SkillFactory

Language modelling
Sequence to Sequence task

Sidorov Nikita

MLE (NLP) Sber

What we will learn today

- Task of modelling text;
- LM as a formula;
- examples of application language modelling (LM);
- classical approach for LM;
- how to measure quality of language model;
- Neural LM;
- loss for neural LM;
- generation techniques;
- sequence to sequence tasks.

Modelling language

What we want from language model?

Modelling language

What we want from language model?

We want to somehow forecast future words by some previous context.

Modelling language

What we want from language model?

We want to somehow forecast future words by some previous context.

It means -> **Language Models (LM) estimate probability of token or several tokens in a row.**

LM in formula

We can say that our model needs to compute probability of a sentence

$$P_{(w_1, w_2, \ldots, w_n)} = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2)\ldots p(w_n|w_1, w_2, .., w_{n-1})$$
$$= \prod_{i=1}^{n} p(w_i|w_1, \ldots, w_{i-1})$$

LM in formula

Related task: compute probability of upcoming word.

$$p(w_n | w_1, w_2, .., w_{n-1})$$

LM in formula

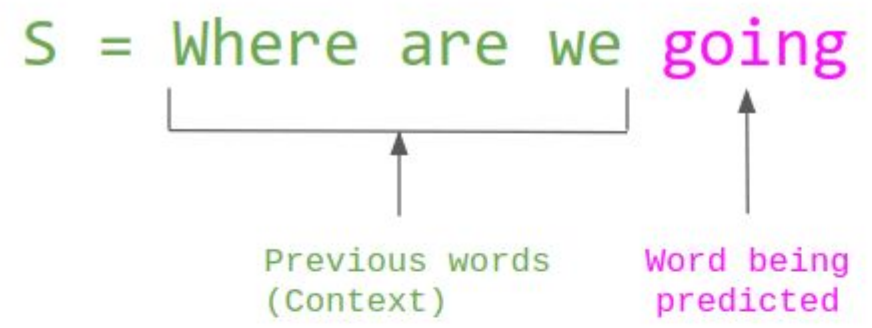
Model that compute

either this -> $$p(w_n|w_1, w_2, .., w_{n-1})$$

or this -> $$P_{(w_1, w_2, \ldots, w_n)} = \prod_{i=1}^{n} p(w_i|w_1, ..., w_{i-1})$$

is called Language Model.

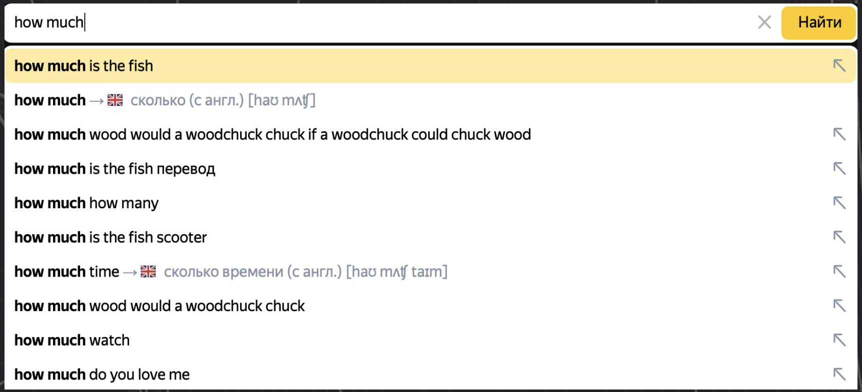
LM in formula



P(S) = P(Where) x P(are | Where) x P(we | Where are) x P(going | Where are we)

Applications of LM

- machine translation
- spelling correction
- web search engine
- keyboard advices
- authors identification
- etc



LM formula calculation

How to count probabilities?

LM formula calculation

How to count probabilities?

$$P(\text{going} \mid \text{Where are we}) = \frac{\mathit{Count}(\text{Where are we going})}{\mathit{Count}(\text{Where are we})}$$

LM formula calculation

How to count probabilities?

$$P(\text{going} \mid \text{Where are we}) = \frac{\mathit{Count}(\text{Where are we going})}{\mathit{Count}(\text{Where are we})}$$

What`s the problem to compute it?

LM formula calculation

How to count probabilities?

$$P(\text{going} \mid \text{Where are we}) = \frac{\mathit{Count}(\text{Where are we going})}{\mathit{Count}(\text{Where are we})}$$

What`s the problem to compute it?

1. **It is too many possible sentences!**
2. **We`ll never see enough data for estimating this.**

Markov assumption

Simplifying assumption:

P(going | Where are we) ≈ P(going| we)

OR

P(going | Where are we) ≈ P(going| are we)

Markov assumption

More formally: $$P(y_t \mid y_1, y_2, \dots, y_{t-1}) = P(y_t \mid y_{t-n+1}, \dots, y_{t-1})$$

The probability of word depends only on **fixed** number of previous words.

N-gram LM

How to build it?

