

# Обработка естественного языка. Задача языкового моделирования.

МФТИ, Сбертех

# Что такое Natural Language Processing?

- Natural Language Processing(NLP) – набор техник и методов по обработке естественного языка
- Отвлечение от компьютерной лингвистики – смесь компьютерных наук и лингвистики. Занимается изучением грамматических и лексических свойств языка.
- NLP в DL – многослойные нейронные сети, несерьёзные названия статей и зоопарк с Sesame Street. *И немного лингвистики.*



## We used Neural Networks to Detect Clickbait: You won't believe what happened Next!

Ankesh Anand<sup>1</sup>, Tanmoy Chakraborty<sup>2</sup>, and Noseong Park<sup>3</sup>

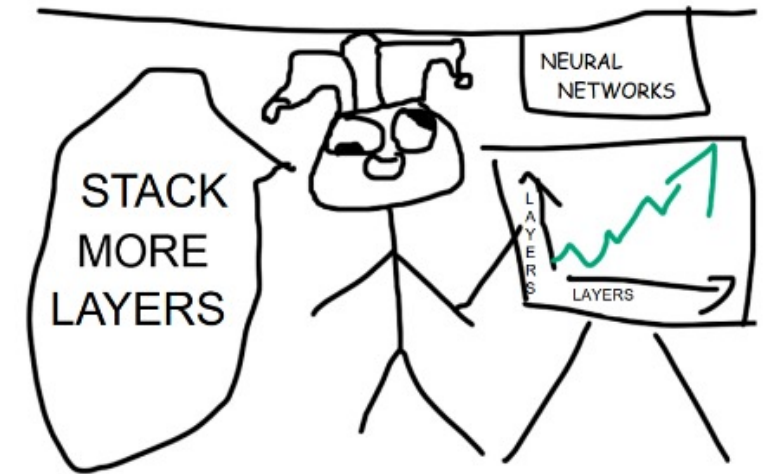
<sup>1</sup> Indian Institute of Technology, Kharagpur, India  
anandank@iitk.quebec,

<sup>2</sup> University of Maryland, College Park, USA  
tanchak@umiacs.umd.edu

<sup>3</sup> University of North Carolina, Charlotte, USA  
npark2@uncc.edu

**Abstract.** Online content publishers often use catchy headlines for their articles in order to attract users to their websites. These headlines, popularly known as *clickbait*, exploit a user's curiosity gap and lure them to click on links that often disappoint them. Existing methods for automatically detecting clickbaits rely on heavy feature engineering and domain knowledge. Here, we introduce a neural network architecture based on *Recurrent Neural Networks* for detecting clickbaits. Our model relies on distributed word representations learned from a large unannotated corpora, and character embeddings learned via Convolutional Neural Networks. Experimental results on a dataset of news headlines show that our model outperforms existing techniques for clickbait detection with an accuracy of 0.98 with F1-score of 0.98 and ROC-AUC of 0.99.

**Keywords:** Clickbait Detection, Deep Learning, Neural Networks

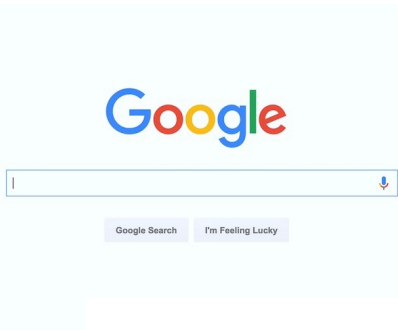


- Языковое моделирование и контекстно-независимые эмбединги
- Рекуррентные нейронные сети
- Трансформеры
- Инструменты NLP(морфология, синтаксис, семантика)
- Информационный поиск – поиск и ранжирование
- Дистилляция больших нейронных сетей
- Задача Text2Machine

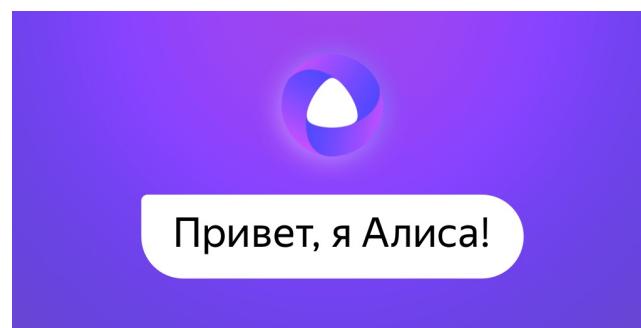
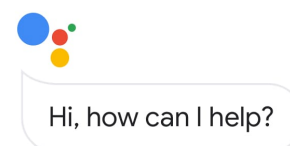
## Задачи анализа текста

- Классификация текста – text classification
  - > Вызови мне такси
  - < call\_taxi
- Выделение именных сущностей – NER
  - > США выдвинуло обвинения против Facebook
  - < B-CNT O O O B-ORG
- Выделение ключевых слов
  - Напомни мне вечером позвонить в химчистку
- Выделение отношений между сущностями
  - > Кто снял фильм Титаник –
  - < ?x – dr:director – “Титаник»
- Морфологический анализ
- Синтаксический анализ
- Семантический анализ
  - Вася<sub>агентс</sub> ударил по мячу<sub>пациенс</sub>
- Тематическое моделирование
- Информационный поиск
  - Поиск в коллекции документов
  - Вопросно-ответные системы
- Задачи по генерации текста
  - Диалоговые модели
  - Машинный перевод
  - Автоматическое дополнение текста

