Language Models

MIPT

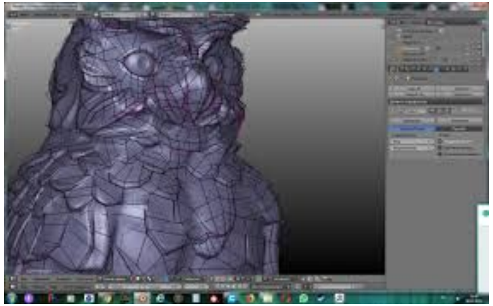
25.02.2021

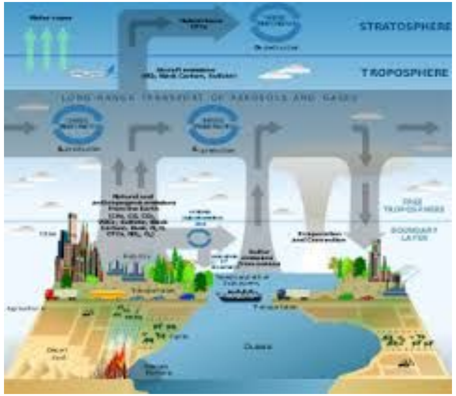
Anton Emelianov, Alena Fenogenova.

Task description

Language Modeling

- What does it mean to "model something"?



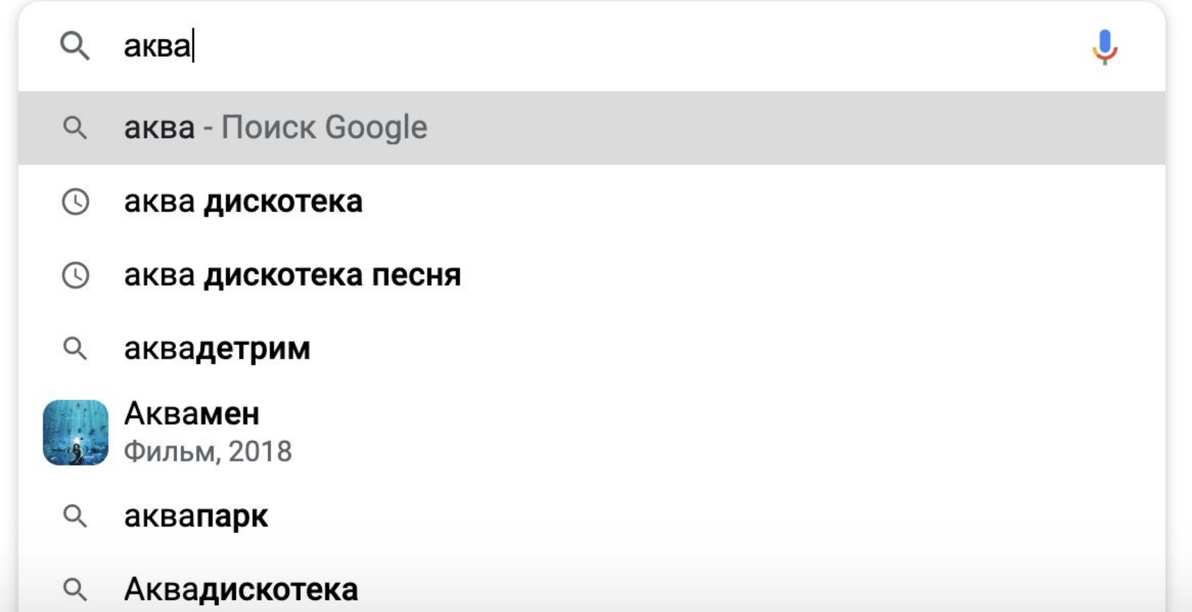


- **Language Models (LMs)** estimate the probability of different linguistic units:
  - symbols,
  - tokens,
  - token sequences.

- *But how can this be useful?*

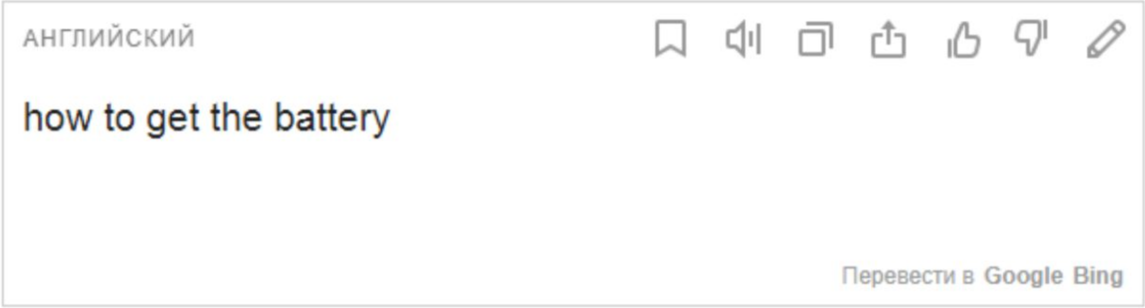
Where language models are used?

Search service



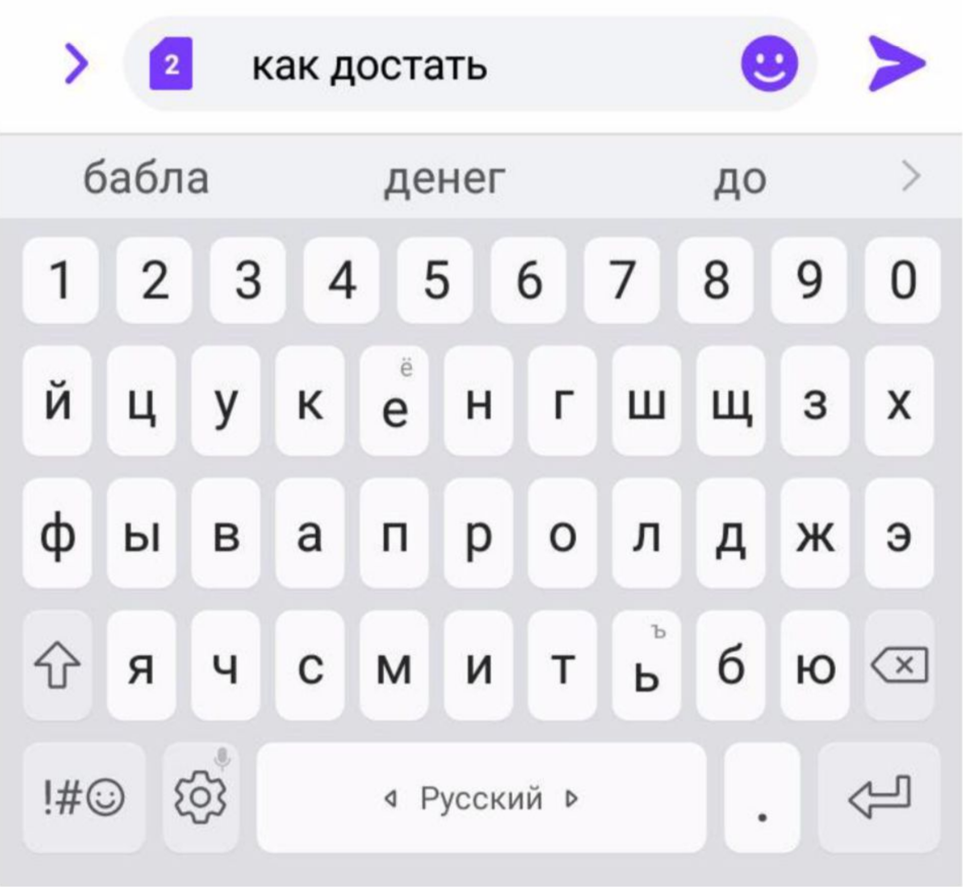
Where language models are used?

Translation service



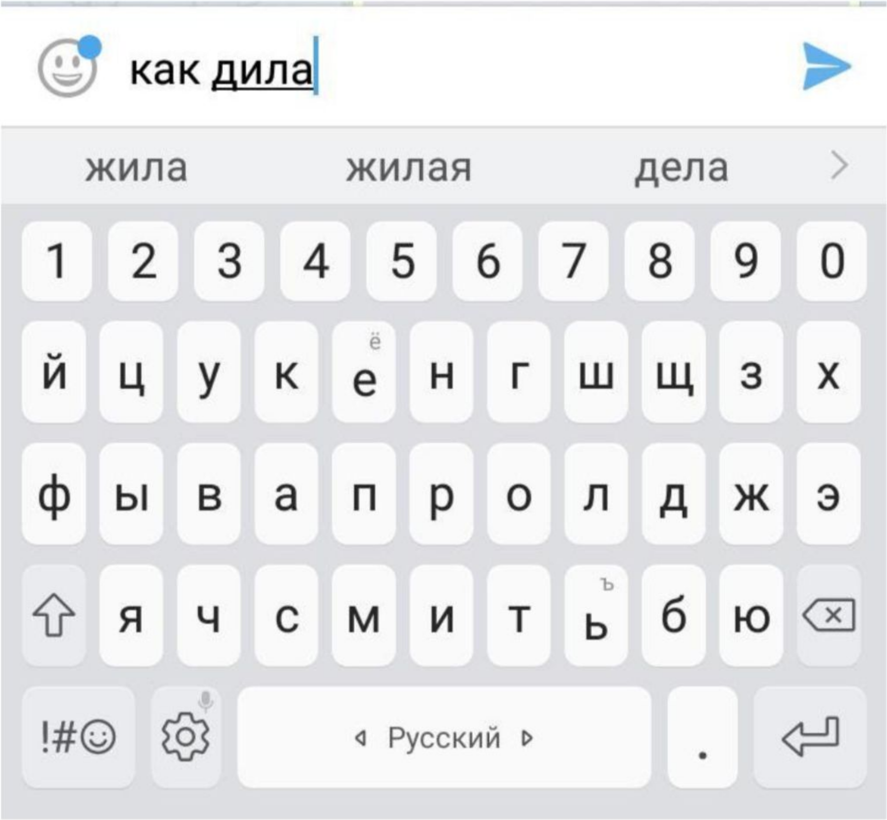
Where language models are used?

Mobile keyboard



Where language models are used?

Correction of typos



LMs

We deal with LMs every day!

- We can choose one option between the same sounding (similar) phrases:

Я хочу назвать моего кота наполеон.

Я хочу назвать моего кота на поле он.

Я хочу назвать моего кота Наполеон.

- For automatic models, the probability of the sentence will help in the solution.

Probability of sentence: Intuition

- "**Probability of a sentence**" = as much as possible in natural language.

- Only a specific language is considered:

P(Кот лежит на диване) > P(На диване кот лежит)

P(Красивая девочка играла в мяч) > P(В мяч играла девочка красивая)

LMs in NLP

- It is very difficult to know the real probability of a sequence of tokens.

- But we can use a language model to approximate this probability.