1. Make a simplification assumption x^{t+1} depends only on preceding *n-1* words.

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\dots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\overbrace{\boldsymbol{x}^{(t)},\dots,\boldsymbol{x}^{(t-n+2)}}^{n\text{-1 words}})$$

N-gram LM

How to build it?

1. Make a simplification assumption x^{t+1} depends only on preceding *n-1* words.

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$

N-gram LM

How to build it?

1. Make a simplification assumption x^{t+1} depends only on preceding *n-1* words.

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$

2. Move from probabilities to counts of corresponding n-grams.

N-gram LM example

Assume we have 4-gram language model.

when the lector come into the class, the students opened their ______

N-gram LM example

Assume we have 4-gram language model.

~~when the lector come into the class, the~~ students opened their ______

discard condition on this

N-gram LM example

Assume we have 4-gram language model.

~~when the lector come into the class, the~~ students opened their ______

discard   condition on this

$$P(\boldsymbol{w} \mid \text{students opened their}) = \frac{\mathit{Count}(\text{students opened their } \boldsymbol{w})}{\mathit{Count}(\text{students opened their})}$$

N-gram LM example

Assume we have 4-gram language model.

~~*when the lector come into the class, the*~~ *students opened their* ______

discard — condition on this

$$P(\boldsymbol{w} \mid \text{students opened their}) = \frac{\mathit{Count}(\text{students opened their } \boldsymbol{w})}{\mathit{Count}(\text{students opened their})}$$

Tip:
In practice usually use 5-gram model

N-gram LM example

Assume we have 4-gram language model.

~~*when the lector come into the class, the*~~ *students opened their* ______

discard condition on this

$$P(\boldsymbol{w} \mid \text{students opened their}) = \frac{\textit{Count}(\text{students opened their } \boldsymbol{w})}{\textit{Count}(\text{students opened their})}$$

For example, in our corpus we have:

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times -> P(books| context) = 0.4
- "students opened their notebooks" occurred 200 times -> P(notebooks| context) = 0.2

Problems with N-gram LM

- can not memorize long context
- can be 0 occurrence of specific n-gram
- can be 0 occurrence of specific n-1 gram
- storing problem (long n-grams significantly increase size of LM)

How to evaluate quality of LM

Idea: If our model construct good sentences it assigns higher probabilities to "real" or "frequently observed" words.

How to evaluate quality of LM

Idea: If our model construct good sentences it assigns higher probabilities to “real” or “frequently observed” words.

As usual we will evaluate quality on some unseen data that can differ from training data.

How to evaluate quality of LM

Idea: If our model construct good sentences it assigns higher probabilities to “real” or “frequently observed” words.

As usual we will evaluate quality on some unseen data that can differ from training data.

Evaluation metric tells us how well our model does on test dataset.

How to evaluate quality of LM

First approach

1. Evaluate each of your LM on your task (MT system, spelling correction)

How to evaluate quality of LM

First approach

1. Evaluate each of your LM on your task (MT system, spelling correction)
2. Get accuracy (or other metric) for each LM

How to evaluate quality of LM

First approach

1. Evaluate each of your LM on your task (MT system, spelling correction)
2. Get accuracy (or other metric) for each LM
3. Compare metrics for each LM.

How to evaluate quality of LM

First approach

1. Evaluate each of your LM on your task (MT system, spelling correction)
2. Get accuracy (or other metric) for each LM
3. Compare metrics for each LM.

This approach named **extrinsic evaluation**.
Sometimes evaluation can take days or even weeks.

How to evaluate quality of LM

<u>Second approach</u>

It is cold <u>**intrinsic**</u>. We will explore metrics called **perplexity**.

Idea: better model assigns higher probability to the word that actually occurs.

Perplexity

Intuition: When reading a new text, how much is model “surprised”?

Perplexity

Intuition: When reading a new text, how much is model “surprised”?

Perplexity is the probability of the test set normalized by the number of words.

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Perplexity

Intuition: When reading a new text, how much is model "surprised"?

Perplexity is the probability of the test set normalized by the number of words.

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Minimizing perplexity is the same as maximizing the probability.

Perplexity intuition

Sometimes perplexity is called - average branching factor. So in any point of sentence on average how many different things can happen.

Perplexity intuition

Sometimes perplexity is called - average branching factor. So in any point of sentence on average how many different things can happen.

Let`s suppose a sentence consisting of random digits. For example length of sentence is 5.

What is the perplexity of this sentence according to a model, that assign P = 1/10 to each digit?

Perplexity intuition

Sometimes perplexity is called - average branching factor. So in any point of sentence on average how many different things can happen.

Let`s suppose a sentence consisting of random digits. For example length of sentence is 5.

What is the perplexity of this sentence according to a model, that assign P = 1/10 to each digit?

P(W) = P($w_1 w_2 w_3 w_4 w_5$) = $P(w) = \left(\left(\frac{1}{10}\right)^5\right)^{-1/5}$ = 10

Perplexity intuition

Best perplexity score is 1. If the model is perfect and assigns probability 1 to correct tokens.

The worst perplexity is |V|. If the model knows nothing about the data, it assigns probability 1/|V| to all tokens, regardless of context.