## Поисковые системы



## Голосовые ассистенты



## Системы анализа текстов



# Задача языкового моделирования

**Статистическая языковая модель** это распределение вероятностей над последовательностью слов. Для последовательности длины  $n$  – языковая модель позволяет вычислить вероятность  $P(w_1, w_2, \dots, w_n)$  встретить такую последовательность в исследуемом языке.

$$P(w_1, w_2, \dots, w_n) = P(w_1) * P(w_2|w_1) * P(w_3|w_1, w_2) \dots P(w_n|y_1, \dots, y_{n-1}) = \prod_{t=1}^n P(y_t|y_{<t})$$

$$P(\text{Мама мыла раму}) = P(\text{раму}|\text{Мама мыла}) * P(\text{мыла}|\text{Мама}) * P(\text{Мама})$$

$$P(x_i | \text{Мама мыла}) = \frac{\text{Count}(\text{Мама мыла } x_i)}{\text{Count}(\text{Мама мыла})}$$


Часто ли это словосочетание встречается в языке?


- Очень разреженная матрица вероятностей словосочетаний
- Проклятие размерности – Какая модель нам нужна, чтобы сгенерировать последовательность в 10 символов с языковым словарем в 100000?

# Векторное представление слов

В дискретном пространстве статистическое моделирование работает очень плохо – любое изменение дискретной переменной ведет к значительному изменению целевой функции. В случае, если каждая дискретная переменная имеет большую область значений – каждый объект будет очень сильно отличаться друг от друга (например, расстояние Хэмминга).

Использование действительных значений может помочь в задаче моделирования. Для этого мы можем использовать гладкие функции, например нейронные сети. С помощью нейронных сетей, мы даже можем выучить некоторое представление для наших объектов, которое мы можем сравнивать друг с другом.

Мама мыла раму   $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Мама мыла раму   $\begin{bmatrix} 0.34 & 0.82 & -0.54 \\ -0.29 & 0.26 & 1.05 \\ 0.52 & 0.14 & 0.43 \end{bmatrix}$

# Статистические методы

Для представления вектора слова, можно использовать статистические способы представления на основании корпуса. Пусть есть корпус  $C$ , который состоит из множества документов  $D_1, \dots, D_n$ . Построим по этим данным вектор слова  $w$ .

- **Мешок слов (Bag of Words)** – считаем количества раз, сколько слово  $w$  встречается внутри каждого документа. Получаем вектор  $E_w \in R^{1 \times |C|}$ .
  - 👍 Получаем менее разреженный вектор, чем one-hot кодирование.
  - 👎 Если корпус русский, слово **и** встретится чаще, чем **осциллятор**. Значит ли это, что оно более важное?
- **TF-IDF** – количество употреблений слова взвешивается на основании частоты употреблений в рамках документа и корпуса.
  - **TF** (Term Frequency (частота слов)) – важность слова  $w$  в рамках отдельного  $D_i$ . Считается, как количества встречаемости слова  $w$  к общему количеству слов документа.

$$TF(w, D_i) = \frac{n_w}{|D_i|}$$

- **IDF** (Inverse document frequency/обратная частота документа) – инверсия частоты, с которой слово  $w$  встречается в документа корпуса  $C$ . Считается, как логарифм от отношения общего кол-ва документов корпуса к числу документов, где встречается  $w$ .

$$IDF(w, C) = \log\left(\frac{|C|}{|\{w \in D_i | D_i \in C\}|}\right)$$

$$Tf - Idf(w, D, C) = TF(w, D) * IDF(w, C)$$



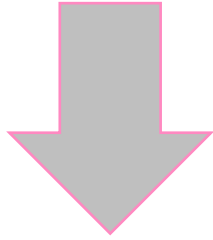
- Идея дистрибутивной семантики - **Смысл слова определяется его контекстом.**
- Слово можно представить с помощью контекста слов в котором слово встречается.
- Пример:
  - Влад заточил **косу** на завтра
  - У Ани была длинная, русая **коса**
  - В Калининграде есть песчаная **коса**
- Слово – **коса** можно представить с помощью множества контекстов, в котором он употребляется.



