

Глубокое обучение и вообще

Ульянкин Филипп

13 марта 2021 г.

Посиделка 10: Из слов в вектора и обратно

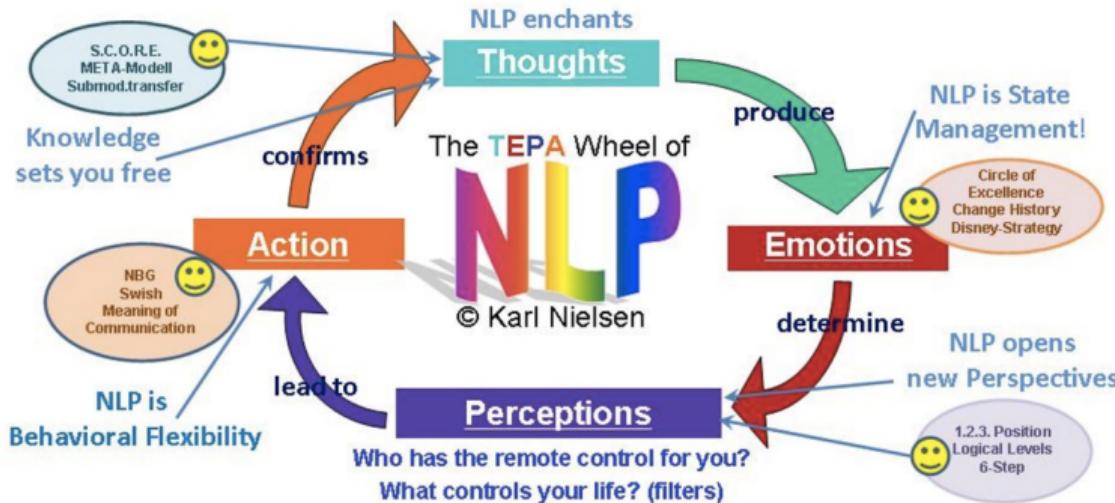
Agenda

- Небольшое введение в анализ текстов
- Представления для текстов, эмбединги
- Свёрточные нейросети для текстов

NLP (Natural Language Processing)



NLP, the freedom in Thinking, Feeling, Perceiving and Behavior





freedom in thinking, receiving, behavior

Feelings

S.C.O.R.E.
META-Modell
Submod.transfer

Knowledge
sets you free

confirms

NBG
Swish
Meaning of
Communication

NLP is
Behavioral F

Action

produce

Emotions

NLP is State
Management!

Circle of
Excellence
Change History
Disney-Strategy

Karl Nielsen

Perceptions

Who has the remote control for you?
What controls your life? (filters)