- Like all models, language models “behave well” in some cases and “badly” in others.

LMs

- Language models can be divided into two types:
  - **Count-based Models** (Statistical) - statistical language models.
  - **Neural Language Models** - language models based on neural networks.

N-gram LMs

N-gram LM: count-based LM

- How to **estimate** the **likelihood** of encountering a sentence within a specific language (eg Russian)?

$$s = (x^{(1)}, x^{(2)}, \dots, x^{(n)})$$

- Suppose we have training data: large text (corpus C) in Russian.
- And we have broken it down into sentences.
- We want to estimate the probability s using the available data.

N-gram LM: count-based LM

- How to **estimate** the **likelihood** of encountering a sentence within a specific language (eg Russian)?

$$s = (x^{(1)}, x^{(2)}, \ldots, x^{(n)})$$

- Suppose we have training data: large text (corpus C) in Russian.
- And we have broken it down into sentences.
- We want to estimate the probability s using the available data.

$$MLE : P(s) = \frac{Count(s \in C)}{|C|}$$

What is the probability to pick a green ball?

$$\frac{5}{5+6+4+3} = \frac{5}{18}$$

N-gram LM: count-based LM

- **What if the sentence s did not appear even once in the data?**

s_1=«Длина тела кархародонтозавра достигала 12 метров.»

VS

s_2=«достигала 12 метров.»

- The first sentence $s1$ makes more sense than the second one - $s2$.
- But the probability of the sentence $s1$ is $P(s1) = 0$, and the probability the second sentence $s2$ can be nonzero $P(s2) > 0$!

- The previous approach MLE does not work on full sentences.

N-gram LM: count-based LM

- What if the sentence s did not appear even once in the data?

N-gram LM: count-based LM

- **What if the sentence s did not appear even once in the data?**
- *Idea*:
  - Let's approximate the probability $P(s)$ by combining the probabilities of smaller parts of the sentence that are more common.

N-gram LM: count-based LM

- **What if the sentence s did not appear even once in the data?**
- *Idea*:
  - Let's approximate the probability $P(s)$ by combining the probabilities of smaller parts of the sentence that are more common.
- *Decision*:
  - The **N-gram Language Model**

N-gram LM: intro

- We want to estimate the probability of a sequence of words (tokens):

$$P(s = (x^{(1)}, x^{(2)}, \ldots, x^{(n)}))$$

  - Example: *«Я хочу назвать кота Наполеон»*
- This is the joint probability of meeting words (tokens) in the sentence s.
- Joint probability:

$$P(X, Y) = P(Y|X)P(X).$$

- Then

$$P(\text{Я хочу назвать кота Наполеон}) =$$
$$= P(\text{Наполеон}|\text{Я хочу назвать кота}) \times P(\text{Я хочу назвать кота}) =$$
$$= P(\text{Наполеон}|\text{Я хочу назвать кота}) \times P(\text{кота}|\text{Я хочу назвать}) \times$$
$$P(\text{назвать}|\text{Я хочу}) \times P(\text{хочу}|\text{Я}) \times P(\text{Я})$$

N-gram LM: intro

- **What problem remains?**

N-gram LM: intro

- **What problem remains?**
- *Chain rule:*

$$P\left(x^{(1)}, x^{(2)}, \ldots, x^{(n)}\right) = \prod_{i}^{n} P\left(x^{(i)} | x^{(1)}, \ldots, x^{(i-1)}\right)$$

- But many conditional probabilities are zero anyway!
- If we want to get the probability:

$$P(\text{Длина тела кархародонтозавра достигала 12 метров})$$

- You need to be able to:

$$P(\text{метров} | \text{Длина тела кархародонтозавра достигала 12})$$

N-gram LM: intro

- Let's make an **assumption about independence**: the probability of a word (token) depends only on a fixed number of previous words (**history**).
- **Markov assumption**: $P\left(x^{(i)} | x^{(1)}, \dots, x^{(i-1)}\right) = P\left(x^{(i)} | x^{(i-n+1)}, \dots, x^{(i-1)}\right)$
- **trigram model**: $P\left(x^{(i)} | x^{(1)}, \dots, x^{(i-1)}\right) \approx P\left(x^{(i)} | x^{(i-2)}, x^{(i-1)}\right)$
- **bigram model** $P\left(x^{(i)} | x^{(1)}, \dots, x^{(i-1)}\right) \approx P\left(x^{(i)} | x^{(i-1)}\right)$
- **unigram model** $P\left(x^{(i)} | x^{(1)}, \dots, x^{(i-1)}\right) \approx P\left(x^{(i)}\right)$

eight trigram model

N-gram LM

- Our assumption

$$P\left(x^{(i)}|x^{(1)},\dots,x^{(i-1)}\right)=P\left(x^{(i)}|\overbrace{x^{(i-n+1)},\dots,x^{(i-1)}}^{\text{n-1 words}}\right)=$$

- Definition of conditional probability

n-gram probability___

$$=\frac{\boxed{P\left(x^{(i-n+1)},\dots,x^{(i)}\right)}}{\boxed{P\left(x^{(i-n+1)},\dots,x^{(i-1)}\right)}}=$$

n-1 gram probability

- **Question**: how to get the probability of n-gram and (n-1) -gram?

N-gram LM

- **Question**: how to get the probability of n-gram and (n-1) -gram?

- **Answer**: Let's calculate using a large teaching text (corpus) that we have.
  - Statistical approximation

$$\approx \frac{Count\left(x^{(i-n+1)}, \dots, x^{(i)}\right)}{Count\left(x^{(i-n+1)}, \dots, x^{(i-1)}\right)}$$

Example: 4-gram model

~~Котик очень~~ $\underbrace{\text{тихо спал на}}_{\text{условия на этом}}$ ______

не участвует

условия на этом

$$P\left(x^{(i)} \middle| \text{Котик очень тихо спал на}\right) \approx \frac{Count\left(\text{тихо спал на } x^{(i)}\right)}{Count(\text{тихо спал на})}$$

Problems with n-gram models

$$P\left(x^{(i)} \middle| \text{Котик очень тихо спал на}\right) \approx \frac{Count\left(\text{тихо спал на } x^{(i)}\right)}{\boxed{Count\left(\text{тихо спал на}\right)}}$$

Problem: what if "quietly slept on" does not appear in the data. Then there is no way to calculate the probability $x(i)$!

Problems with n-gram models

$$P\left(x^{(i)} \middle| \text{Котик очень тихо спал на}\right) \approx \frac{Count\left(\text{тихо спал на } x^{(i)}\right)}{\boxed{Count(\text{тихо спал на})}}$$

Problem: what if "quietly slept on" does not appear in the data. Then there is no way to calculate the probability $x(i)$!

(Partial) **solution**: **backoff** smoothing.

Backoff smoothing

- Sometimes using less context helps

- **Backoff**:
  - We use trigram, if there is in the sample
  - otherwise bigram
  - otherwise unigram

- If not «тихо спал на», try «спал на»

$$P(x^{(i)}|\text{тихо спал на}) \approx P(x^{(i)}|\text{спал на})$$

- If not «спал на», try «на»

$$P(x^{(i)}|\text{спал на}) \approx P(x^{(i)}|\text{на})$$

- If not «на», try *x(i)*

$$P(x^{(i)}|\text{на}) \approx P(x^{(i)})$$

Linear interpolation

- Interpolation mixes unigram, bigram, trigram …

$$\hat{P}\left(x^{(i)} | x^{(i-3)}, x^{(i-2)}, x^{(i-1)}\right) \approx \lambda_3 P\left(x^{(i)} | x^{(i-3)}, x^{(i-2)}, x^{(i-1)}\right) + \lambda_2 P\left(x^{(i)} | x^{(i-2)}, x^{(i-1)}\right) + \lambda_1 P\left(x^{(i)} | x^{(i-1)}\right) + \lambda_0 P\left(x^{(i)}\right)$$

$$\sum_{i=0}^{n-1} \lambda_i = 1$$

- **Question**: how to choose λ_i?

Linear interpolation

- Interpolation mixes unigram, bigram, trigram …

$$\hat{P}\left(x^{(i)} | x^{(i-3)}, x^{(i-2)}, x^{(i-1)}\right) \approx \lambda_3 P\left(x^{(i)} | x^{(i-3)}, x^{(i-2)}, x^{(i-1)}\right) + \lambda_2 P\left(x^{(i)} | x^{(i-2)}, x^{(i-1)}\right) + \lambda_1 P\left(x^{(i)} | x^{(i-1)}\right) + \lambda_0 P\left(x^{(i)}\right)$$

$$\sum_{i=0}^{n-1} \lambda_i = 1$$

- **Question**: how to choose λ_i?
- **Answer**: use a validation dataset.

Problems with n-gram models

Problem: what if "тихо спал на x (i)" did not occur in the data. Then the probability $x(i)$ is 0!

(Partial) **solution**: Various smoothing (add-one, Kneser-Ney, et al.)

$$P(x^{(i)} | \text{Котик очень тихо спал на}) \approx \frac{\boxed{Count(\text{тихо спал на } x(i))}}{\boxed{Count(\text{тихо спал на})}}$$

Problem: what if "тихо спал на" did not occur in the data. Then there is no way to calculate the probability $x(i)$!

(Partial) **solution**: Backoff smoothing

1. Kneser-Ney smoothing, URL: https://en.wikipedia.org/wiki/Kneser%E2%80%93Ney_smoothing

Laplace smoothing (add-one)

- Suppose that each word (n-gram) occurs at least 1 time.
- Add 1 to the numerator and denominator.
- If 1 is too rough an estimate, then add δ for each word $x(i) \in V$.

$$\hat{P}\left(x^{(i)} | x^{(i-n+1)}, \dots, x^{(i)}\right) = \frac{\delta + P\left(x^{(i-n+1)}, \dots, x^{(i)}\right)}{\delta|V| + P\left(x^{(i-n+1)}, \dots, x^{(i-1)}\right)}$$

How to evaluate language models?