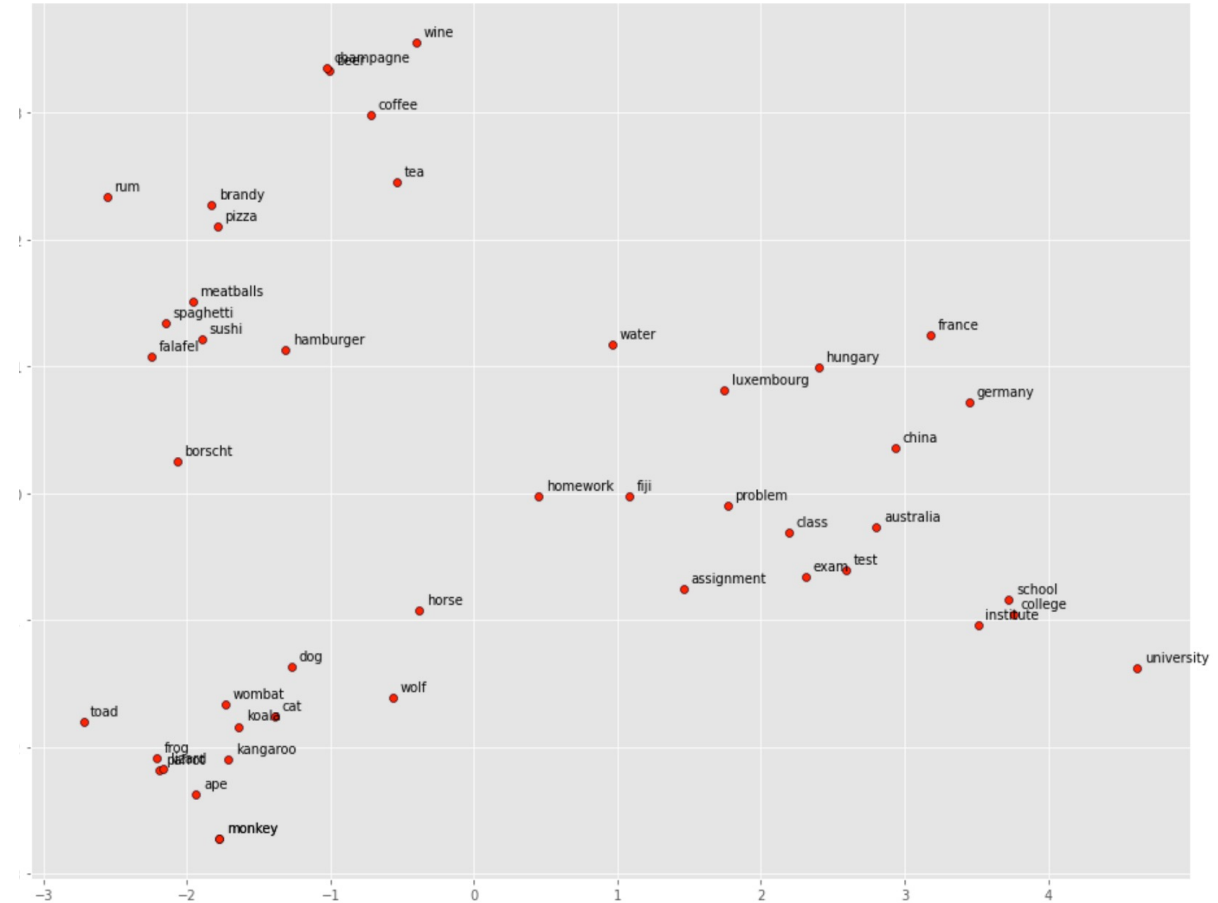
Джон Руперт Фёрс  
*You shall know the word by  
the company it keeps.*

# Векторное представление слов

Нужно обучить модель **X** так, что векторное представление слова **A** из одного контекста **C** будет близко к векторному представлению слова **B** из этого же контекста **C**. Слова использованные в разных контекстах – будут иметь сильно отличающиеся.

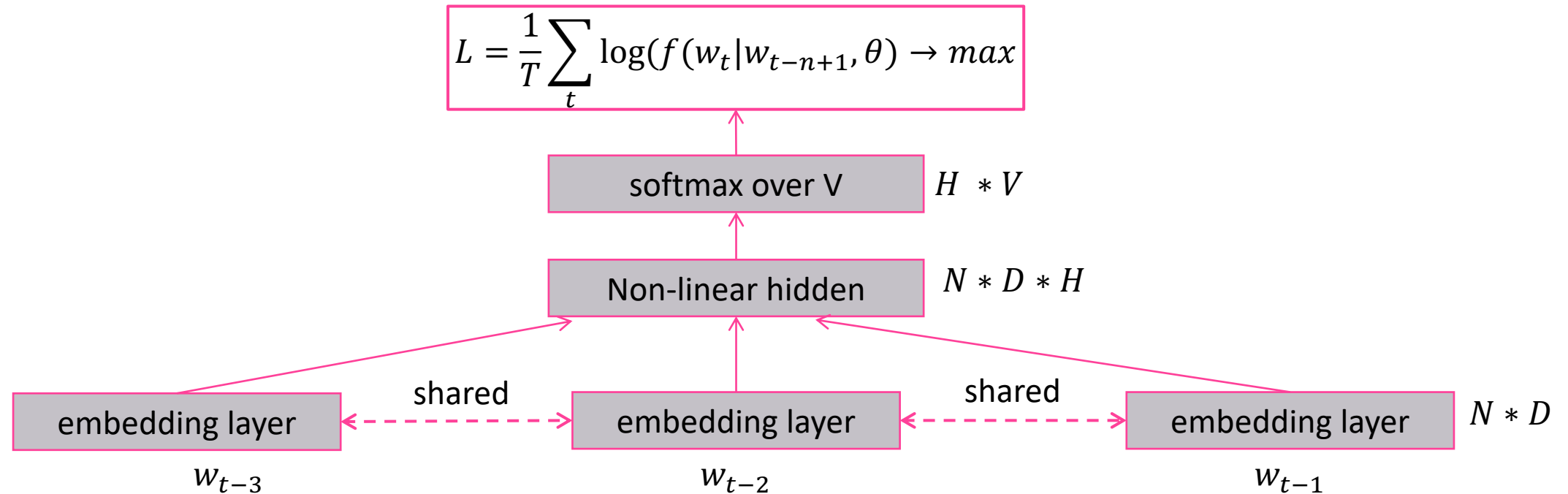


- + Можно сравнивать слова друг с другом
- + Можно решать задачи языкового моделирования