Recap: what we need form LM

$$P(y_1, y_2, \ldots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \cdots \cdot P(y_n|y_1, \ldots, y_{n-1}) = \prod_{t=1}^{n} P(y_t|y_{<t})$$

We need to: define how to compute the conditional probabilities $P(y_n|y_1 \ldots y_{n-1})$

Recap: what we need form LM

$$P(y_1, y_2, \dots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \dots \cdot P(y_n|y_1, \dots, y_{n-1}) = \prod_{t=1}^{n} P(y_t|y_{<t})$$

<u>We need to</u>: define how to compute the conditional probabilities $P(y_n|y_1 \dots y_{n-1})$

In neural networks, we do as usually:

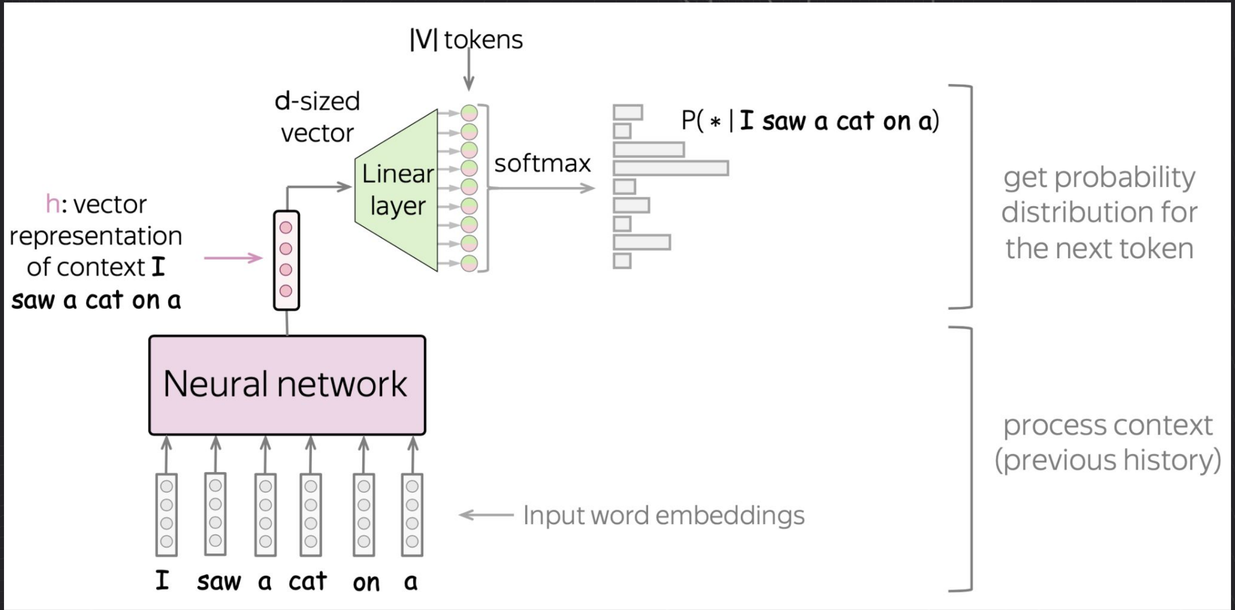
Train a NN to predict them.

Neural LM

General view:

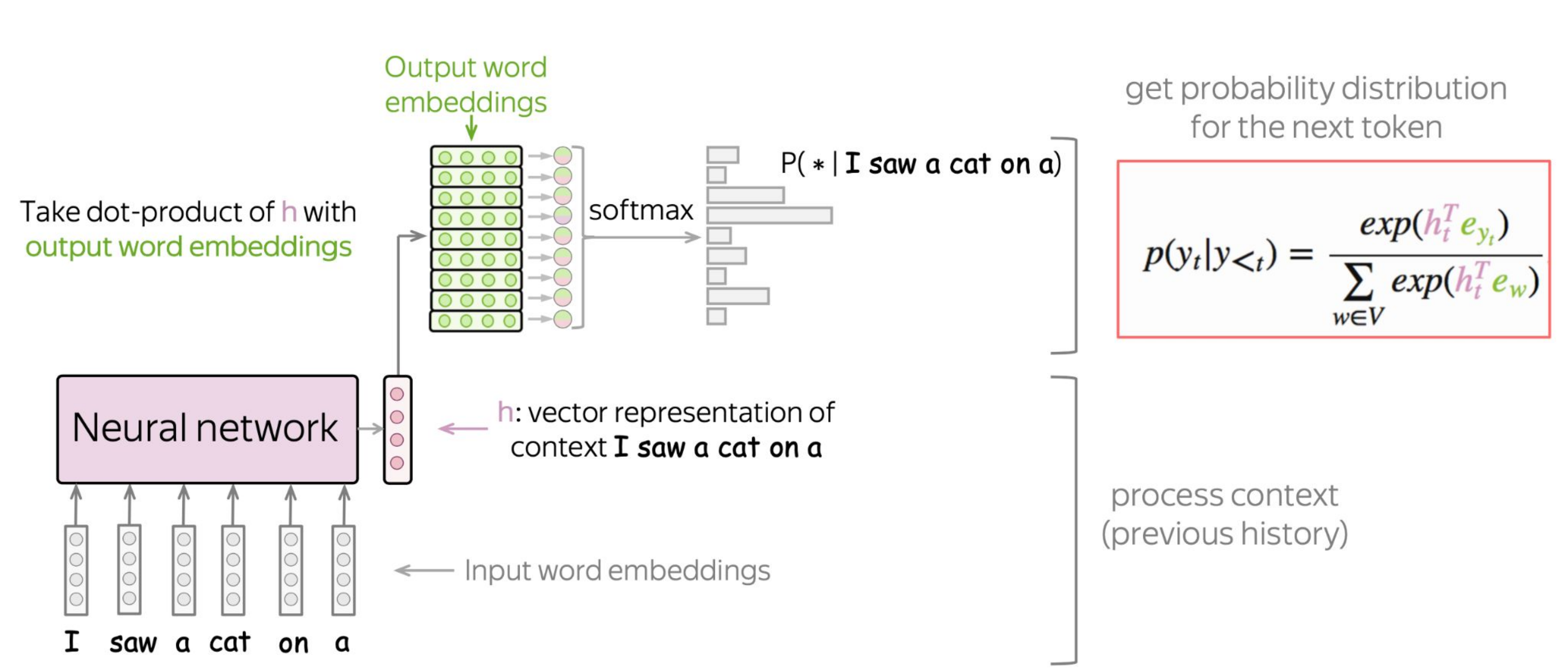
- process context (get vector representation of the previous context)
- evaluate probabilities (predict probability distribution for next token)