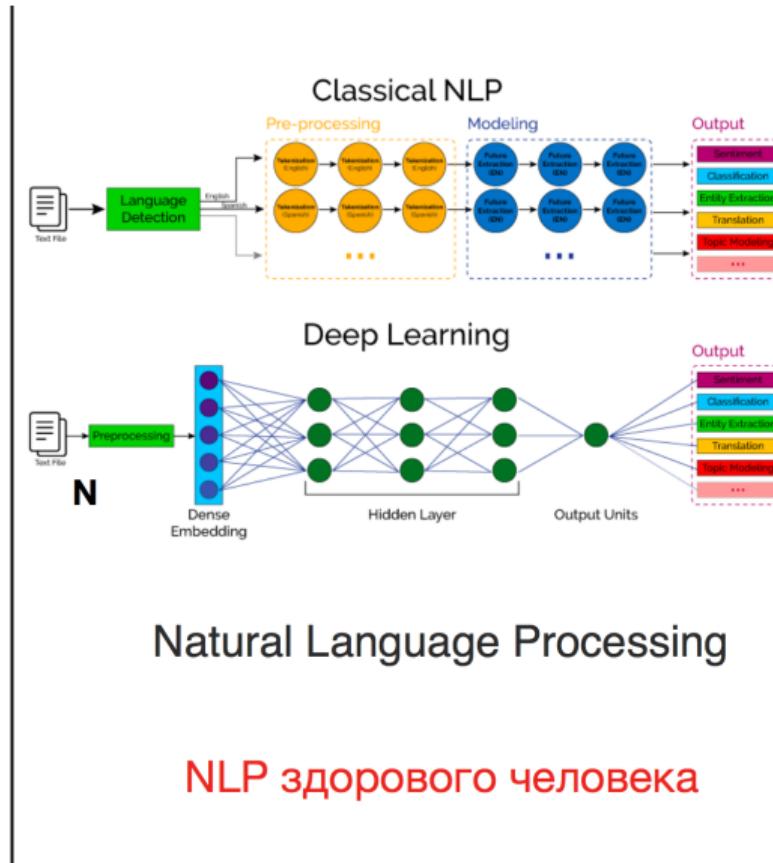
NLP opens
new Perspectives

1,2,3, Position
Metaphorical Levels
One-Step



Neuro-linguistic programming

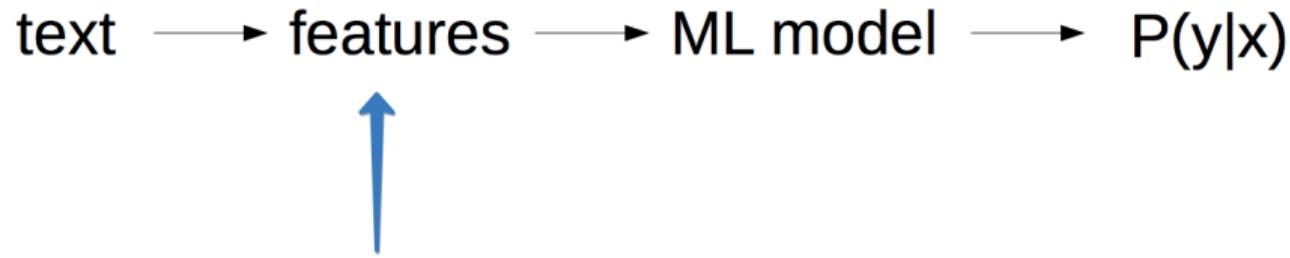
NLP курильщика



Natural Language Processing

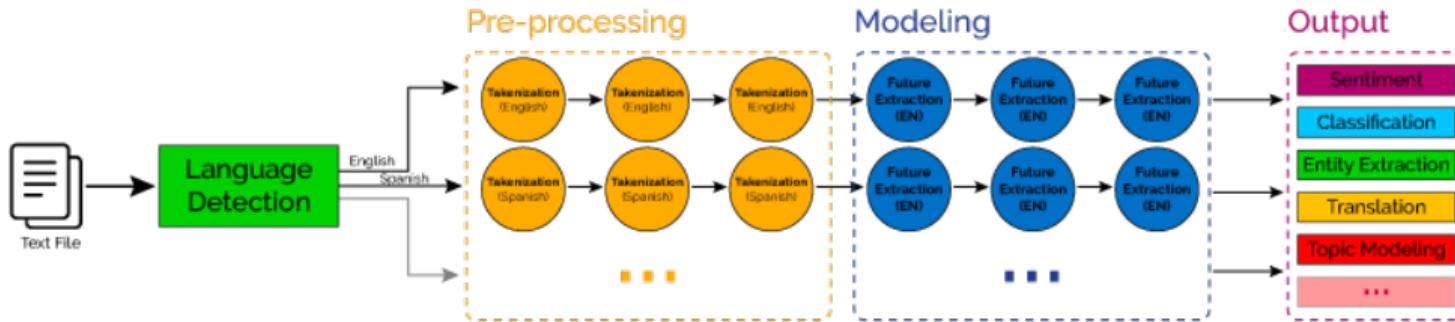
NLP здорового человека

Модели на текстах

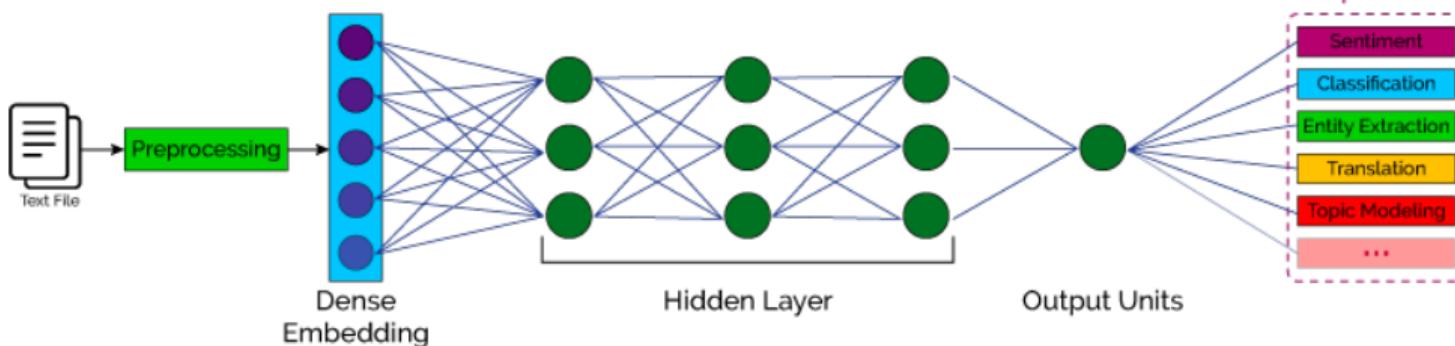


Как представить текст
в виде, который могла
бы понять модель?

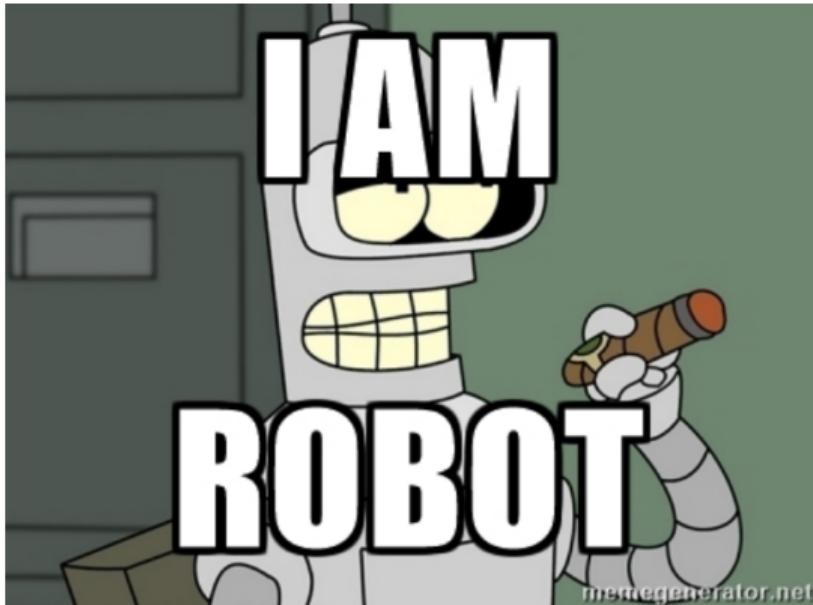
Classical NLP



Deep Learning



Что такое текст?