# Вероятностные нейронные языковые модели

1. Для каждого слова создаем свой вектор в матрице эмбедингов  $C(V * d)$ ,  $d$  – размерность эмбединга.
2. Используем модель с одним нелинейным слоем( $\tanh$ ) задачи языкового моделирования.
3. Обучаем векторные представления слов в матрице  $C$  и параметры обучаемой функции.



$$Q = N * D + N * D * H + H * V$$

[A Neural Probabilistic Language Model\(2003\)](#)

[Efficient Estimation of Word Representations in Vector Space](#) (Mikolov, 2013) – статья в которой была представлена технология Word2Vec.

- Есть большой корпус текста
- Каждому слову корпуса соответствует определенный вектор
- Алгоритм:
  - Идем через каждое слово  $t$  в корпусе окном фиксированной длины  $N$ .  
Центральное слово –  $c$ , слова из контекста(окна) –  $o$
  - Считаем  $p(c|o)$  или  $p(o|c)$
  - Задача – научить модель определять наиболее вероятное слово с помощью центрального слова или контекстного слова.
- Результат – мы получаем эмбединги для слов корпуса, таким образом, что эмбединги слов из одного контекста становятся близки к друг другу.

# Общая идея word2vec

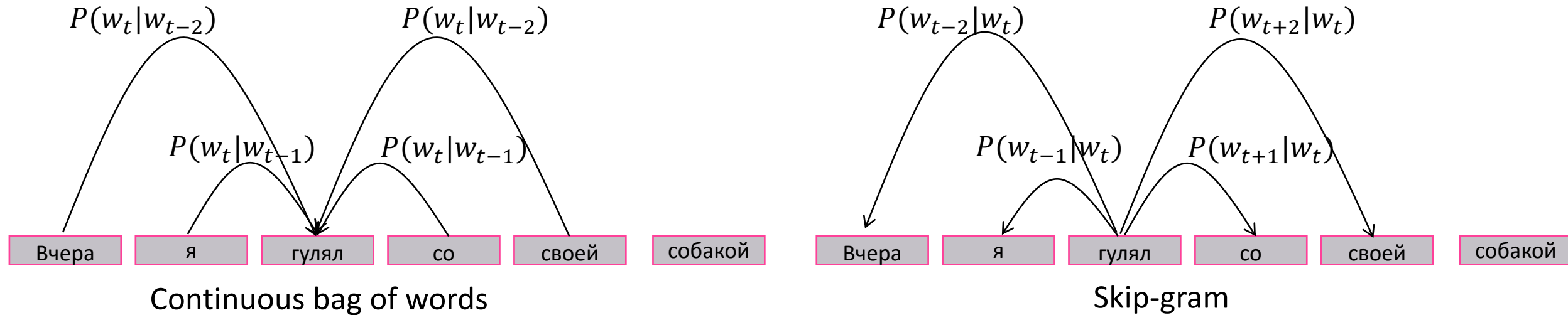
[Efficient Estimation of Word Representations in Vector Space](#) – статья в которой была представлена технология Word2Vec.

Для каждой позиции  $t = 1, \dots, T$  в корпусе предсказываем слова из контекста размера  $m$ , при данном центральном слове  $w_t$ . Максимизация логарифма правдоподобия:

$$L(\theta) = \prod_{t=1}^T \prod_{-m \leq j \leq m} P(w_{t+j} | w_t; \theta)$$

Переходя к задаче минимизации:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m} \log P(w_{t+j} | w_t; \theta)$$



$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m} \log P(w_{t+j} | w_t; \theta) \rightarrow \min$$

Как можно посчитать  $P(w_{t+j} | w_t; \theta)$ ?

- Для каждого слова будем использовать – 2 вектора
  - Вектор  $v_c$  - если слово  $c$  – центральное слово
  - Вектор  $u_o$  - если слово  $o$  – слово из контекста