- **Internal assessment**:
  - Cross-entropy:

$$H_M(w_1 w_2 \dots w_n) = -\frac{1}{n} \cdot \log P_M(w_1 w_2 \dots w_n)$$

  - shows how well the model is able to predict the next word.
  - Perplexity (stated in articles):

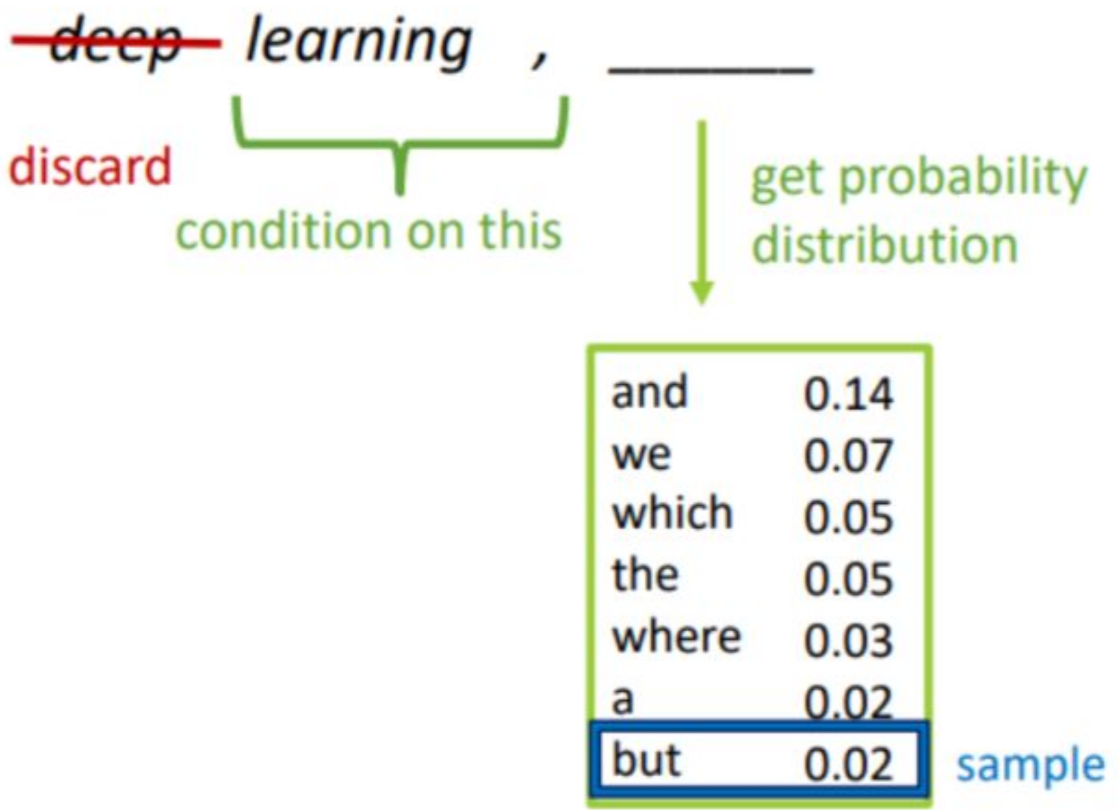
$$\text{perplexity} = 2^{cross-entropy}$$

- **External assessment**:
  - A specific task: we replace one language model with another and look at the quality metric of this task.

LMs

- Language models can be divided into two types:
  - **Count-based Models** (Statistical) - statistical language models.
    - Markov assumption of order n;
    - Approximation of probabilities n-gram (counting and smoothing).
  - **Neural Language Models** - language models based on neural networks.

How to generate text using N-gram LMs?



How to generate text using N-gram LMs?

deep learning , but also is central to human performance . however , using structural similarity index measure than other partitioned sampling schemes , while making the approach with empirical data has the effect of phonetics has received little attention within the context of information on ...

How to generate text using N-gram LMs?

deep learning , but also is central to human performance . however , using structural similarity index measure than other partitioned sampling schemes , while making the approach with empirical data has the effect of phonetics has received little attention within the context of information on ...

What's wrong with this text?

How to generate text using N-gram LMs?

deep learning , but also is central to human performance . however , using structural similarity index measure than other partitioned sampling schemes , while making the approach with empirical data has the effect of phonetics has received little attention within the context of information on ...

What's wrong with this text?

It's completely meaningless!

Neural LMs

Neural Language Models (NLM)

Fixed Window Neural Language Models

- Final distribution:

$$\hat{y} = softmax(Uh + b_2) \in R^{|V|}$$

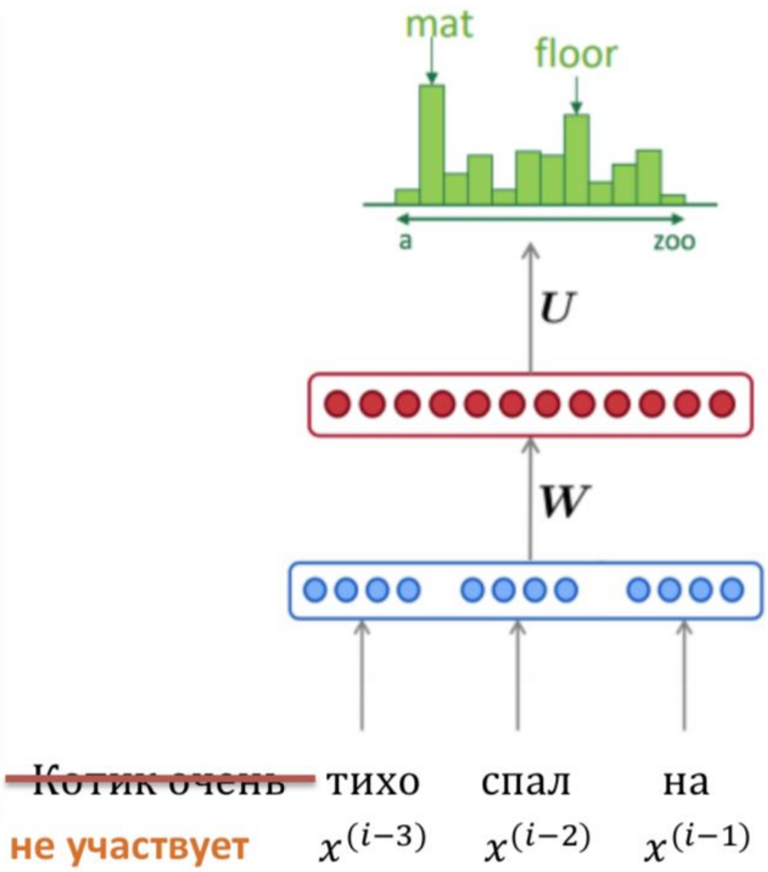
- Hidden layer (or other feed-forward NN)

$$h = f(Wx + b_1)$$

- Concatenation of embeddings

$$x = \left(x^{(i-3)}, x^{(i-2)}, x^{(i-1)}\right)$$

- Word embeddings



Neural Language Models (NLM)

Fixed Window Neural Language Models

- What are the **improvements** in comparison with N-gram LM:

Neural Language Models (NLM)

Fixed Window Neural Language Models

- What are the **improvements** in comparison with N-gram LM:
  - No sparsity problem
  - Model size O (V) (instead of O (exp (V))

Neural Language Models (NLM)

Fixed Window Neural Language Models

- What are the **improvements** in comparison with N-gram LM:
  - No sparsity problem
  - Model size O (V) (instead of O (exp (V))
- What **problems** remained:

Neural Language Models (NLM)

Fixed Window Neural Language Models

- What are the **improvements** in comparison with N-gram LM:
  - No sparsity problem
  - Model size O (V) (instead of O (exp (V))
- What **problems** remained:
  - Fixed size windows - very small, cannot be changed
  - Fixed word order
  - Weights can only be used inside the window

Neural Language Models (NLM)

Recurrent Language Models (RNN LM)

- Final distribution:

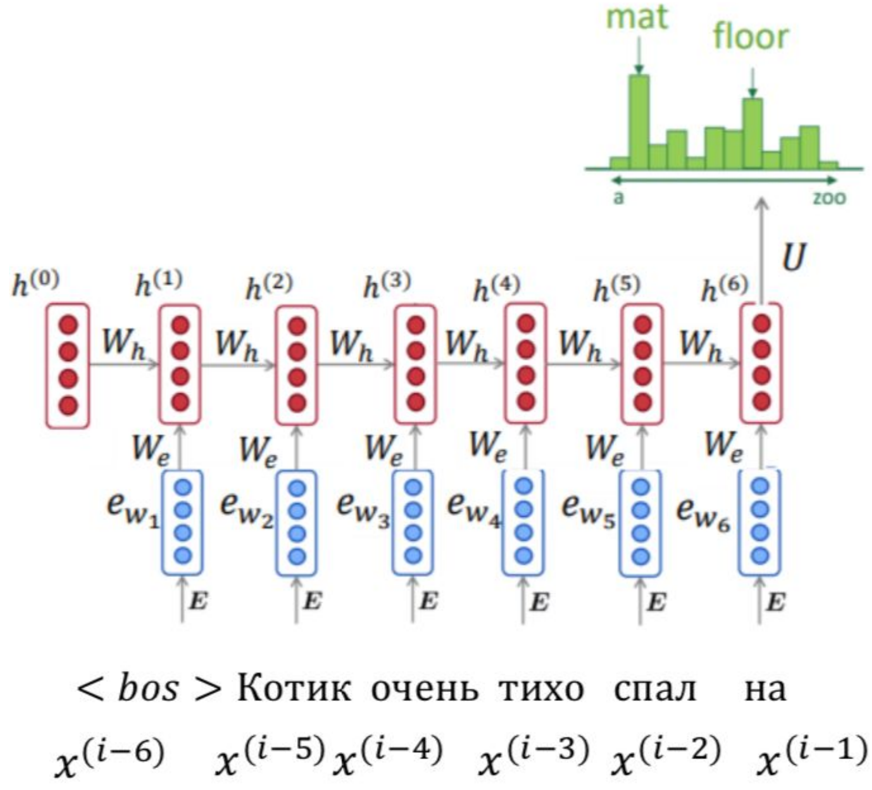
$$\hat{y} = softmax(Uh + b_2) \in R^{|V|}$$

- Hidden states

$$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_x e^{(t-1)} + b_1\right)$$

- Word embeddings

- Words



Neural Language Models (NLM)

Recurrent Language Models (RNN LM)

- What are the **improvements** in comparison with N-gram LM:

Neural Language Models (NLM)

Recurrent Language Models (RNN LM)

- What are the **improvements** in comparison with N-gram LM:
  - We can process sequences of any length
  - Model size does not depend on the size of the input
  - Calculations for step t (in theory) depend on the set of previous steps
  - Words representations depend on steps

Neural Language Models (NLM)

Recurrent Language Models (RNN LM)

- What are the **improvements** in comparison with N-gram LM:
  - We can process sequences of any length
  - Model size does not depend on the size of the input
  - Calculations for step t (in theory) depend on the set of previous steps
  - Words representations depend on steps
- What **problems** remained:

Neural Language Models (NLM)