- Текст (документ) — это последовательность токенов (слов)
- Токен (слово) — это последовательность символов

One-Hot Encoding

1. Нежился на пляже
2. Копали яму на пляже
3. Копал картошку
4. Ел картошки и картошку



	нежиться	пляж	копать	яма	картошка	есть
1	1	1	0	0	0	0
2	0	1	1	1	0	0
3	0	0	1	0	1	0
4	0	0	0	0	2	1

One-Hot Encoding

- В словаре много слов, у нас будет слишком много признаков
- В векторе нет никакой информации о смысле слова
- Семантически похожие тексты могут иметь очень разные представления
- Непонятно, что делать, если в тексте появляется какое-то новое слово

Анализ текстов в одном слайде



Порядок слов неважен

Неважен слов порядок

Слов порядок неважен

Он был хорошим человеком и джедаем.
Люди держат деньги в банке.
Падаван, дай мне огурец из банки.



1. токенизация
2. очистка от стоп-слов

[~~он~~, был, хорошим, человеком, ~~и~~, джедаем]
[люди, держат, деньги, ~~в~~, банке]
[падаван, дай, мне, огурец, ~~из~~, банки]

лемматизация

[быть, хороший, человек, джедай]
[человек, держать, деньги, банк]
[падаван, дать, огурец, банка]

3. нормализация

стемминг

[бы, хорош, чел, джед]
[люд, держ, ден, банк]
[падаван, дай, огур, банк]

5. ОНЕ или tf-idf,
а затем
моделирование

4. очистка от слишком редких слов

Гипотеза мешка слов

- **Гипотеза мешка слов:** нам плевать на взаимное расположение слов. Порядок слов в предложении никак не сказывается на его смысле. **Следуя гипотезе, мы теряем часть информации.**
- Рассматриваем каждое слово, как переменную \Rightarrow большое пространство признаков. Нужно его урезать \Rightarrow **Теряем ещё информацию.**
- Хотим маленькое пространство признаков и много информации в нём!

Дистрибутивная гипотеза

- **Дистрибутивная гипотеза:** слова с похожим смыслом будут встречаться в похожих контекстах.
- Мы можем попытаться заложить в векторное представление слова информацию о его контексте
- Есть разные подходы: count-based и prediction-based, мы поговорим о втором

Из слов в вектора и обратно



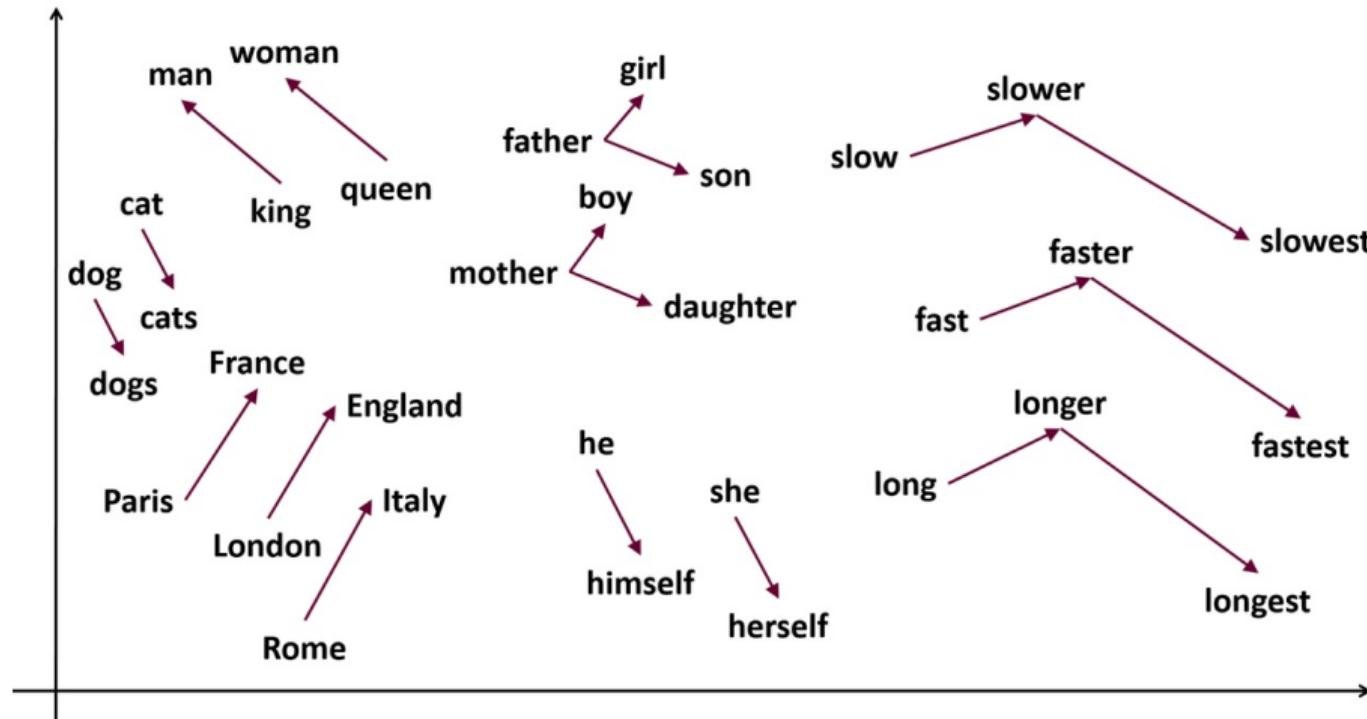
Words embeddings

- **Наша цель:** хотим научить компьютер понимать слова
- Идея! Давайте превратим наши слова в вектора размера d
- На вектора понакладываем хотелок!



