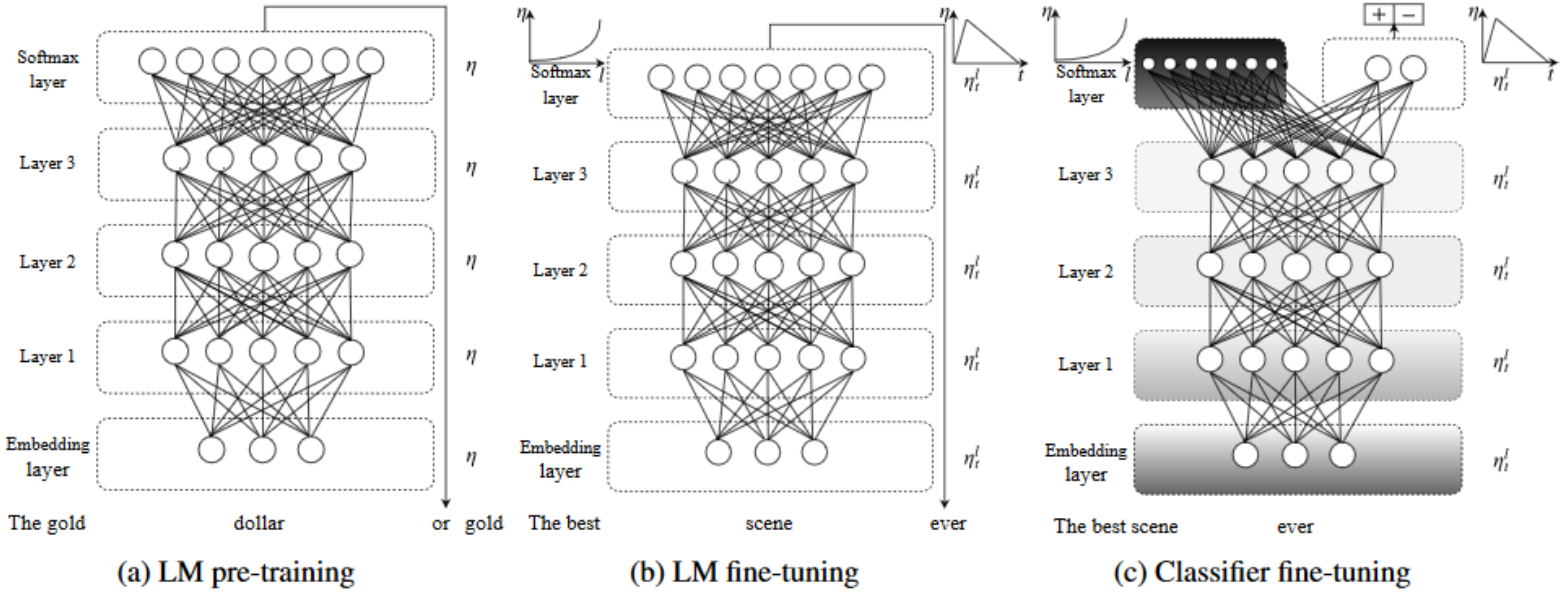
ELMo advantages

- Pretrained model is trained on a more complex task than word2vec and uses deeper and more sophisticated architecture
- Unlike word2vec the resulting representation is context dependent
- Model allows for large gains on most NLP tasks.
- Some researchers compare pretraining LMs to the creation of ImageNet for Computer Vision