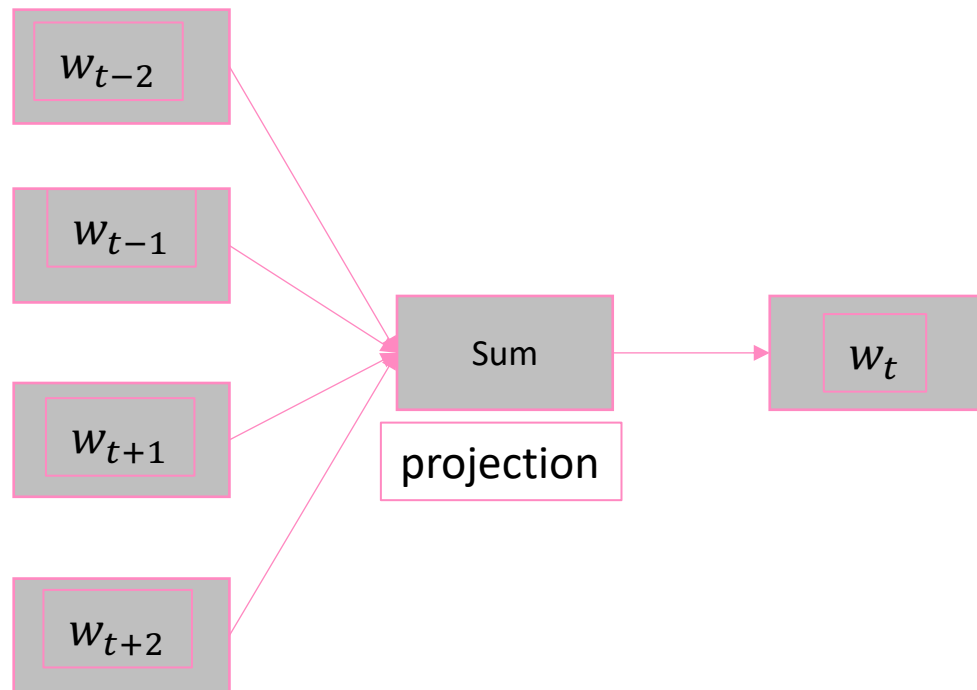
$u_o^T v_c$  - с помощью  
скалярного  
произведения  
определяем  
схожесть  
центрального слова  
и контекстного.

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)} = p \rightarrow \max$$

Нормализуем на  
сумму вероятностей  
по всем словам из  
словаря – получаем  
значение условной  
вероятности  $P(o|c)$

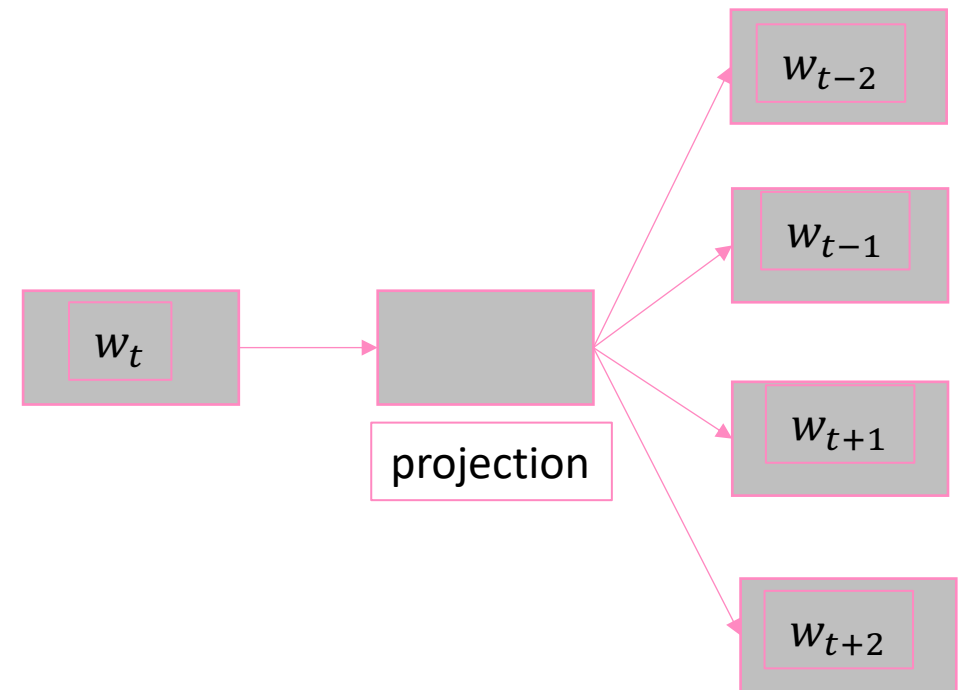
## CBOW

Предсказание центрального слова  $v_c$  по контексту



## Skip-gram

Предсказание распределения по центральному слову по контекстным словам



# Производные word2vec(skip-gram)

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m} \log P(w_{t+j} | w_t; \theta) \rightarrow \min \quad P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

$$\frac{d}{dv_c} \log(P(o|c)) = \frac{d}{dv_c} \log \left( \frac{\exp(u_o^T v_c)}{\sum_{w=1}^V \exp(u_w^T v_c)} \right) = \underbrace{\frac{d}{dv_c} \log(\exp(u_o^T v_c))}_1 - \underbrace{\frac{d}{dv_c} \log \left( \sum_{w=1}^V \exp(u_w^T v_c) \right)}_2$$

1.  $\frac{d}{dv_c} \log(\exp(u_o^T v_c)) = \frac{d}{dv_c} u_o^T v_c = u_o$

$$\frac{d}{dx} f(g(x)) = \frac{df}{dg} * \frac{dg}{dx}$$

2.  $\frac{d}{dv_c} \log \left( \sum_{w=1}^V \exp(u_w^T v_c) \right) = \frac{1}{\sum_{w=1}^V \exp(u_w^T v_c)} * \frac{d}{dv_c} \sum_{x=1}^V \exp(u_x^T v_c) = \frac{1}{\sum_{w=1}^V \exp(u_w^T v_c)} * \sum_{x=1}^V \frac{d}{dv_c} \exp(u_x^T v_c) =$

$$= \frac{1}{\sum_{w=1}^V \exp(u_w^T v_c)} * \sum_{x=1}^V u_x \exp(u_x^T v_c) = \sum_{x=1}^V \left( \frac{u_x \exp(u_x^T v_c)}{\sum_{w=1}^V \exp(u_w^T v_c)} \right) u_x = \sum_{x=1}^V p(x|c) u_x$$

$$\frac{d}{dv_c} \log(P(o|c)) = u_o - \sum_{x=1}^V p(x|c) u_x = \text{observed} - \text{predicted}$$