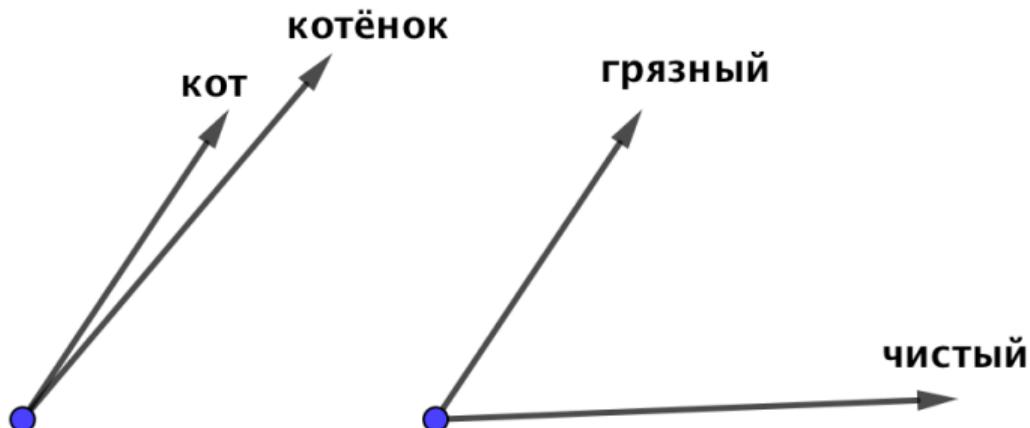
Хотелка первая

- Хотим, чтобы модель улавливала семантические свойства слов



Хотелка вторая

- Модель понимала, где близкие по смыслу слова: кот, котёнок, кошка, тигр, лев, ...



Хотелка третья

- Арифметика!



$$- \text{ (pink silhouette)} + \text{ (blue silhouette)} = \text{ (Khal Drogo)}$$



Томаш Миколов

- Звучит как магия, но правда работает
- В 2013 году модель предложена чешским аспирантом Томашем Миколовым
- После работал в Google, сейчас ушёл в Facebook



<https://arxiv.org/abs/1301.3781>

Основная идея

- Если выкинуть слово, то оно должно хорошо восстанавливаться по представлениям соседних слов (контексту)
- Будем обучать представления слов, выкидывая слова из середины и пытаясь восстановить их

Постановка задачи

контекст
к слов до
к слов после

Вчера **на обед Король Лев съел Пумбу**

W1 W2

W0

W3 W4

- Слово **Король** мы хотим предсказать по его контексту
- w_i - представление слова в виде вектора, его эмбединг

Постановка задачи

контекст

к слов до

к слов после

$$w_c = \frac{w_1 + w_2 + w_3 + w_4}{4}$$

Вчера на обед Король Лев съел Пумбу

w1

w2

w0

w3

w4

- Контекст — k слов до рассматриваемого и k после него.

Постановка задачи

контекст
к слов до
к слов после

$$w_c = \frac{w_1 + w_2 + w_3 + w_4}{4}$$

Вчера на обед Король Лев съел Пумбу

W1 W2

W0

W3

W4

- Определим вероятность встретить наше слово в контексте c :

$$P(w_0 | w_c) = \frac{e^{w_0^T w_c}}{\sum_i e^{w_i^T w_c}}$$

Метод максимального правдоподобия

Падаван, в будущем ты станешь джедаем, как твой учитель. Он был хорошим вуки и джедаем.

- Какова вероятность встретить в тексте слово джедай, p - ?

Метод максимального правдоподобия

*Падаван, в будущем ты станешь **джедаем**, как твой учитель. Он был хорошим вуки и **джедаем**.*

- Какова вероятность встретить в тексте слово джедай, p - ?
- Всего 15 слов, джедай встретился дважды

Метод максимального правдоподобия

*Падаван, в будущем ты станешь **джедаем**, как твой учитель. Он был хорошим вуки и **джедаем**.*

- Какова вероятность встретить в тексте слово джедай, p - ?
- Всего 15 слов, джедай встретился дважды
- Более формально, мы можем выписать правдоподобие:

$$L(p) = (1-p) \cdot (1-p) \cdot (1-p) \cdot (1-p) \cdot (1-p) \cdot \textcolor{brown}{p} \cdot (1-p) \cdots \\ \cdots \cdot (1-p) \cdot \textcolor{brown}{p} = (1-p)^{13} \cdot p^2 \rightarrow \max_p$$

Метод максимального правдоподобия

- Вероятность встретить наше слово в контексте c :

$$P(w_0 \mid w_c) = \frac{e^{w_0^T w_c}}{\sum e^{w_i^T w_c}}$$

- Правдоподобие для нашего текста:

$$L(W) = \prod_{i,c} P(w_i \mid w_c) \rightarrow \max_W$$

$$\ln L(W) = \sum_{i,c} \ln P(w_i \mid w_c) \rightarrow \max_W$$

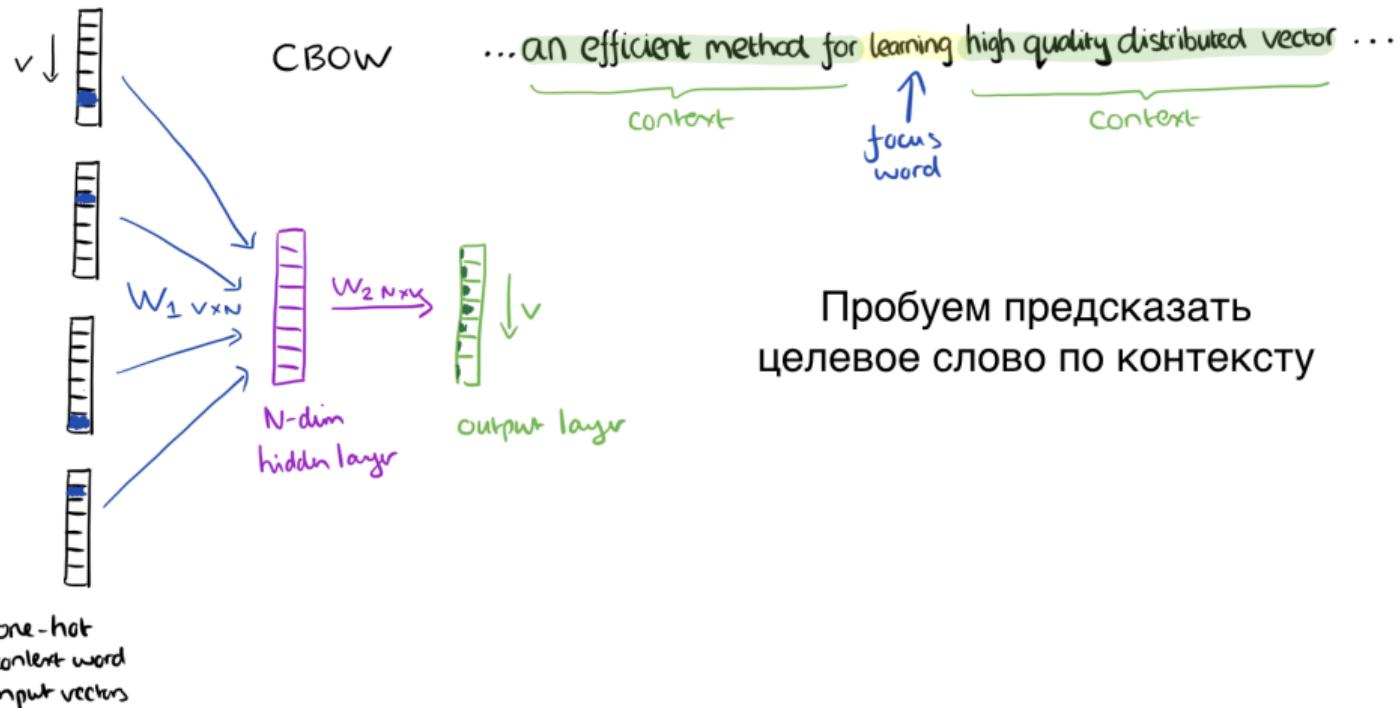
- Мы хотим настроить вектора W для слов так, чтобы $P(w_0 \mid w_c)$ была высокой, если слово w_0 встречается в контексте w_c

Проблемы

- Очень много параметров W , по каждому брать производную?!
- Любая взвешенная сумма — нейросеть, давайте запишем правдоподобие в виде нейросети и обучим её, для этого есть два подхода
- Для простоты будем использовать две матрицы: W_1 для контекста слова и W_2 для векторов
- СВОУ (непрерывный мешок слов) — в нём мы по заданному контексту слова пытаемся предсказать слово
- Skip-gram — по заданному слову пытаемся предсказать его контекст