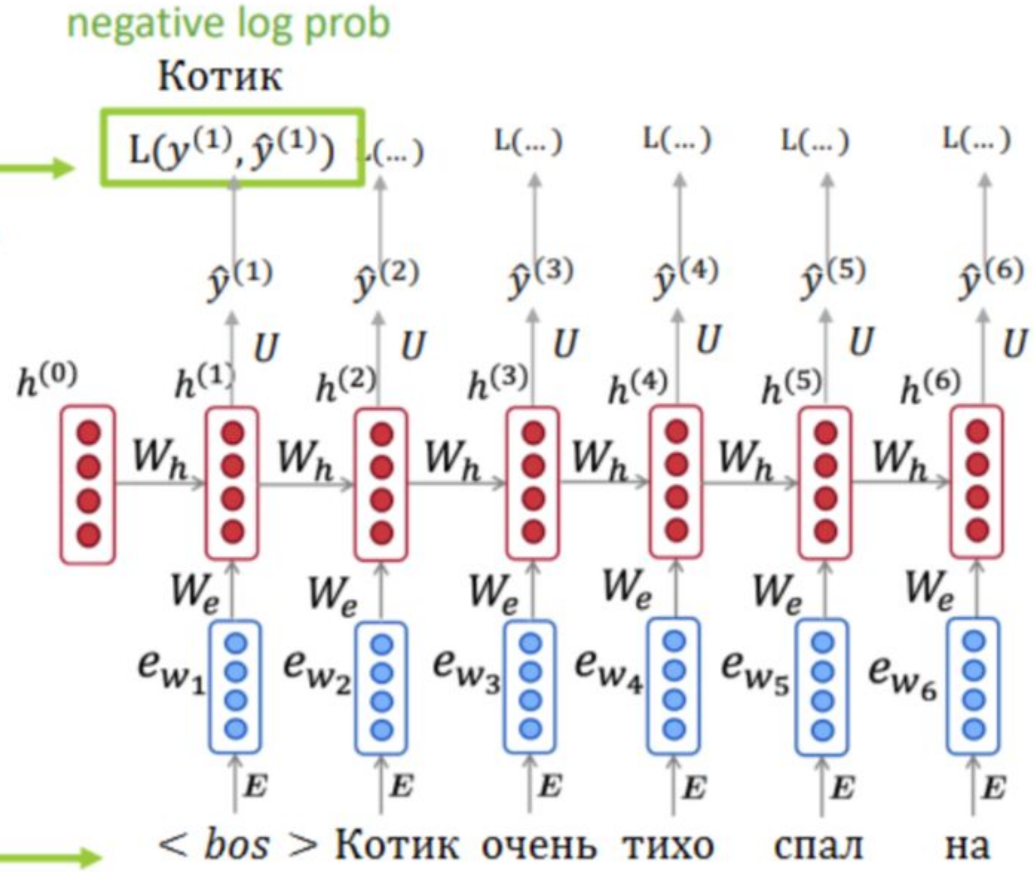
Recurrent Language Models (RNN LM)

- What are the **improvements** in comparison with N-gram LM:
  - We can process sequences of any length
  - Model size does not depend on the size of the input
  - Calculations for step t (in theory) depend on the set of previous steps
  - Words representations depend on steps
- What **problems** remained:
  - Computing may be slow
  - In practice, the network may not have access information many steps back

Training recurrent language models

Loss

$$L(y^{(t)}, \hat{y}^{(t)}) = \sum_{j=1}^{|V|} y_j^{(t)} \log \hat{y}_j^{(t)}$$

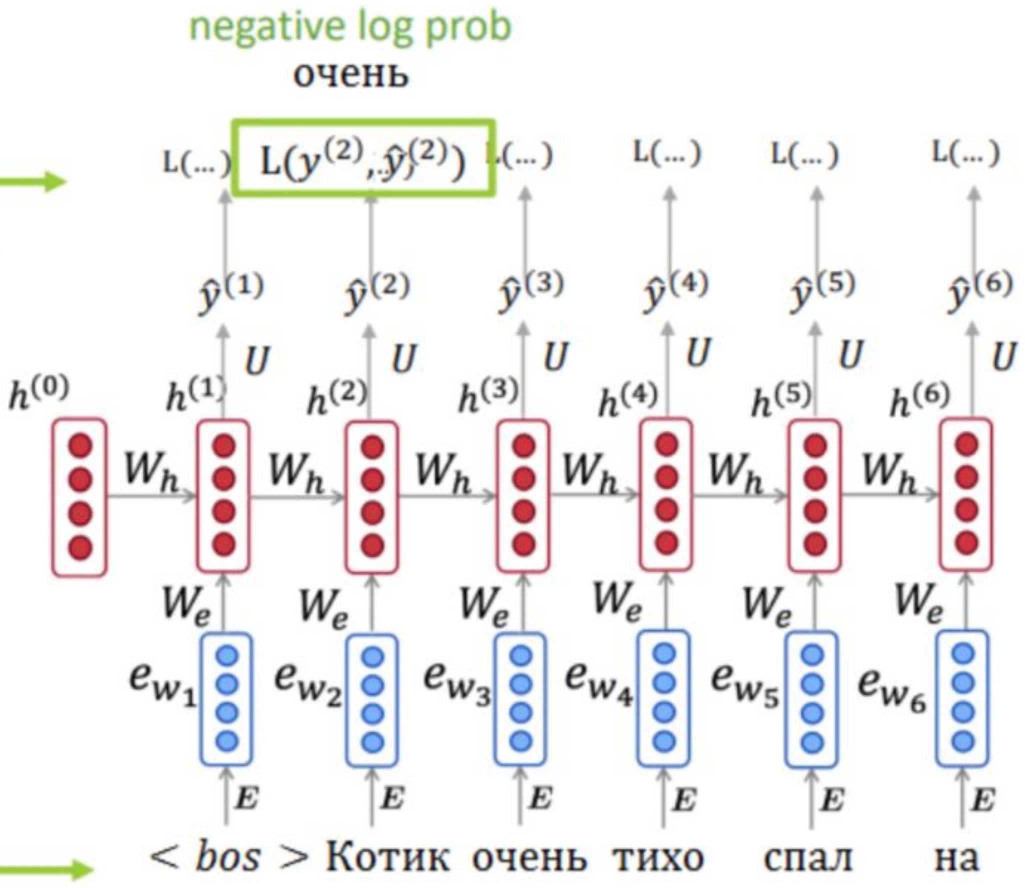


corpus

Training recurrent language models

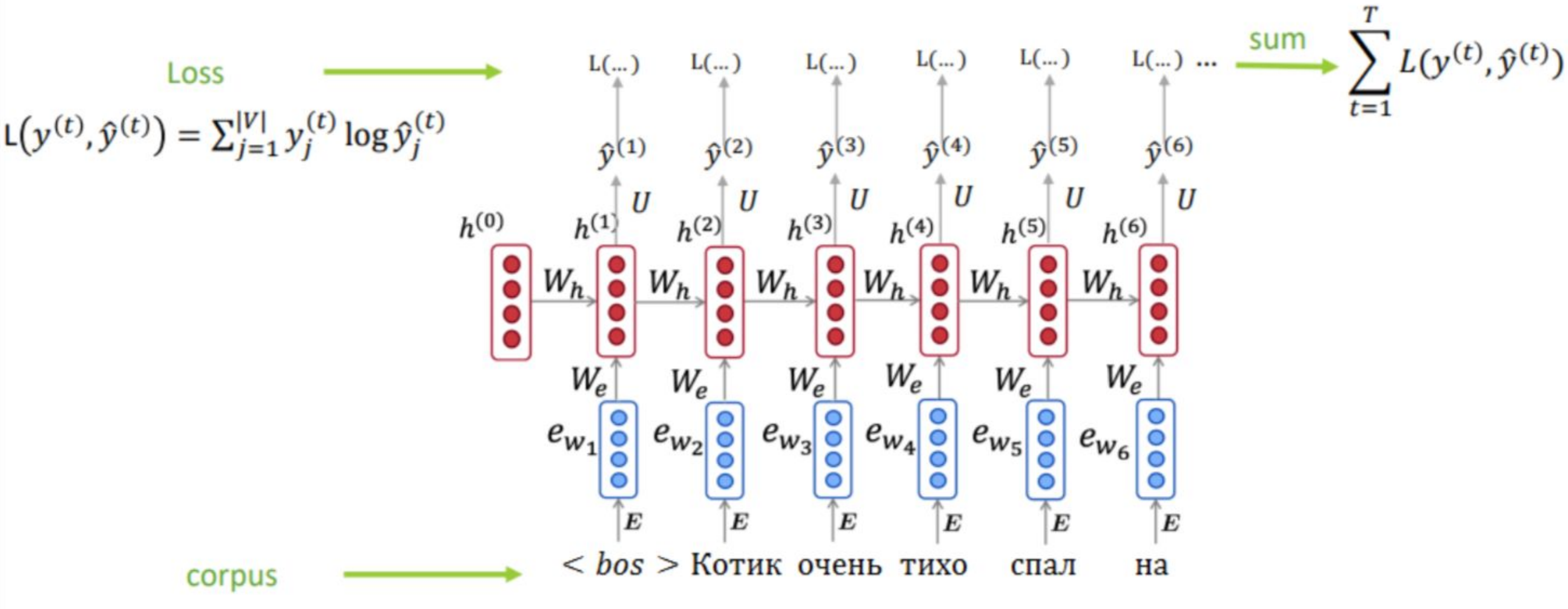
Loss

$$L(y^{(t)}, \hat{y}^{(t)}) = \sum_{j=1}^{|V|} y_j^{(t)} \log \hat{y}_j^{(t)}$$



corpus

Training recurrent language models

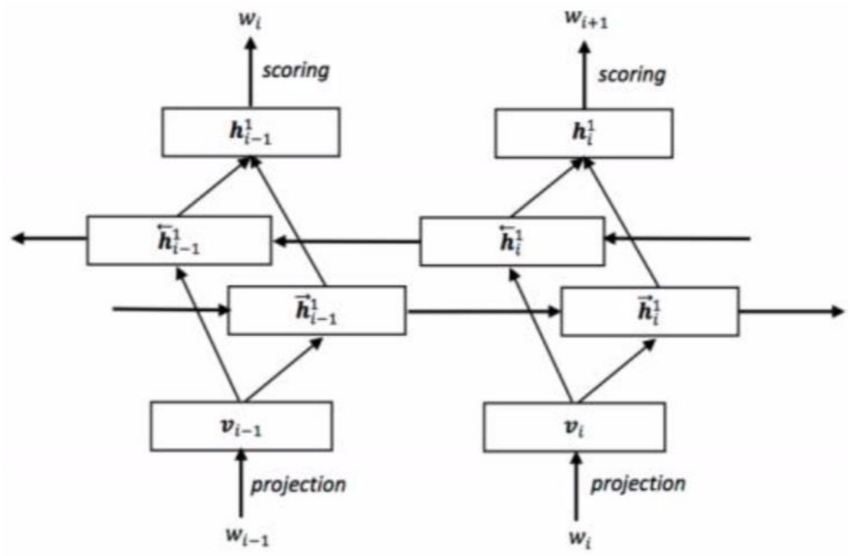


LMs

- Language models can be divided into two types:
  - **Count-based Models** (Statistical) - statistical language models.
    - Markov assumption of order n;
    - Approximation of probabilities n-gram (counting and smoothing).
  - **Neural Language Models** - language models based on neural networks.
    - Solved the problem of sparseness of the N-gram model by representing words using vectors? - Partially.
    - Word parameters are part of the learning process for the model.