Задача w2v – минимизировать разницу между истинным значением слова и мат. Ожиданием по всем словам словаря при условии данного центрального слова




# Алгоритм обучения word2vec

Цель обучения – минимизировать разницу между истинным значением центрального слова/контекста и предсказанным с помощью кросс-энтропии. Близость предсказанного слова по центру/контексту к истинному слову обозначает способность модели определять семантику слова.

## CBOW

Вход модели

(векторы контекстных слов  $x_1, \dots, x_i$  размера  $D$ , вектор центрального слова  $y_j$ , окно  $W$ )



$$\tilde{x} = \sum_{i=0}^W x_i, \tilde{x} \in R^{1 \times D}$$
$$\text{logits} = \tilde{x} \cdot M, M \in R^{D \times V}$$
$$\hat{y} = \text{softmax}(\text{logits})$$
$$L(y, \hat{y}) = - \sum_{j=1}^{|V|} y_j \log(\hat{y}) \rightarrow \min$$



$$Q = W * D + D * V$$

## Skip-gram

Вход модели

(вектор центрального слова  $x_i$  размера  $D$ , вектор контекстного слова  $y_j$ , окно  $W$ )


$$\text{logits} = x_j \cdot M, M \in R^{D \times V}$$
$$\hat{y} = \text{softmax}(\text{logits})$$
$$L(y, \hat{y}) = - \sum_{j=1}^{|V|} \sum_{k=0}^W y_j \log(\hat{y}) \rightarrow \min$$


$$Q = W * (D + D * V)$$

Результат – матрица эмбедингов слов  $V^T$  из корпуса для решения down-stream задач

# Результаты обучения word2vec

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Первые 5 - семантические запросы, остальные - синтаксические

Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

В среднем – skip-gram показывает лучшее качество.

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days]
			Semantic	Syntactic	Total	
3 epoch CBOW	300	783M	15.5	53.1	36.1	1
3 epoch Skip-gram	300	783M	50.0	55.9	53.3	3
1 epoch CBOW	300	783M	13.8	49.9	33.6	0.3
1 epoch CBOW	300	1.6B	16.1	52.6	36.1	0.6
1 epoch CBOW	600	783M	15.4	53.3	36.2	0.7
1 epoch Skip-gram	300	783M	45.6	52.2	49.2	1
1 epoch Skip-gram	300	1.6B	52.2	55.1	53.8	2
1 epoch Skip-gram	600	783M	56.7	54.5	55.5	2.5

CBOW обучается быстрее чем Skip-Gram и дает сравнимое качество на задаче сопоставления синтаксических отношений.

Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
Collobert-Weston NNLM	50	660M	9.3	12.3	11.0
Turian NNLM	50	37M	1.4	2.6	2.1
Turian NNLM	200	37M	1.4	2.2	1.8
Mnih NNLM	50	37M	1.8	9.1	5.8
Mnih NNLM	100	37M	3.3	13.2	8.8
Mikolov RNNLM	80	320M	4.9	18.4	12.7
Mikolov RNNLM	640	320M	8.6	36.5	24.6
Huang NNLM	50	990M	13.3	11.6	12.3
Our NNLM	20	6B	12.9	26.4	20.3
Our NNLM	50	6B	27.9	55.8	43.2
Our NNLM	100	6B	34.2	<b>64.5</b>	50.8
CBOW	300	783M	15.5	53.1	36.1
Skip-gram	300	783M	<b>50.0</b>	55.9	<b>53.3</b>

Сравнение NNLM, CBOW и Skip-Gram по объему датасета и размерности вектора

# Hard-negative sampling

В skip-gram стоит задача максимизации правдоподобия  $p(o|c)$

$$L(\theta) = \prod_{c \in \text{Text}} \left[ \prod_{o \in O(c)} P(o|c; \theta) \right] = \prod_{t=1}^T \prod_{-m \leq j \leq m} P(w_{t+j} | w_t; \theta) \rightarrow \max$$

Мы можем это моделировать через

$$P(o|c) = \operatorname{argmax}_{\theta} \sum_{(c,o) \in D} \log\left(\frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}\right) = \sum_{(c,o) \in D} (\log(\exp(u_o^T v_c)) - \log(\sum_{w \in V} \exp(u_w^T v_c)))$$

Очень дорого считать

**Как можно заменить функцию оптимизации на более эффективную, сохраняя ее смысл?** При моделировании исходной функции мы используем данные из одного контекста –  $p(D = 1|c, o)$ . В таком случае, при максимизации правдоподобия:

$$\operatorname{argmax}_{\theta} p(D = 1|c, o, \theta) = \frac{1}{1 + e^{-v_c v_o}}$$