Обучение модели

CBOW

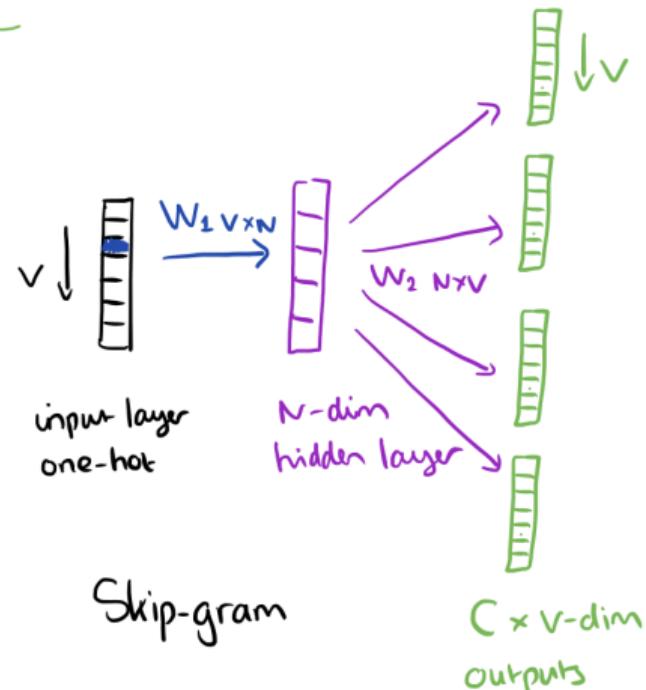


Skip-gram

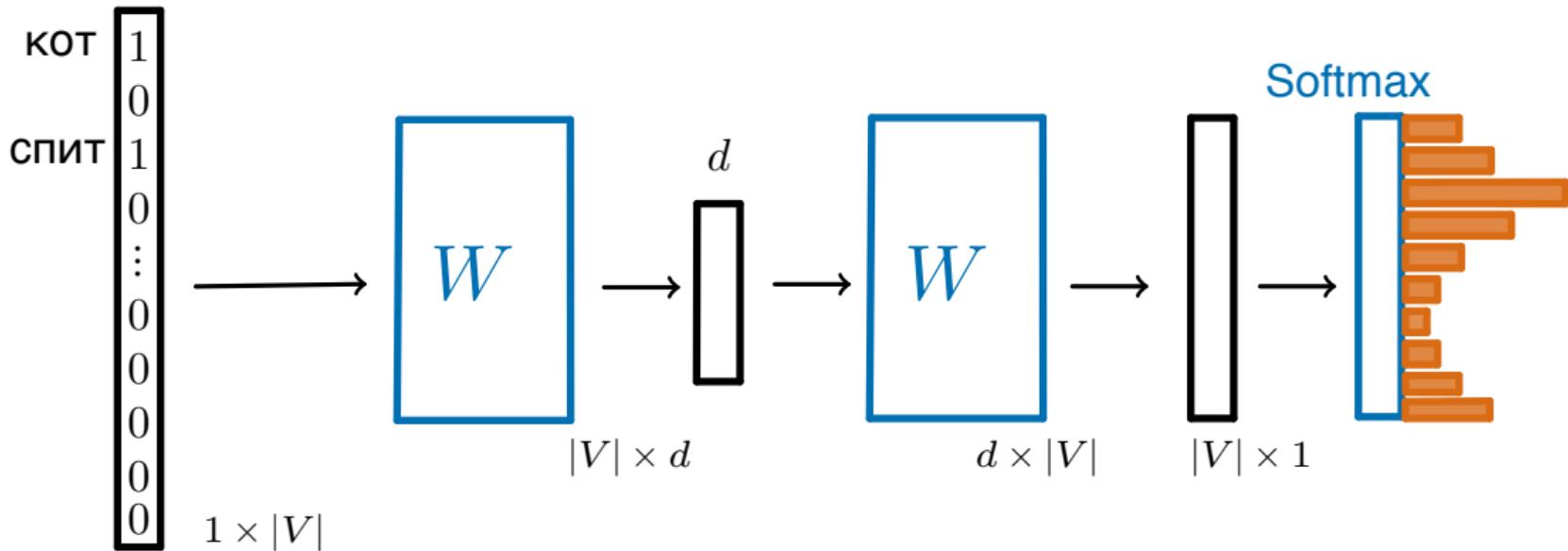
...an efficient method for learning high quality distributed vector ...



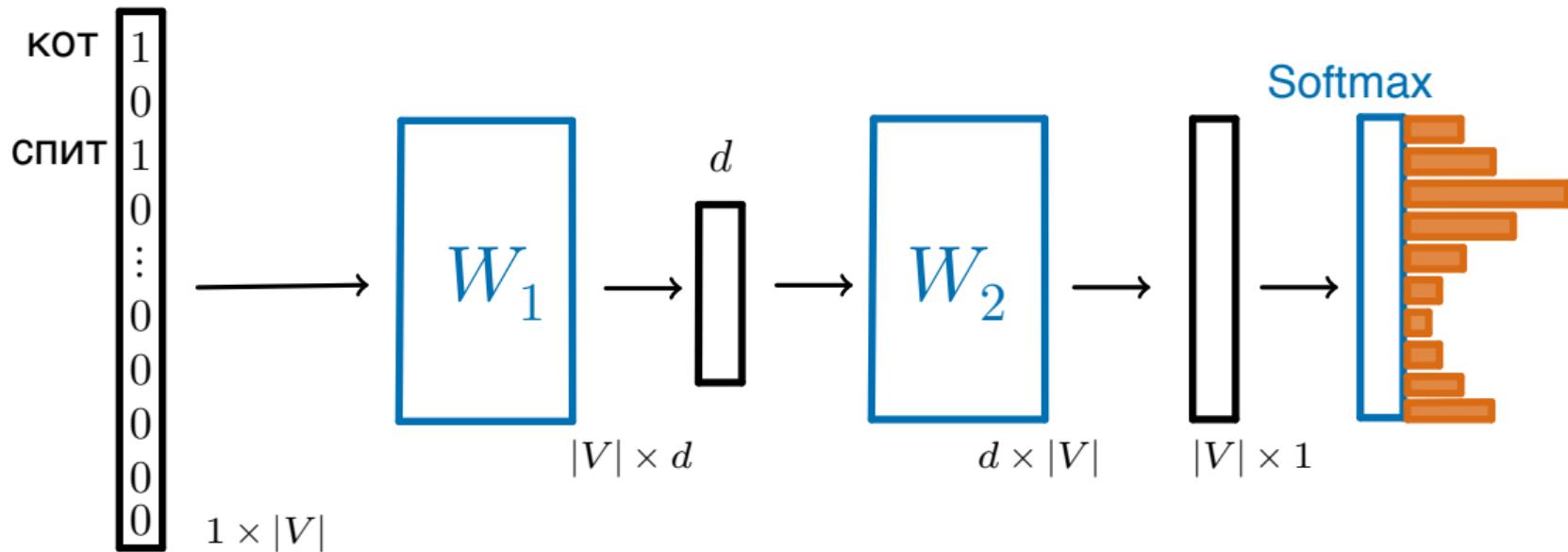
Пробуем предсказать
Контекст по целевому слову