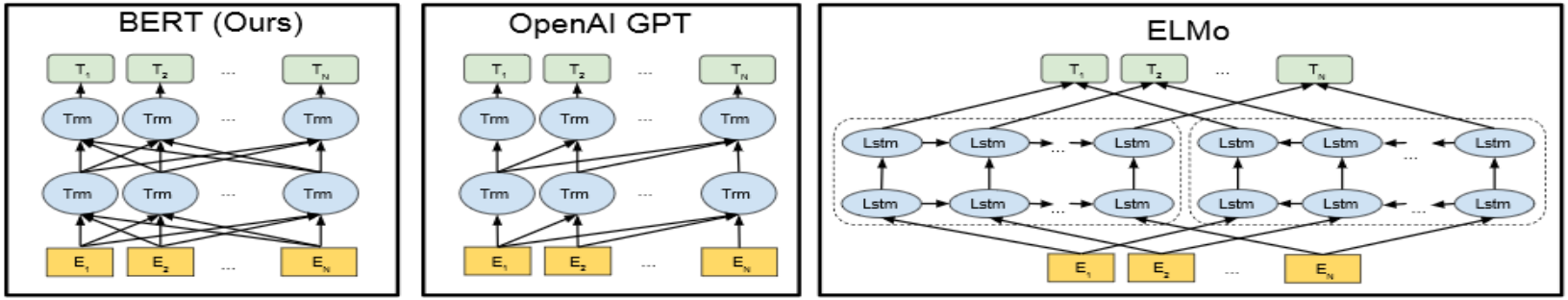
ULMFit

- ULMFit – SOTA for text classification task in 2018-2019

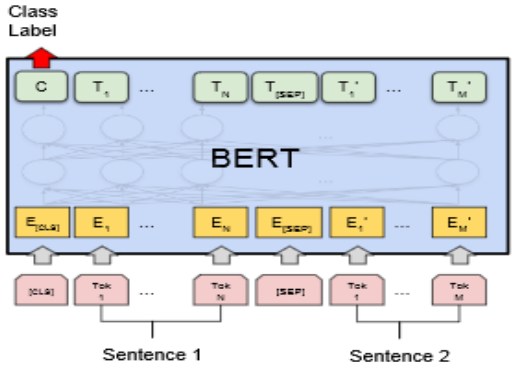


BERT

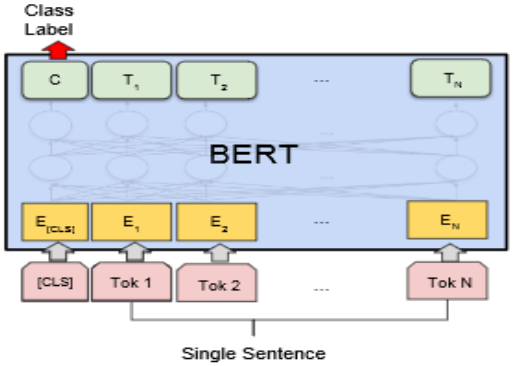
- BERT: Transformers instead of RNNs in language model architecture
- Pretrained on extremely large corpora on two tasks:
  - Masked language modeling
  - Next sentence prediction
- SOTA for many NLP tasks



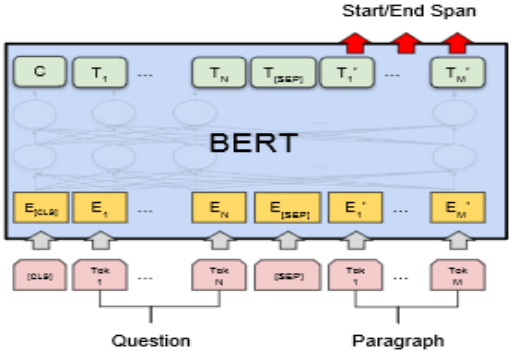
Using BERT on various tasks



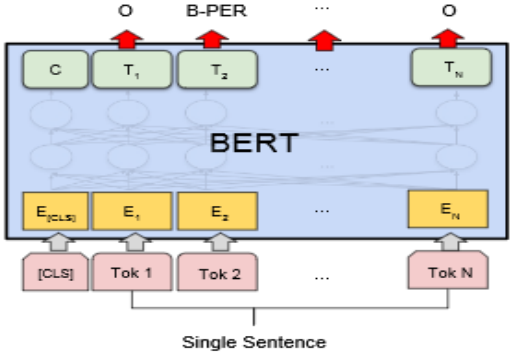
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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