В таком случае, при достаточно большом  $\theta$  -  $p(D = 1|c, o, \theta) = 1$  для каждой пары  $c, o$ . Добавим в уравнение новое слагаемое, которые наложит ограничения на некоторые комбинации  $c, o$  – с некоторой вероятностью будем брать слова из разных контекстов, т.е.  $p(D = 0|c, o, \theta) = 1 - p(D = 1|c, o, \theta)$ .

$$\operatorname{argmax}_{\theta} \prod_{(c,o) \in D} p(D = 1|c, o, \theta) \prod_{(c,o) \in D'} p(D = 0|c, o, \theta) = \operatorname{argmax}_{\theta} \prod_{(c,o) \in D} p(D = 1|c, o, \theta) \prod_{(c,o) \in D'} 1 - p(D = 1|c, o, \theta)$$

# Hard-negative sampling

$$\begin{aligned}
 \operatorname{argmax}_{\theta} \prod_{(c,o) \in D} p(D = 1|c, o, \theta) \prod_{(c,o) \in D'} p(D = 0|c, o, \theta) &= \operatorname{argmax}_{\theta} \prod_{(c,o) \in D} p(D = 1|c, o, \theta) \prod_{(c,o) \in D'} 1 - p(D = 1|c, o, \theta) \\
 &= \operatorname{argmax}_{\theta} \prod_{(c,o) \in D} \log\left(\frac{1}{1 + e^{-v_c v_o}}\right) + \prod_{(c,o) \in D'} \log\left(1 - \frac{1}{1 + e^{-v_c v_o}}\right) \\
 &= \operatorname{argmax}_{\theta} \prod_{(c,o) \in D} \log\left(\frac{1}{1 + e^{-v_c v_o}}\right) + \prod_{(c,o) \in D'} \log\left(\frac{1}{1 + e^{v_c v_o}}\right)
 \end{aligned}$$

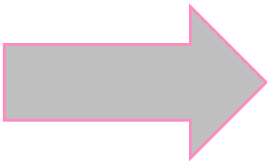
При  $\sigma(x) = \frac{1}{1+e^{-x}}$

$$= \operatorname{argmax}_{\theta} \prod_{(c,o) \in D} \log \sigma(v_c v_o) + \prod_{(c,o) \in D'} \log \sigma(-v_c v_o)$$

1 epoch Skip-gram	300	783M	45.6	52.2	49.2	1
1 epoch Skip-gram	300	1.6B	52.2	55.1	53.8	2
1 epoch Skip-gram	600	783M	56.7	54.5	55.5	2.5

Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56



Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	<b>61</b>
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53

Проблема word2vec – обучение эмбеддингов происходит без учета количества слов, встречающихся вместе. Часто встречающиеся вместе слова дают больше одинаковых тренировочных пар, не добавляющих информации при обучении. Идея Glove – обучение модели опираясь на матрицу частотности встречаемости слов в контексте. Glove обучает матрицу эмбеддингов таким образом, что скалярное произведение векторов пропорционально логарифму вероятности совпадений.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k ice)$	$1.9 \times 10^{-4}$	$6.6 \times 10^{-5}$	$3.0 \times 10^{-3}$	$1.7 \times 10^{-5}$
$P(k steam)$	$2.2 \times 10^{-5}$	$7.8 \times 10^{-4}$	$2.2 \times 10^{-3}$	$1.8 \times 10^{-5}$
$P(k ice)/P(k steam)$	8.9	$8.5 \times 10^{-2}$	1.36	0.96

$X_{ij}$  - количество раз слово  $j$  встречается в контексте слова  $i$

$X_i = \sum_k X_{ik}$  - кол-во слов, которое встречается в контексте слова  $i$

$P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}$  - вероятность, что слово  $j$  встретиться в контексте слова  $i$



	$x = solid$	$x = gas$	$x = water$	$x = random$
$P(x ice)$	large	small	large	small
$P(x steam)$	small	large	large	small
$\frac{P(x ice)}{P(x steam)}$	large	small	$\sim 1$	$\sim 1$

Отношение вероятности встречаемости слов лучше отличает релевантные слова(solid и gas) от нерелевантных(water и fashion), фтакже позволяет различать между релевантными словами.

# Функция ошибки в Glove

Задача – подобрать такую функцию, которая будет пропорциональна отношению вероятности встречаемости слов:

$$F(w_i, w_j, \widetilde{w}_k) = \frac{P_{ik}}{P_{jk}}, w_i, w_j - \text{центральные слова}, \widetilde{w}_k - \text{контекст}$$

Требования к функции:

- Учитывать в векторном пространстве отношение вероятностей. Так как векторные пространства – линейные структуры, воспользуемся разностью векторов.
- Справа у нас скаляр, аргументы функции – вектора. Нам необходимо сохранить линейную структуру, которую мы пытаемся создать- воспользуемся скалярное произведением.

$$F((w_i - w_j)^T \widetilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

Также потребуем от функции гомоморфности, чтобы при изменении классов – контекст/центр – ничего не менялось.

$$F((w_i - w_j)^T \widetilde{w}_k) = \frac{F(w_i^T \widetilde{w}_k)}{F(w_j^T \widetilde{w}_k)} \rightarrow F(w_i^T \widetilde{w}_k) = \frac{X_{ik}}{X_i}$$

Нам подходит  $F = \exp, \log(X_i)$  можно перекинуть в свободный член  $b_i$ , для восстановления симметрии добавив  $\widetilde{b}_k$