Neural LM



example from [Lena Voita NLP course]

Neural LM



example from Lena Voita NLP course

Training neural LM

What kind of task we are solving in terms of Machine Learning?

Training neural LM

What kind of task we are solving in terms of Machine Learning?

Multiclass classification!

Training neural LM

What kind of task we are solving in terms of Machine Learning?

Multiclass classification!

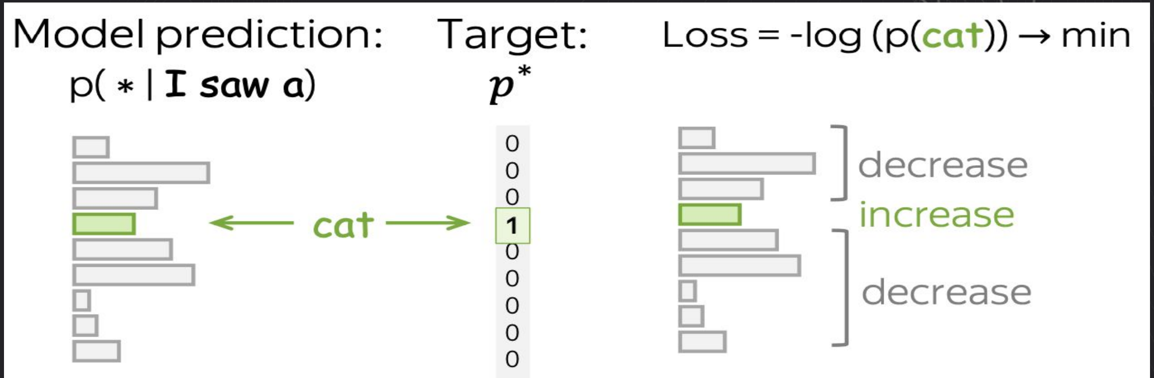
That's why we need cross-entropy loss function.

$$CCE(p,t) = -\sum_{c=1}^{C} t_{o,c} \log\left(p_{o,c}\right)$$

Training neural LM

Target word

Training example: I saw a cat on a mat <EOS>

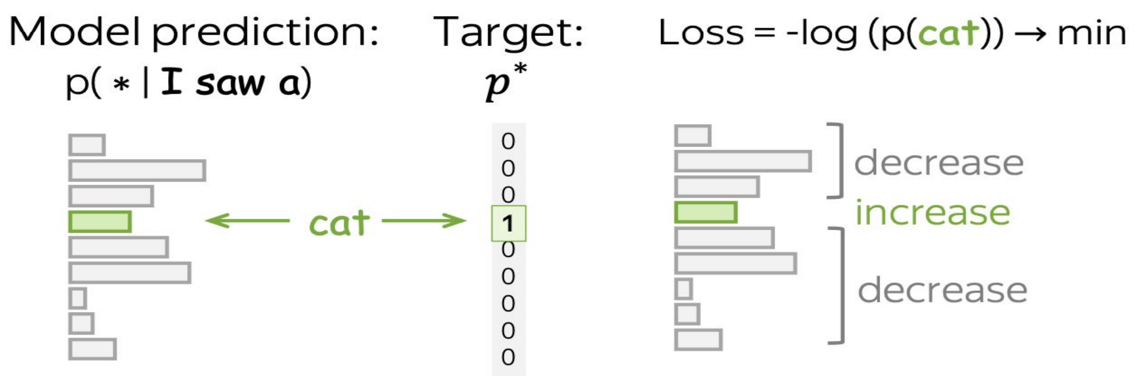


example from [Lena Voita NLP course]()