Bidirectional language models

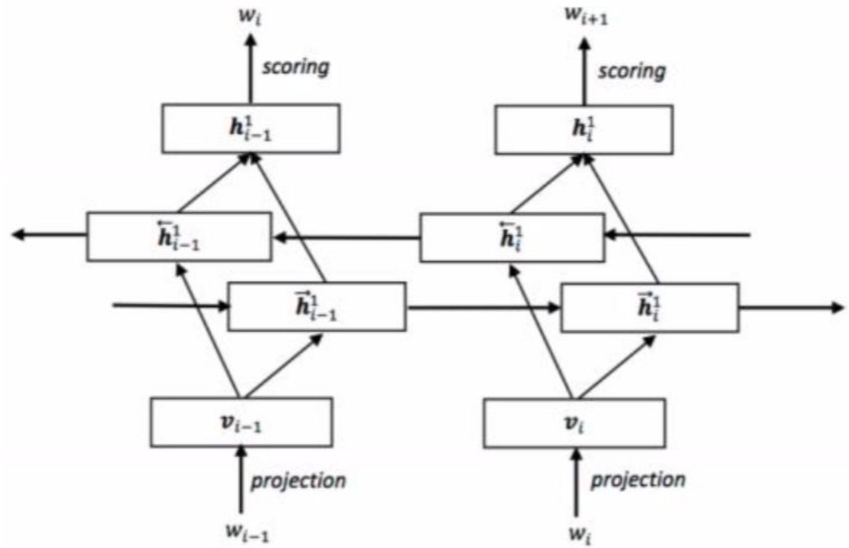
- We can use both left and right contexts.



1. On Training Bi-directional Neural Network Language Model with Noise Contrastive Estimation 55 https://arxiv.org/pdf/1602.06064.pdf

Bidirectional language models

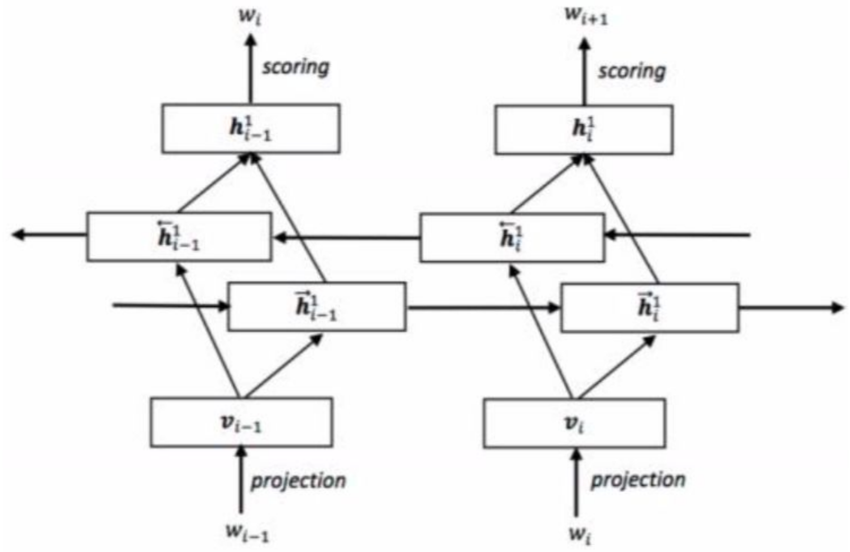
- We can use both left and right contexts.
  - Cannot be used for generation. 
  - When to use such a language model?



1. On Training Bi-directional Neural Network Language Model with Noise Contrastive Estimation 55 https://arxiv.org/pdf/1602.06064.pdf

Bidirectional language models

- We can use both left and right contexts.
  - Cannot be used for generation. 
  - **When to use** such a language model?- in other tasks , for example, correcting typos, NERs, etc. 



1. On Training Bi-directional Neural Network Language Model with Noise Contrastive Estimation 55 https://arxiv.org/pdf/1602.06064.pdf

Convolutional language models

- Also a fixed window!
- But instead of concatenation, **convolutional** layers are used.

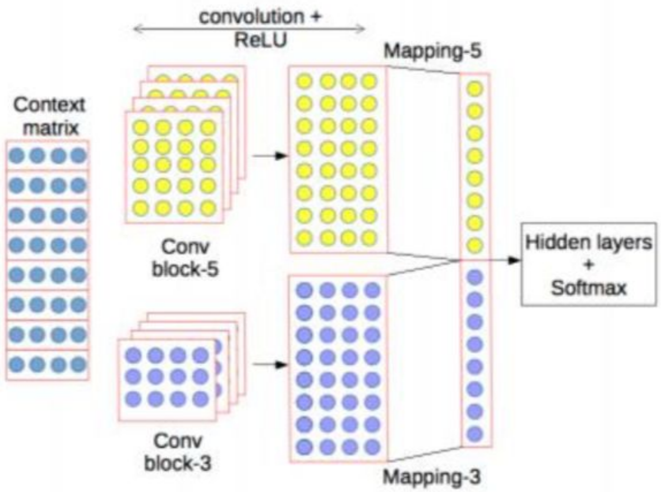


Figure 3: Combining kernels with different sizes. We concatenate the outputs of 2 convolutional blocks with kernel size of 5 and 3 respectively.

1. Pham et al, EMNLP 2016, https://www.aclweb.org/anthology/D16-1123

Elmo - Embeddings from Language Models

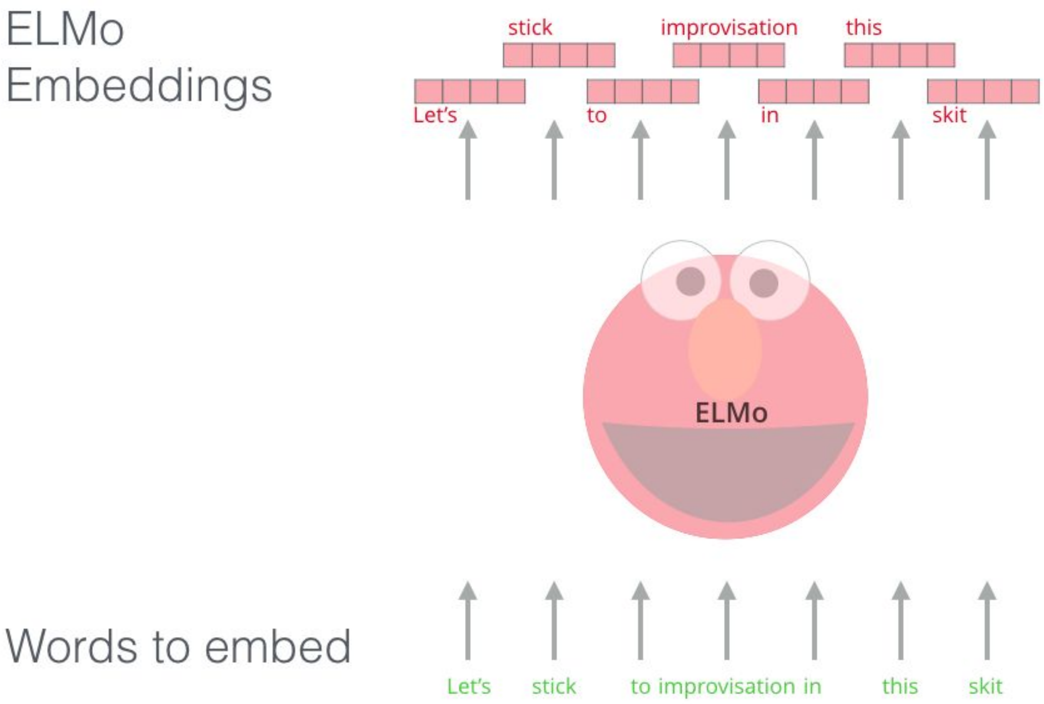
- Deep contextualized word representations (**ELMO**).

(ELMO – Embeddings from Language MOdel)

Deep contextualized word representations https://arxiv.org/pdf/1802.05365.pdf

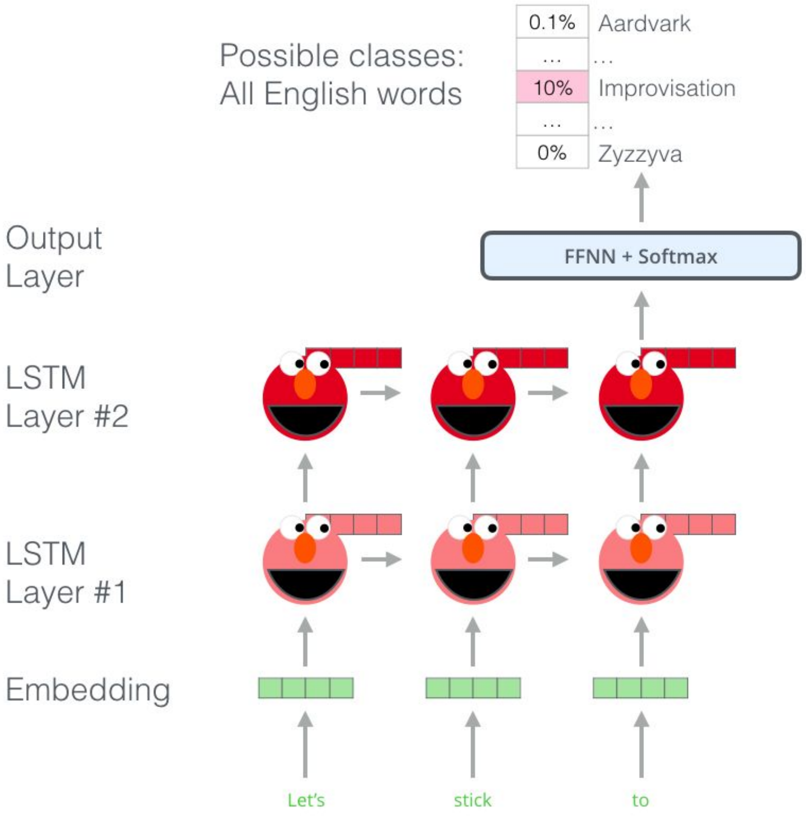
Elmo - Embeddings from Language Models

- Full word embedding: concatenation of word embedding and char-cnn.