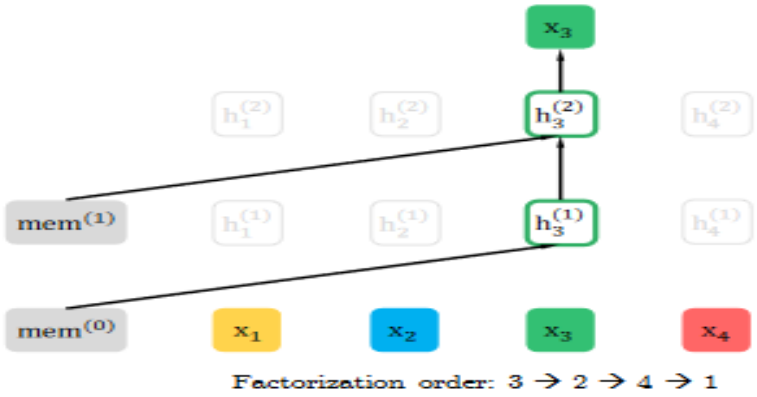


(c) Question Answering Tasks: SQuAD v1.1

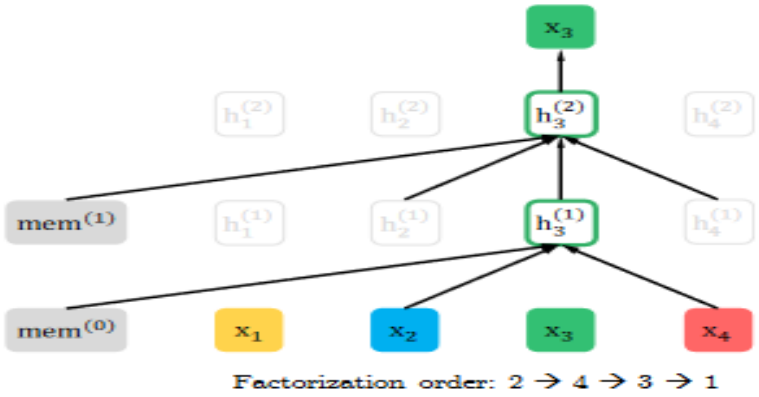


(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

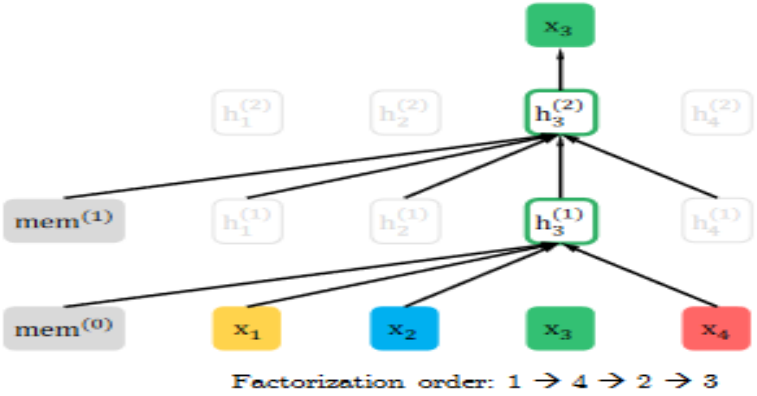
XLNet



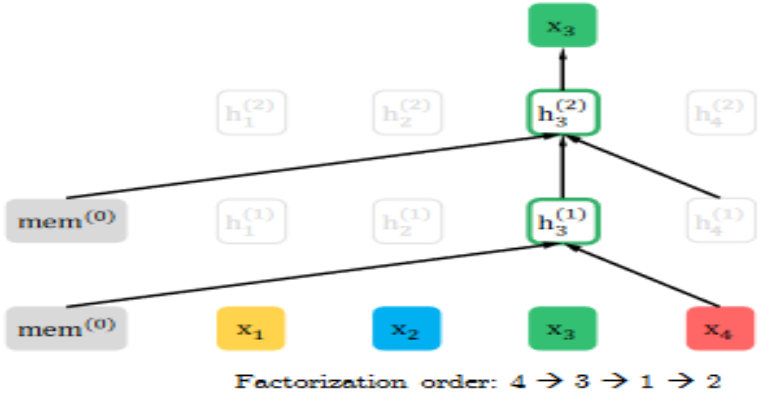
Factorization order: 3 → 2 → 4 → 1



Factorization order: 2 → 4 → 3 → 1



Factorization order: 1 → 4 → 2 → 3



Factorization order: 4 → 3 → 1 → 2

Language model embeddings bibliography

ABBYY

- ELMo: *Peters et al. (2018) Deep contextualized word representations*
- ULMFit: *Howard & Ruder (2018) Universal Language Model Fine-tuning for Text Classication*
- OpenAI Transformer: *Radford et al. (2018) Improving Language Understanding by Generative Pre-Training*
- BERT: *Devlin et al. (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*
- XLNet: *Yang et al. (2019) XLNet: Generalized Autoregressive Pretraining for Language Understanding*
- RoBERTA: *Liu et al. (2019) RoBERTa: A Robustly Optimized BERT Pretraining Approach*

Takeaways

ABBYY

- The language ambiguity on different levels is the main reason why natural language processing is challenging.
- Most NLP tasks can be solved by using Embeddings-CharCNN-BiRNN pipeline.
- Pretraining language models allows for huge performance boosts for most NLP tasks
- For many NLP tasks SOTA can be achieved by using transformer-based language models pretrained on a very large corpora

ABBYY

Thank you for attention

Any questions?

We are hiring: job@abbyy.com