Training neural LM

Target word

Training example: I saw a cat on a mat <EOS>



Cross-entropy loss:

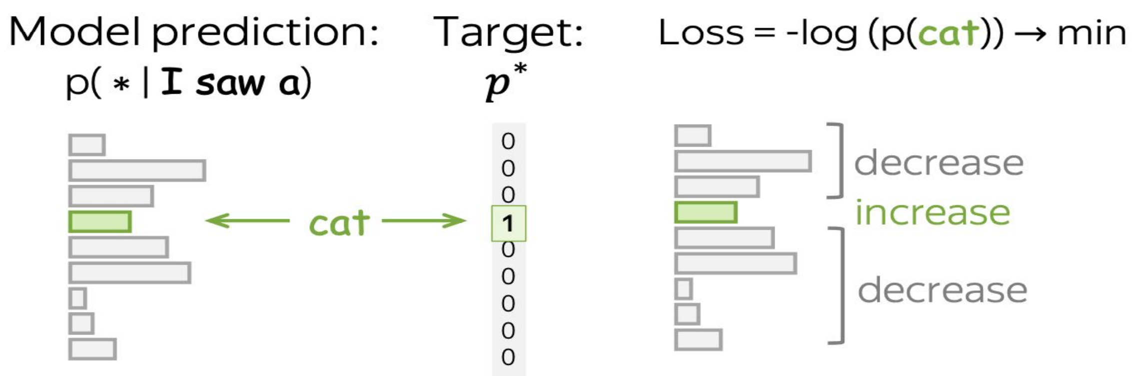
$$-\sum_{i=1}^{|V|} p_i^* \cdot \log P(y_t = i|x) \to min \; (p_k^* = 1, p_i^* = 0, i \neq k)$$

example from [Lena Voita NLP course]

Training neural LM

Target word

Training example: I saw a cat on a mat <EOS>



Cross-entropy loss:

$$-\sum_{i=1}^{|V|} p_i^* \cdot \log P(y_t = i|x) \rightarrow min \ (p_k^* = 1, p_i^* = 0, i \neq k\ )$$

For OHE targets,
this is equivalent to

$$-\log P(y_t = cat|x) \rightarrow min$$

example from Lena Voita NLP course

Again to perplexity

Perplexity is the probability of the test set normalized by the number of words.

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Again to perplexity

Perplexity is the probability of the test set normalized by the number of words.

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

$$= \prod_{t=1}^{T} \left(\frac{1}{\hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}} \right) = \exp(J(\theta))$$

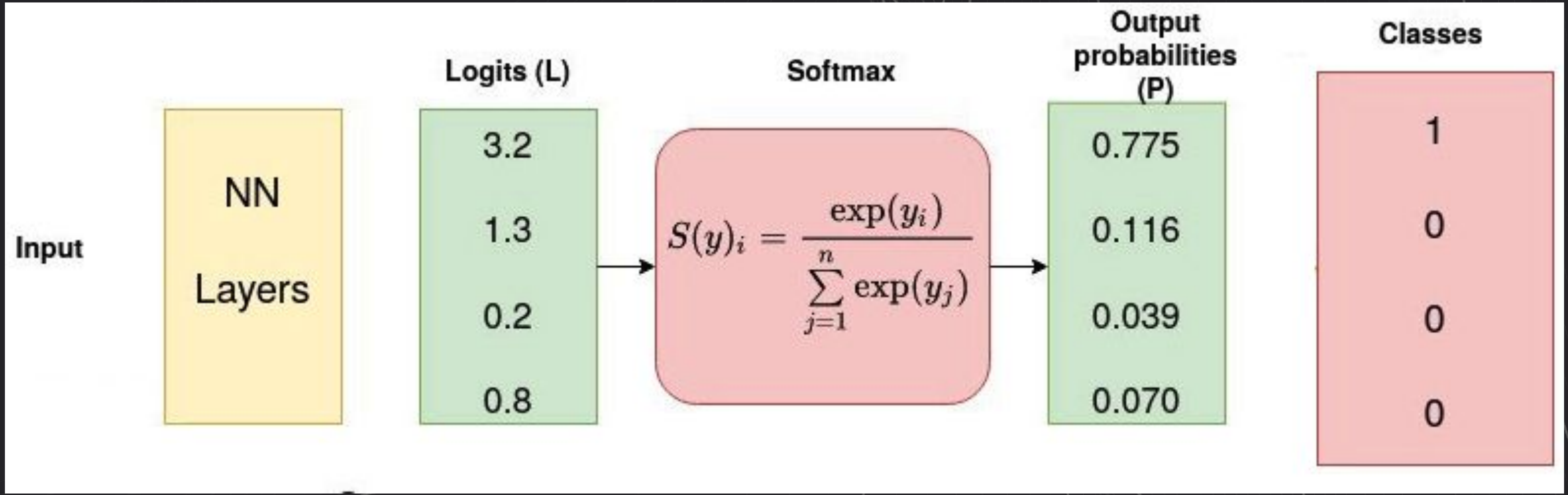
Text generation strategies

We want our generated texts to be:

- texts has to make sense (coherence)
- their must differ from each other (diversity)

Text generation strategies

Before that for determinism we used Softmax.