Elmo - Embeddings from Language Models

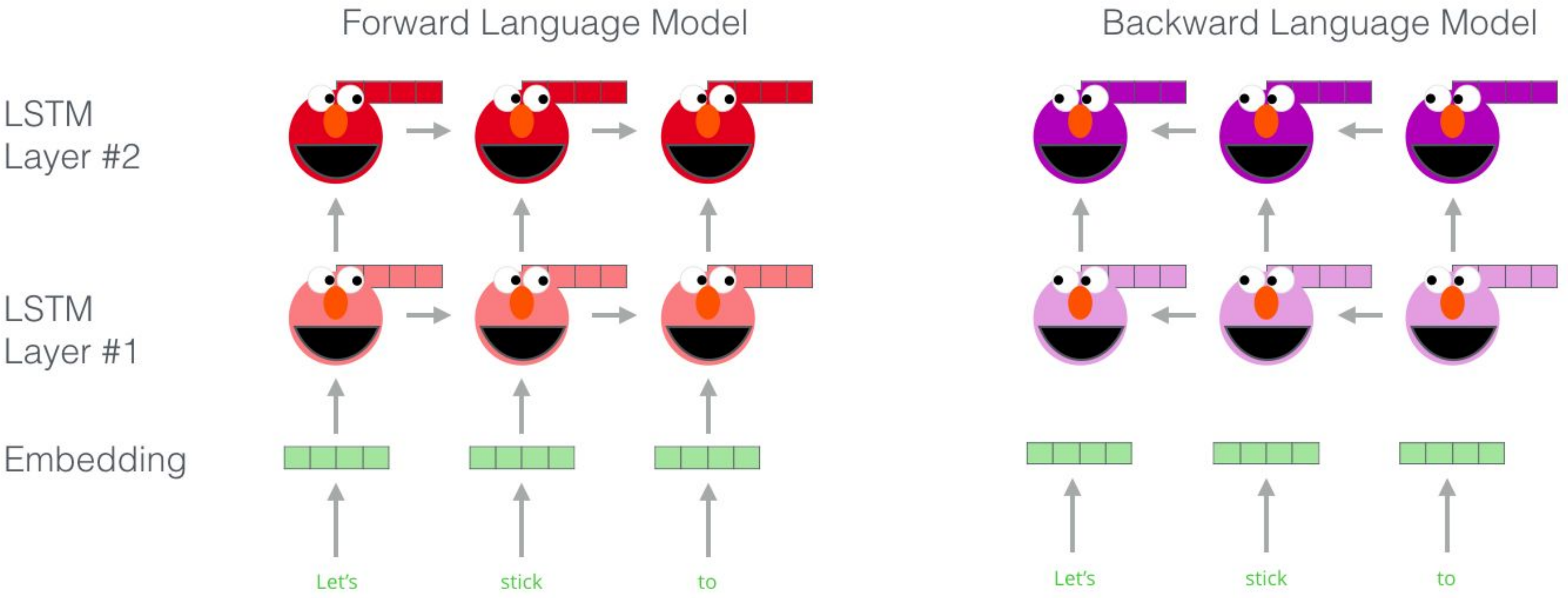
- We train Bi-LSTM on a large dataset.



Elmo - Embeddings from Language Models

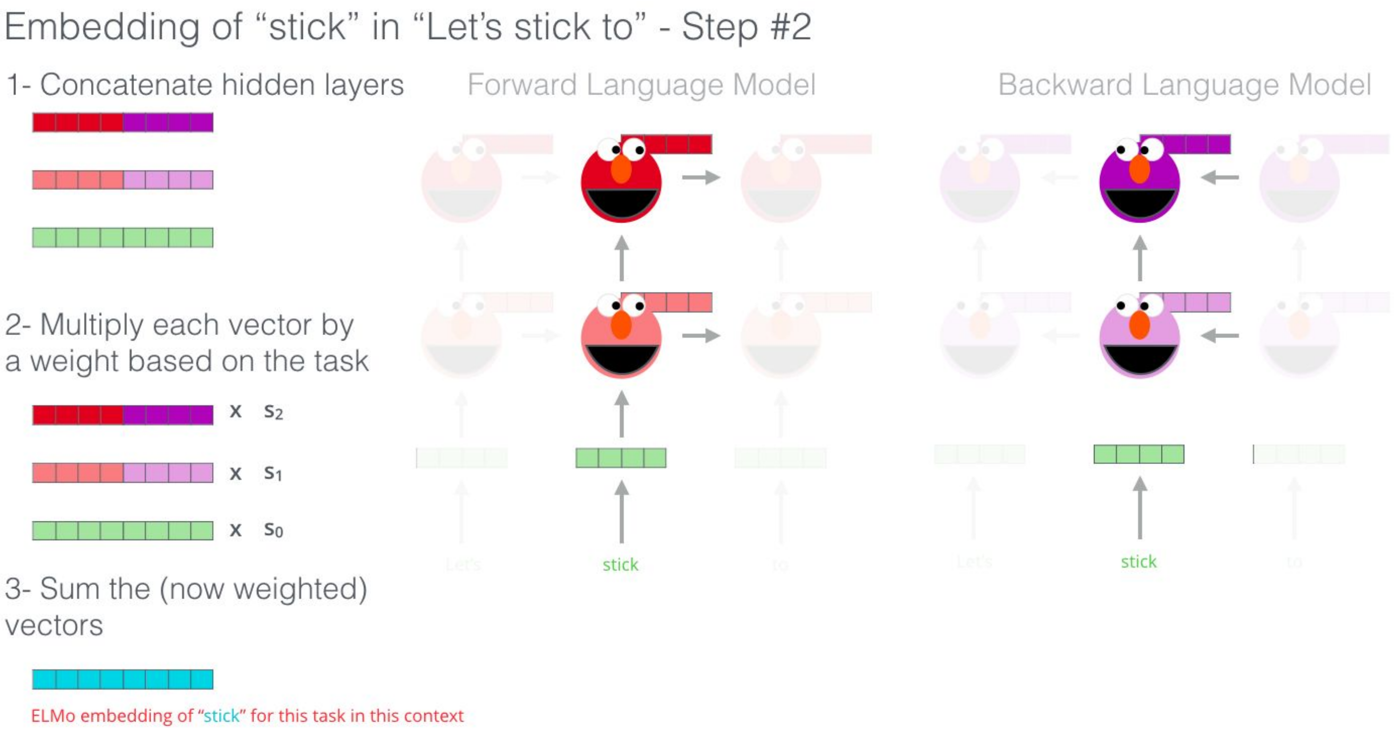
- ELMo actually goes a step further and trains a bi-directional LSTM – so that its language model doesn't only have a sense of the next word, but also the previous word.

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



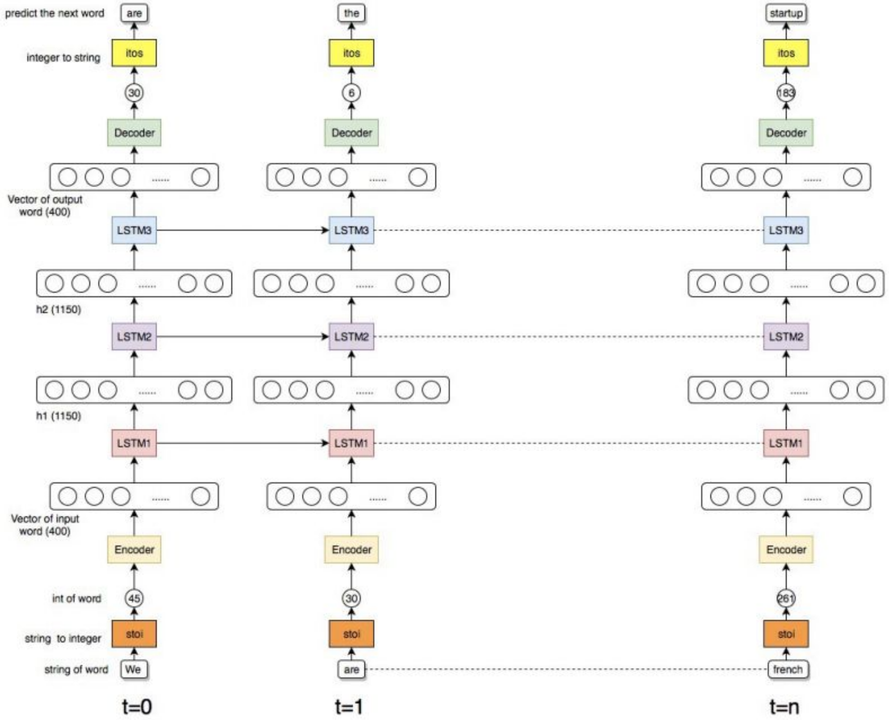
Elmo - Embeddings from Language Models

- ELMo comes up with the contextualized embedding through grouping together the hidden states (and initial embedding) in a certain way (concatenation followed by weighted summation). Learn s0, s1, s2 on other task.



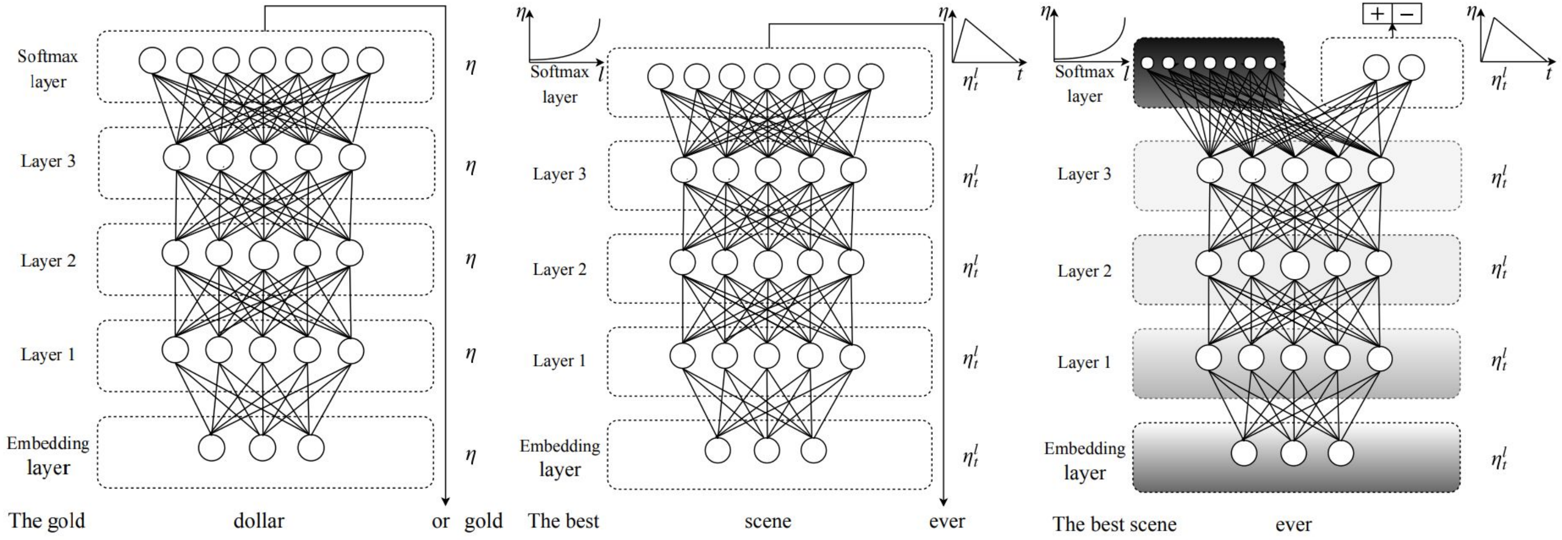
Universal Language Model Fine-tuning (ULMFiT).