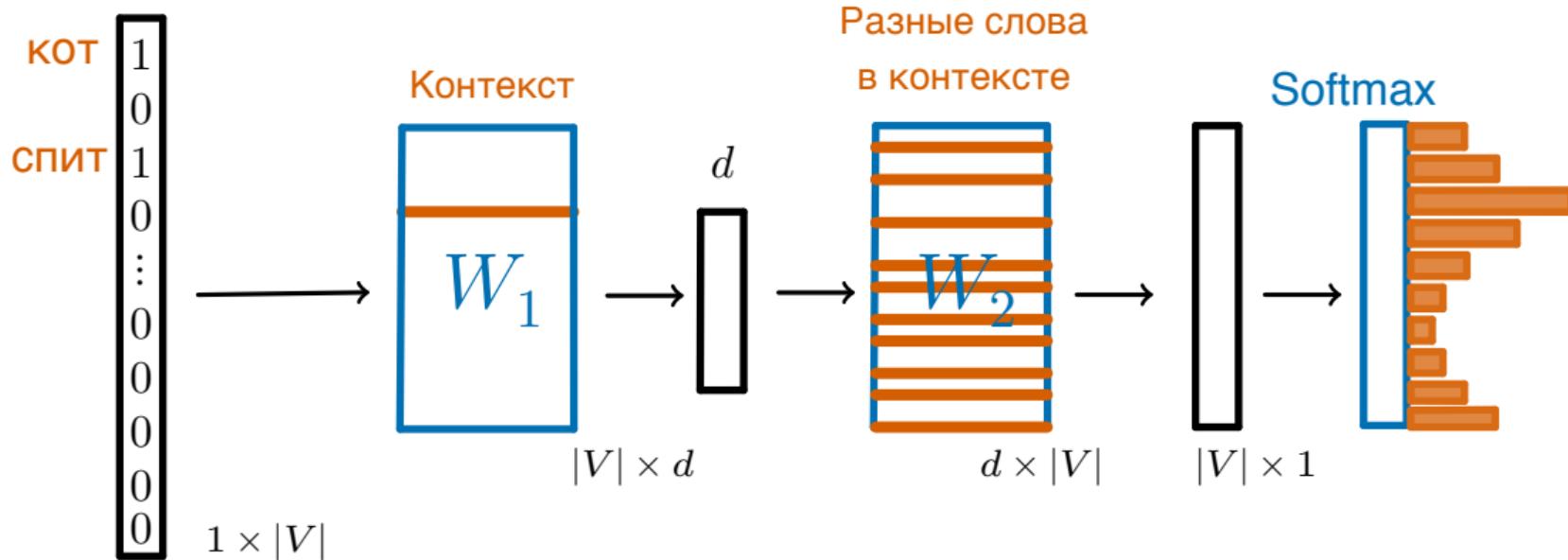
CBOW



CBOW



CBOW



Обучение модели

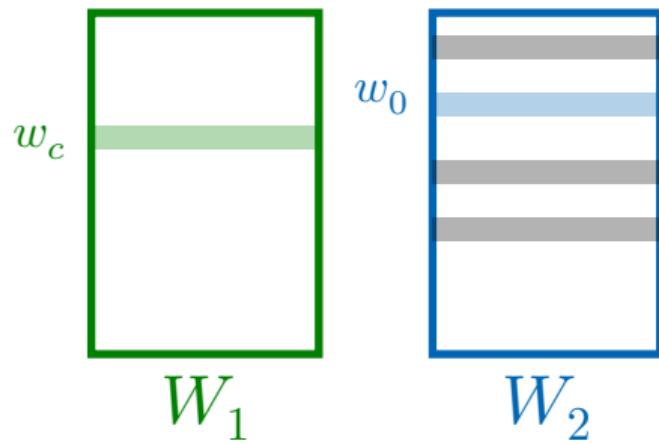
- Трансформирует пространство текстов в d -мерное пространство векторов
- Для обучения требует много данных
- Обычно при обучении контекст не ищут, как среднее, а просто заводят вторую матрицу
- В матрице W_2 будут записаны наши итоговые вектора для слов, в матрице W_1 вектора для контекстов

Один шаг обучения в деталях

$$\sum_{i,c} \ln P(\textcolor{blue}{w}_0 \mid \textcolor{green}{w}_c) \rightarrow \max_W \quad - \sum_{i,c} \ln P(\textcolor{blue}{w}_0 \mid \textcolor{green}{w}_c) \rightarrow \min_W$$

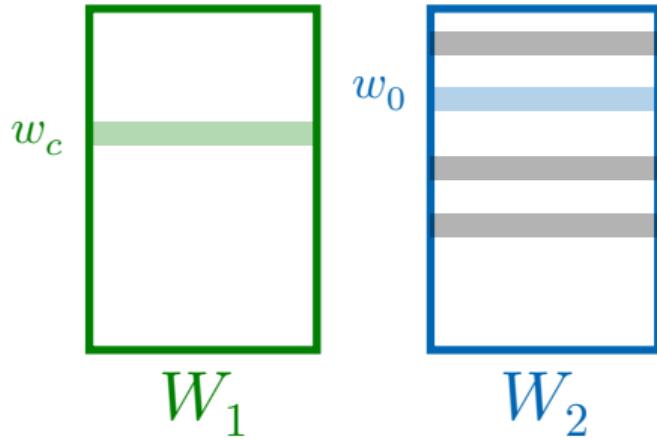
Один шаг обучения в деталях

$$J = -\ln P(\mathbf{w}_0 \mid \mathbf{w}_c)$$



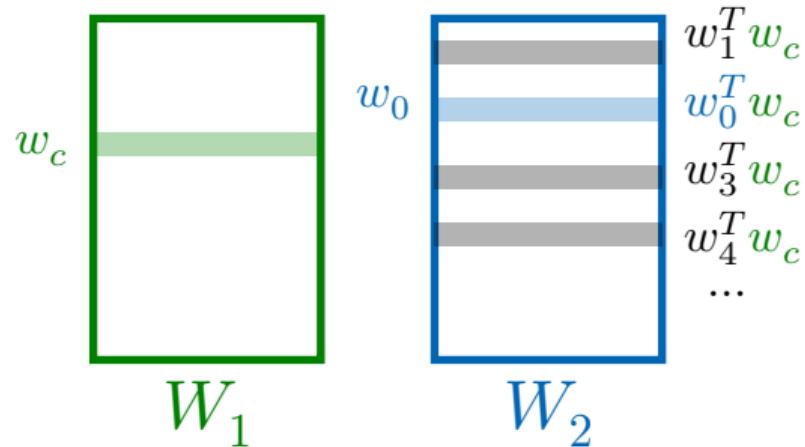
Один шаг обучения в деталях

$$J = -\ln P(\mathbf{w}_0 \mid \mathbf{w}_c) = -\ln \frac{\exp(\mathbf{w}_0^T \mathbf{w}_c)}{\sum_i \exp(\mathbf{w}_i^T \mathbf{w}_0)}$$