Text generation strategies

Now we will use special for of softmax - softmax with temperature.

$$\frac{\exp(h^T w)}{\sum_{w_i \in V} \exp(h^T w_i)} \rightarrow \frac{\exp\left(\frac{h^T w}{\tau}\right)}{\sum_{w_i \in V} \exp\left(\frac{h^T w_i}{\tau}\right)}$$

τ - softmax temperature

Intuition: divide by some temperature to chanhe total entropy of system

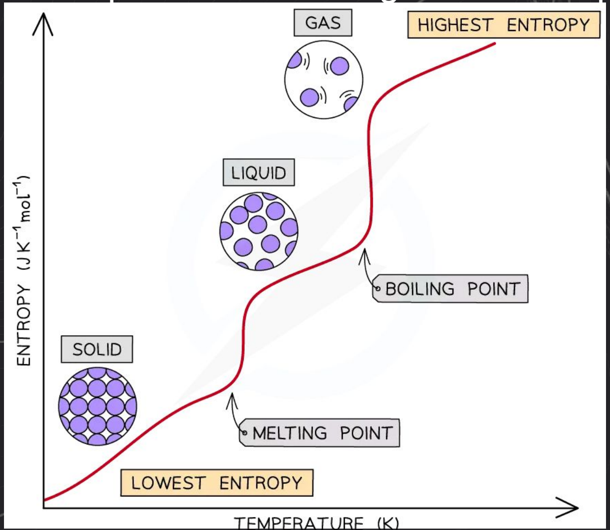
Text generation strategies

Now we will use special for of softmax - softmax with temperature.

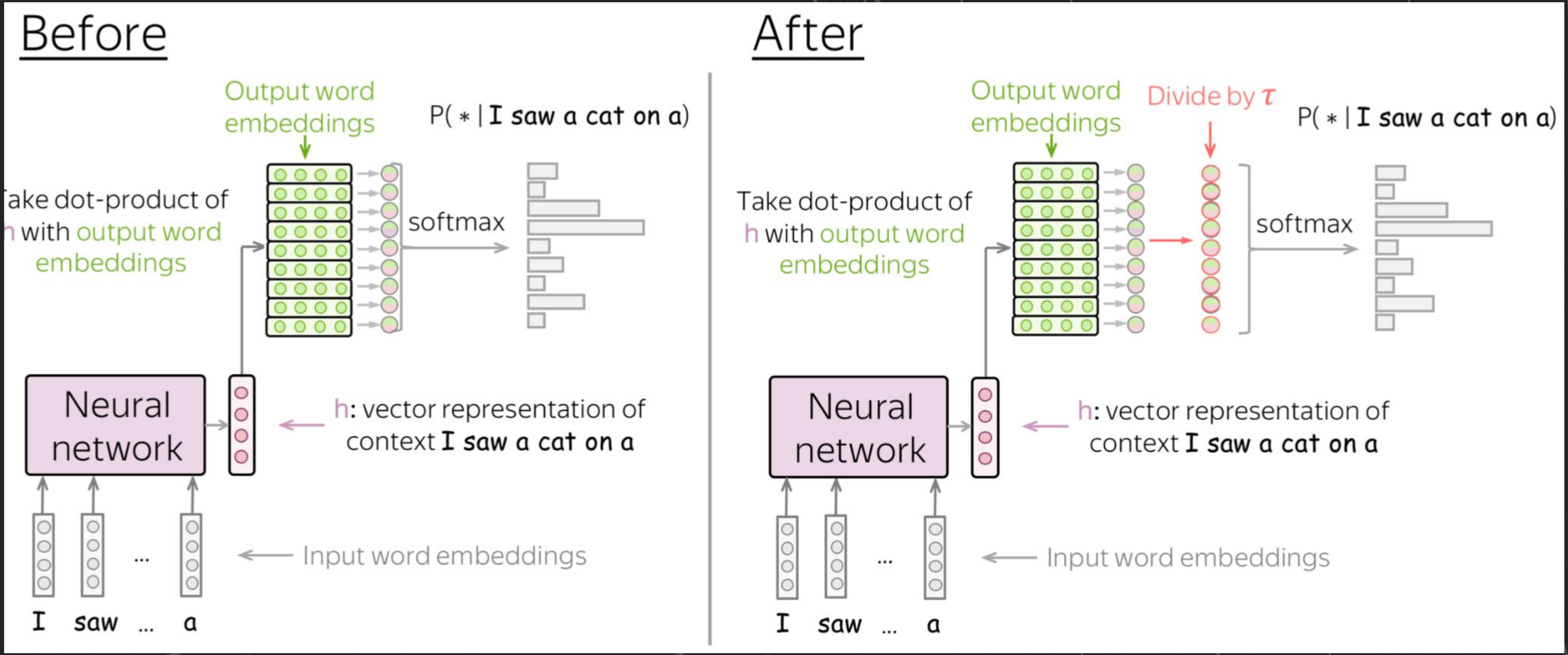
$$\frac{\exp(h^T w)}{\sum_{w_i \in V} \exp(h^T w_i)} \rightarrow \frac{\exp\left(\frac{h^T w}{\tau}\right)}{\sum_{w_i \in V} \exp\left(\frac{h^T w_i}{\tau}\right)}$$