- Universal Language Model Fine-tuning for Text Classification



1. Universal Language Model Fine-tuning for Text Classification https://arxiv.org/pdf/1801.06146.pdf.

Universal Language Model Fine-tuning (ULMFiT).



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

Figure 1: ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning (‘*Discr*’) and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, ‘*Discr*’, and STLR to preserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen).

1. Universal Language Model Fine-tuning for Text Classification https://arxiv.org/pdf/1801.06146.pdf.

Universal Language Model Fine-tuning (ULMFiT).

- STRL - Slanted triangular learning rates

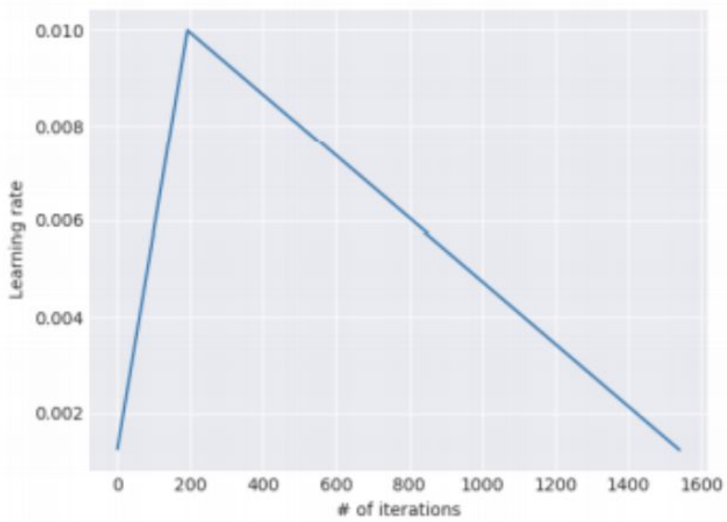


Figure: STRL example.

$$cut = \lfloor T \cdot cut\_frac \rfloor$$

$$p = \begin{cases} t/cut, & \text{if } t < cut \\ 1 - \frac{t - cut}{cut \cdot (1/cut\_frac - 1)}, & \text{otherwise} \end{cases}$$

$$\eta_t = \eta_{max} \cdot \frac{1 + p \cdot (ratio - 1)}{ratio}$$

Where,

- T is number of training iterations
- cut_frac is the fraction of iterations
- cut is the iteration when we switch from increasing to decreasing the LR
- p is the fraction of the number of iterations we have increased or will decrease the LR respectively
- ratio specifies how much smaller the lowest LR is from the maximum LR η_{max}
- η_t is the learning rate at iteration t

Universal Language Model Fine-tuning (ULMFiT).

- hc = [hT, maxpool(H), meanpool(H)]

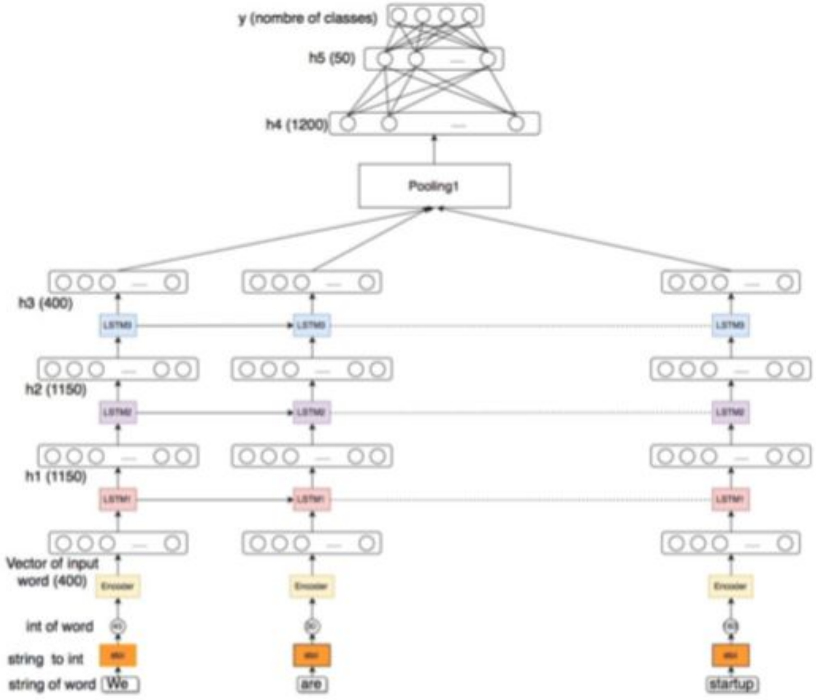


Figure: Classification architecture.

NER/POS

NER/POS task

- Part of speech tagging - Given a sentence or a sequence of words (X), predict its part of speech sequence (Y)

X (words)	the	cat	sat	on	a	mat
Y (POS-tags):	DET	NOUN	VERB	PREP	DET	NOUN

- Pointwise prediction: choose a POS-tag for a word individually
- Sequence models:
  - Generative models: P(y,x)
  - Discriminative models: P(y|x)

NER/POS task

- Generative sequence models

$$\arg\max_{Y} P(Y|X) = \arg\max \frac{P(X|Y)P(Y)}{P(X)} \approx \arg\max P(X|Y)P(Y)$$

- $P(X|Y)$ models word/ P OS tag interactions
- $P(Y)$ models POS / POS interactions

Hidden Markov Models

An HMM is specified by the following components:

$Q = q_1, \ldots, q_T$	states (POS-tags)
$A = (a_{ij})$	transition probability matrix: $a_{ij} = P(Q_i \rightarrow Q_j)$
$O = o_1, \ldots, o_V$	observations (words)
B	emission probabilities
	$b_i(o_t)$ is the probability of q_i generate o_t
$\pi = \pi_1, \ldots, \pi_N$	initial probability distribution

Probabilities should sum to unity:

$\sum_j a_{ij} = 1$

$\sum_i \pi_i = 1$

Hidden Markov Models

- Markov assumptions

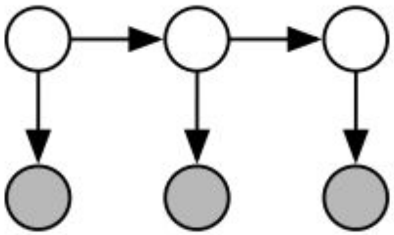
1. The probability of a particular state depends only on the previous state:

$$P(q_i|q_1,\ldots,q_{i-1}) = P(q_i|q_{i-1})$$

2. Output Independence: the probability of an output observation o_i depends only on the state that produced the observation q_i:

q

o



HMMs

$$P(o_i|Q,O) = P(o_i|q_i)$$

Hidden Markov Models

Three tasks of HMM

1. Likelihood: given an observation sequence, estimate the likelihood of the observation sequence.
2. Decoding: given an observation sequence, discover the best hidden state sequence leading to these observations.
3. Learning: train HMM.

Hidden Markov Models

- Forward-backward algorithm

$O_n = o_1, \ldots, o_n$
Forward probabilities: $\alpha_{ij} = P(o_1, \ldots, o_i)$
Forward algorithm:

1. $\alpha_{1j} = a_{0j} b_j(o_1), 1 \leq j \leq |Q|$
2. $\alpha_{ij} = \sum_k^Q \alpha_{i-1,k} a_{kj} b_j(o_i), 1 \leq i \leq n, 1 \leq j \leq |Q|$
3. $P(O_n) = \sum_k^Q \alpha_{nk} a_{kF}$

Backward probabilities: $\beta_{oj} = P(o_{i+1}, \ldots, o_n)$
Backward algorithm:

1. $\beta_{nj} = a_{jF}, 1 \leq j \leq |Q|$
2. $\alpha_{ij} = \sum_k^Q \beta_{i+1,k} a_{jk} b_k(o_{i+1}), 1 \leq i \leq n, 1 \leq j \leq |Q|$
3. $P(O_n) = \sum_k^Q a_{0k} b_(o_1) \beta$

Hidden Markov Models

- Decoding

Input: HMM $= (A, B)$, observations $= o_1, \ldots, o_n$
Output: the most probable sequence of states $= q_1, \ldots, q_n$

$$\hat{q}_n = \arg\max_{q_n} P(q_n|o_n) \approx$$

$$\approx \max_{q_n} \prod_{i=1}^{n} P(o_i|q_i)P(q_i|q_{i-1})$$

Hidden Markov Models

- Viterbi algorithm

Compute path probabilities $V = |n \times T|$. v_{ij} represents the probability that the HMM is in state j after seeing the first i observations.

1. Intialize

$$v_{1j} = a_{0j} b(o_1), 1 \leq j \leq T$$

2. Recursion

$$v_{ij} = \max v_{i-1,k} a_{kj} b_j(o_i), 1 \leq i \leq n, 1 \leq j \leq T$$

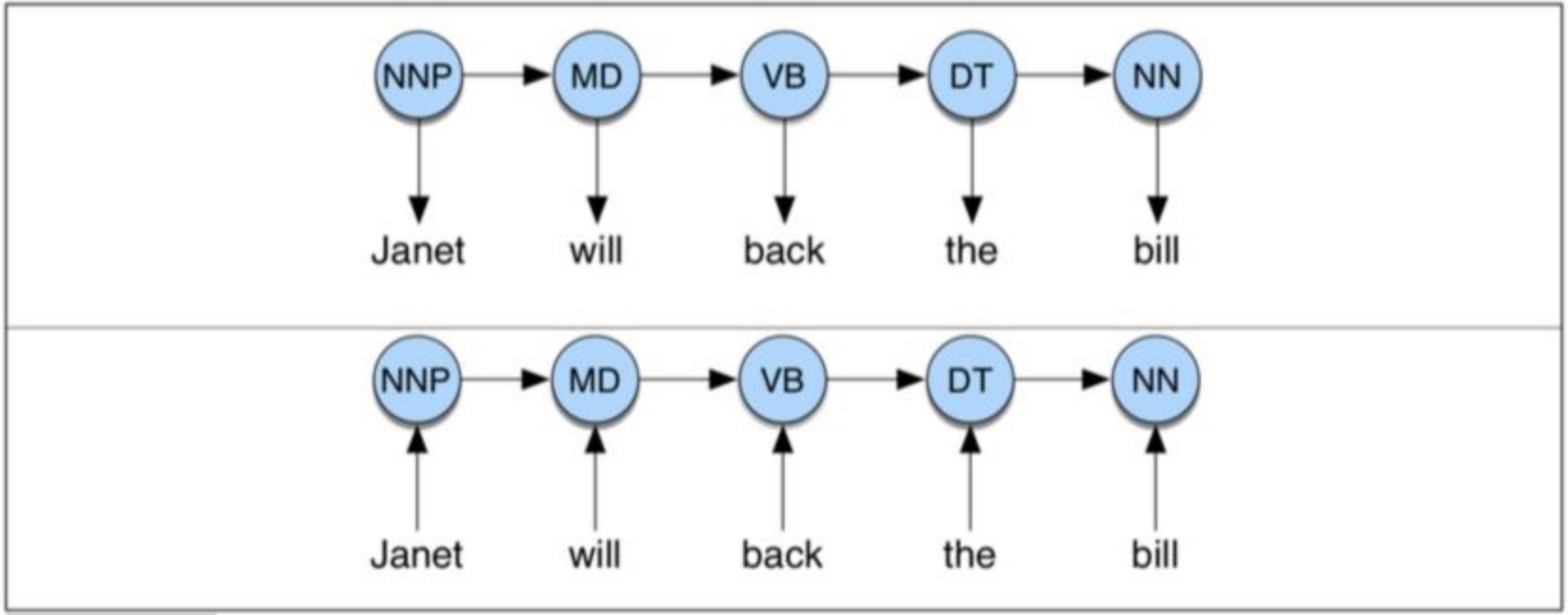
3. End

$$\max_{q \in Q^n} p(o, q) = \max_{1 \leq k \leq T} v_{nk} a_{kF}$$

Maximum Entropy Markov Models (MEMM)