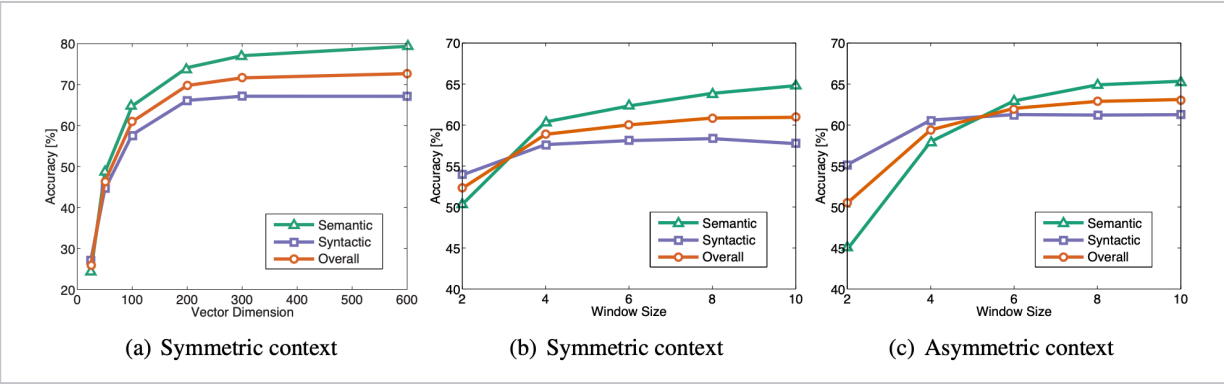
$$e^{w_i^T \widetilde{w}_k} = P_{ik} \rightarrow w_i^T \widetilde{w}_k = \log(X_{ik}) - \log(X_i)$$

$$w_i^T \widetilde{w}_k + b_i + \widetilde{b}_k = \log(X_{ik})$$

Если аргумент логарифма 0, то функция ошибки расходится – добавим еще множитель весовой функции, который зависит от параметров  $x_{max}$  и  $\alpha$ .

$$J = \sum_{i,k=1}^V f(X_{ik}) ((w_i^T \widetilde{w}_k + b_i + \widetilde{b}_k - \log(X_{ik}))^2 \quad f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^\alpha, & \text{if } x < x_{max} \\ 1, & \text{otherwise} \end{cases}$$

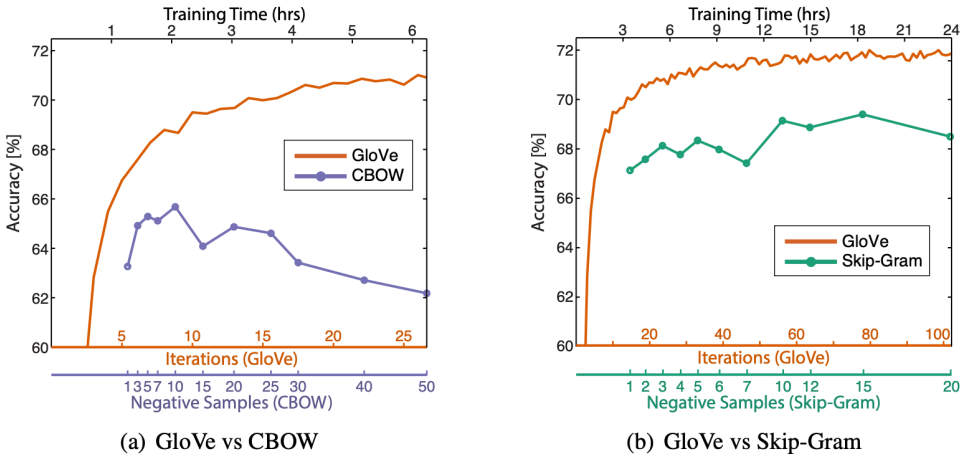
# Glove результаты



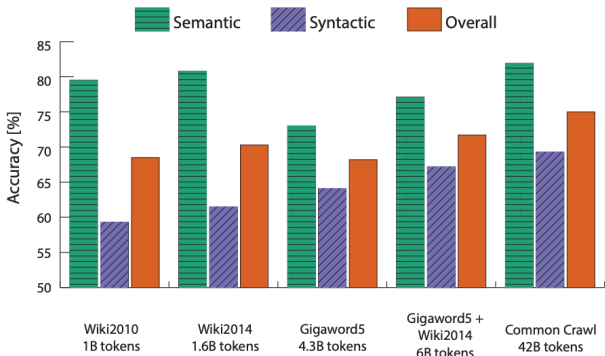
Качество работы модели на задаче построения аналогий(semantic - Москва к России, как Берлин к \_\_\_\_, syntactic - танец к танцевать, как игра к \_\_\_\_) в зависимости от размера вектора и окна.

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW <sup>†</sup>	6B	57.2	65.6	68.2	57.0	32.5
SG <sup>†</sup>	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	75.9	83.6	82.9	59.6	47.8
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

Сравнение результатов модели(spearman rank corr) на задаче сравнения слов с различными методами SVD, CBOW & Skip-gram.



Сравнение времени обучения и качества Glove по сравнению с word2vec.



Сравнение результатов модели на задаче аналогий при обучении на разных корпусах.