τ - softmax temperature

Intuition: divide by some temperature to change total entropy of system

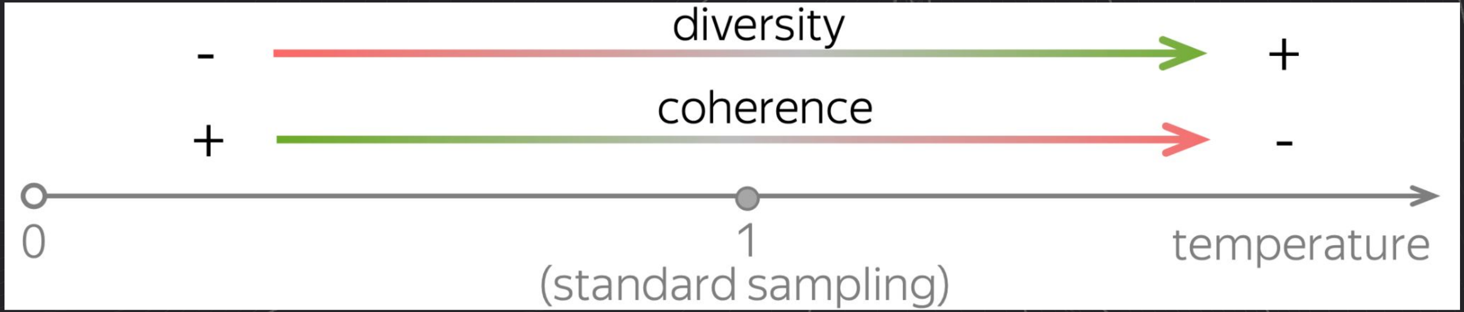


Text generation strategies



Text generation strategies

Both increasing and decreasing improve one of coherence and diversity, but hurt other

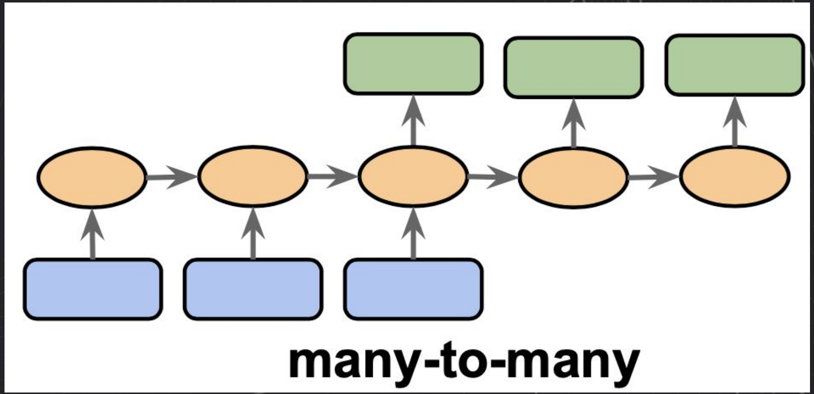


Other types of sampling

- Top-k sampling
- Top-p (Nucleus) sampling

Sequence to Sequence task

Traditionally seq2seq task is associated with translation from one language to another. It doesn't have to be human languages



Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

Statistical machine translation

We want to find best English sentence *y,* for given Russian sentence *x.*

$$\text{argmax}_y \; P(y|x)$$

It’s all like human deal with this task.

Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

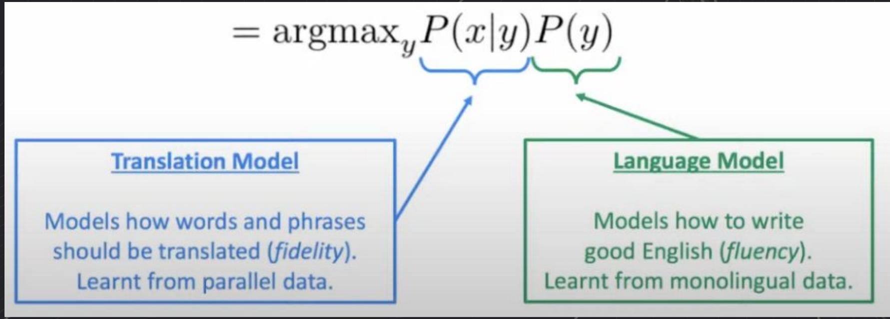
Statistical machine translation

We want to find best English sentence *y*, for given Russian sentence *x*.

$$\mathrm{argmax}_y\ P(y|x)$$

It's all like human deal with this task.