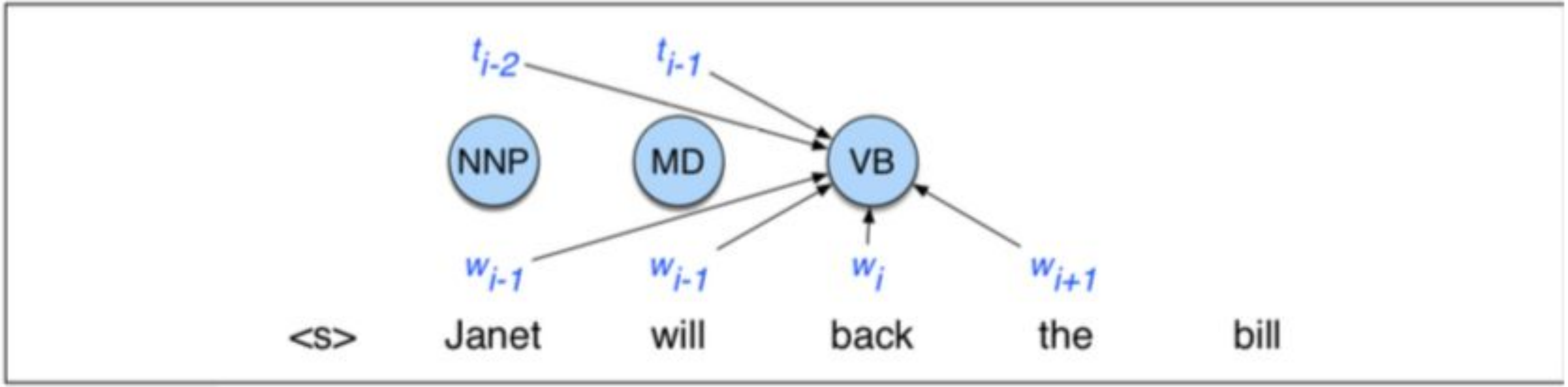
HMM: $\arg\max P(Y|X) = \arg\max_Y P(X|Y)P(Y)$

MEMM: $\arg\max_y P(Y|X) = \arg\max_Y P(y_i|y_{i-1}, x_i)$



Maximum Entropy Markov Models (MEMM)

- Features in a MEMM



- Feature templates: $< t_i, w_{i-2} >$, $< t_i, t_{i-1} >$, $< t_i, t_i, w_i, w_{i+1} >$
- Casing, shape, is number?, is string?, has a dash?, has a digit?, etc.

Maximum Entropy Markov Models (MEMM)

- Decoding MEMM

- Locally normalized logistic regression on a sequencer:

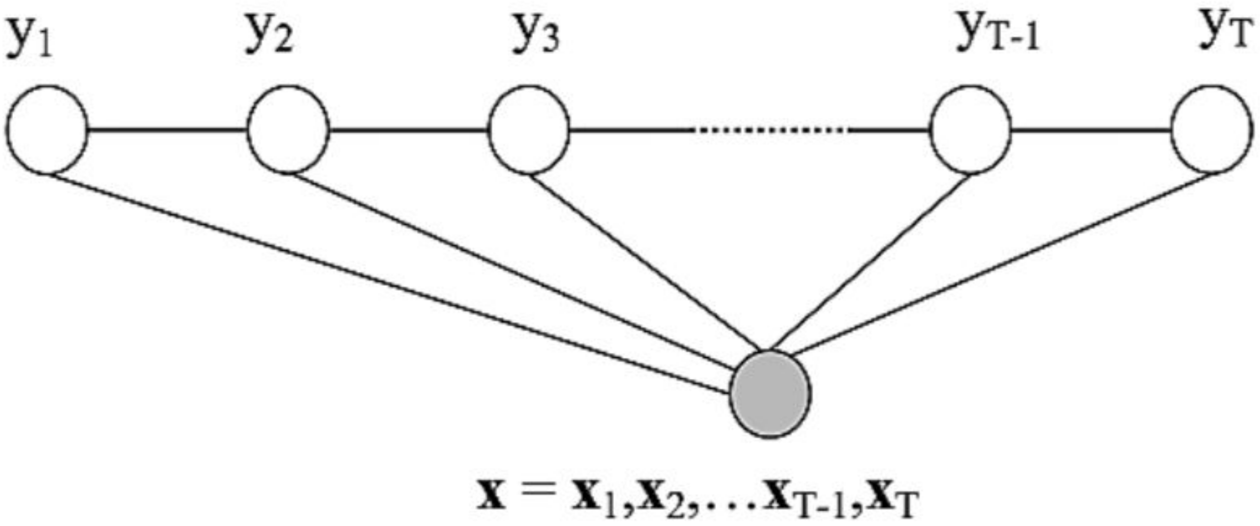
$$\hat{Y} = \arg\max P(Y|X) = \arg\max \prod_i P(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1}) =$$

$$= \arg\max_T \prod_i \frac{exp(\sum_j \theta_j f_j(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1}))}{\sum_{t' \in T} exp(\sum_j \theta_j f_j(t_i', w_{i-l}^{i+l}, t_{i-k}^{i-1}))}$$

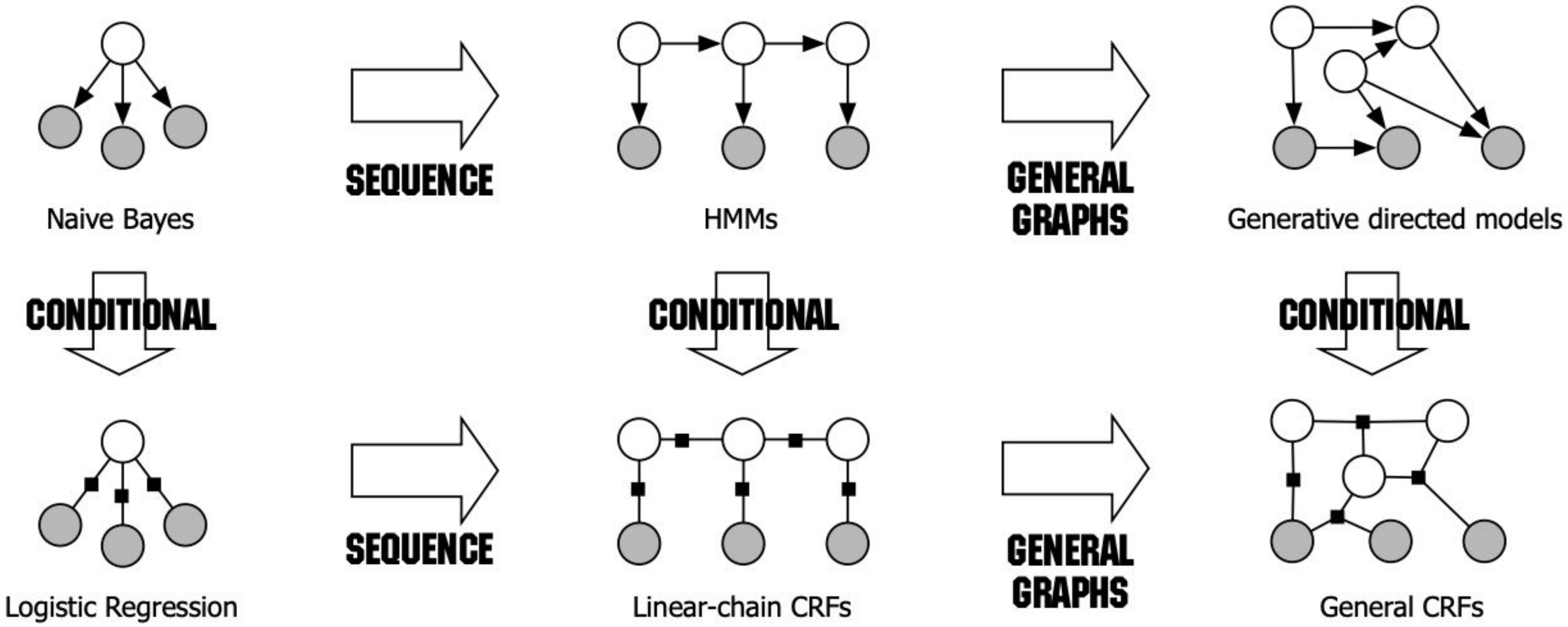
- Viterbi recursion step: $v_{ij} = \max_k v_{i-1,k} P(t_k | t_{k-1}, w_i)$
- Local normalization leads to labels bias: will/NN to/TO fight/VB

Conditional Random Field (CRF)

$$p(Y|X) = \frac{e^{\sum_{i=1}^{k} \lambda_i F_i(y,x)}}{\sum_{y' \in C^n} e^{\sum_{i=1}^{k} \lambda_i F_i(y',x)}}$$



HMM VS CRF


Naive Bayes
SEQUENCE
HMMs
GENERAL GRAPHS
Generative directed models
CONDITIONAL
CONDITIONAL
CONDITIONAL
Logistic Regression
SEQUENCE
Linear-chain CRFs
GENERAL GRAPHS
General CRFs

Questions

Reference

- https://lena-voita.github.io/nlp_course
- BiDir explained https://medium.com/@plusepsilon/the-bidirectional-language-model-1f3961d1fb27
- ELMO explained http://jalammar.github.io/illustrated-bert/
- Stanford NLP course http://cs224n.stanford.edu/
- AWD LSTM https://arxiv.org/pdf/1708.02182.pdf
- CRF tutorial https://people.cs.umass.edu/~mccallum/papers/crf-tutorial.pdf
- Simple explanation of crf https://habr.com/ru/post/241317/