If apply Bayes Rule we will break this down into two components:



Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

Problems of Statistical MT

- systems had many separately-designed components
- lots of feature engineering
- need to design features to capture particular language phenomena
- lots of human effort to maintain
- reported effort for every language pair

Comparison of different MT approaches

Human deal with this task like this

$$\text{argmax}_y\ P(y|x)$$

Computer deal with this task like this

$$\text{argmax}_y\ P(y|x, \theta)$$

where p - some neural model,

and θ - parameters of model

Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

Neural approach for MT

Computer deal with this task like this

$$\operatorname{argmax}_y P(y|x, \theta)$$

where p - some neural model, and θ - parameters of model

Questions we need to ask?

modeling
How does the model for
$p(y|\ x, \theta)$ look like?

learning
How to find best θ?

search
How to find the argmax?

Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

Neural approach for MT

Computer deal with this task like this

$$\text{argmax}_y\ P(y|x, \theta)$$

where p - some neural model, and θ - parameters of model

Questions we need to ask?

modeling
How does the model for $p(y|\ x, \theta)$ look like?

learning
How to find best θ?

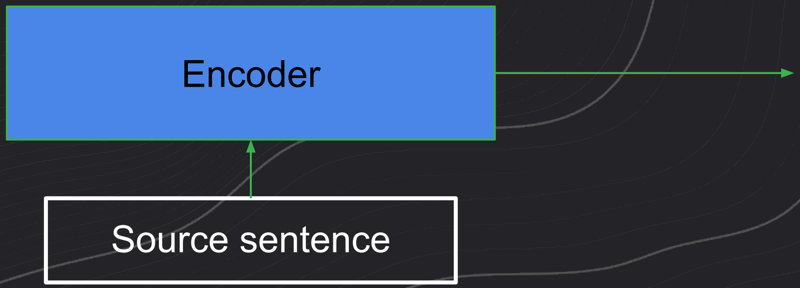
search
How to find the argmax?

Сегодня мы станем гуру НЛП. → Today we will become NLP gurus.

Encoder-Decoder Framework

The standard modeling paradigm:

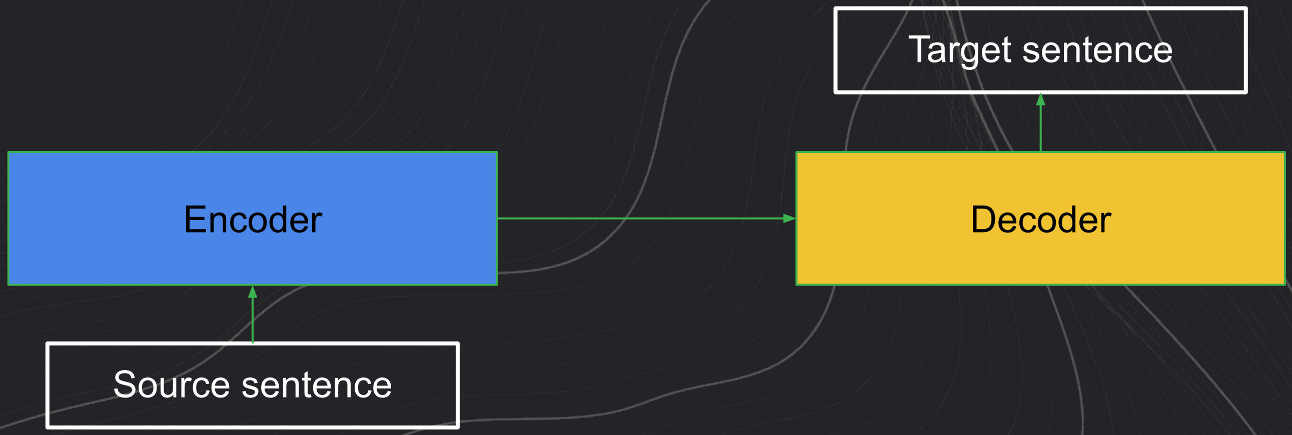
- Encoder - reads the source sentence and builds its representation



Encoder-Decoder Framework

The standard modeling paradigm:

- Encoder - reads the source sentence and builds its representation
- Decoder - uses source representation from the encoder to generate the target sequence



Conditional Language Modeling

Language models:

$$P(y_1, y_2, \dots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \dots \cdot P(y_n|y_1, \dots, y_{n-1}) = \prod_{t=1}^{n} P(y_t|y_{<t})$$

Conditional Language Modeling

Language models:

$$P(y_1, y_2, \ldots, y_n) = \prod_{t=1}^{n} p(y_t | y_{<t})$$

Conditional

Language models:

$$P(y_1, y_2, \ldots, y_n, | x) = \prod_{t=1}^{n} p(y_t | y_{<t}, x)$$

condition on source x

Terminology

- Language model
- conditional probability
- Markov assumption
- n-gram
- extrinsic/intrinsic metric
- perplexity
- cross-entropy
- softmax
- temperature softmax
- top-k/top-p sampling
- seq2seq
- encoder
- decoder
- conditional modeling