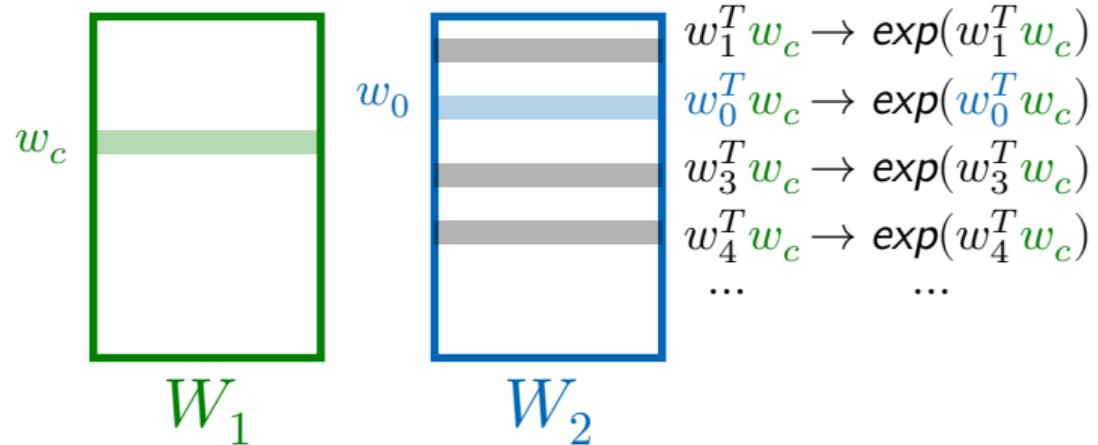
Один шаг обучения в деталях

$$J = -\ln P(\mathbf{w}_0 \mid \mathbf{w}_c) = -\ln \frac{\exp(\mathbf{w}_0^T \mathbf{w}_c)}{\sum_i \exp(\mathbf{w}_i^T \mathbf{w}_0)} = -\mathbf{w}_0^T \mathbf{w}_c + \ln \sum_i \exp(\mathbf{w}_i^T \mathbf{w}_c)$$



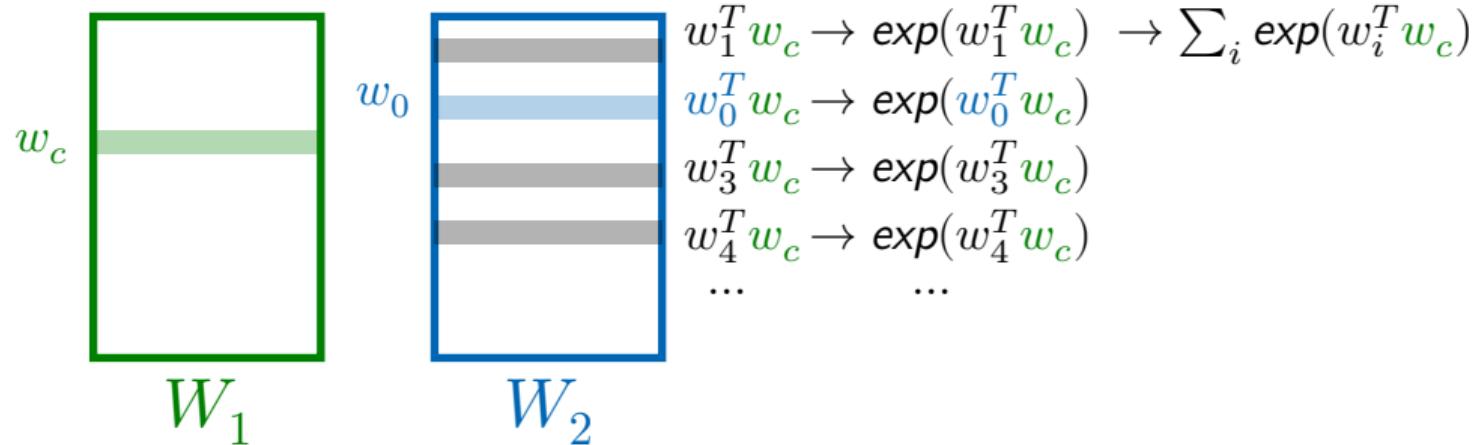
Один шаг обучения в деталях

$$J = -\ln P(\mathbf{w}_0 \mid \mathbf{w}_c) = -\ln \frac{\exp(\mathbf{w}_0^T \mathbf{w}_c)}{\sum_i \exp(\mathbf{w}_i^T \mathbf{w}_0)} = -\mathbf{w}_0^T \mathbf{w}_c + \ln \sum_i \exp(\mathbf{w}_i^T \mathbf{w}_c)$$



Один шаг обучения в деталях

$$J = -\ln P(\mathbf{w}_0 \mid \mathbf{w}_c) = -\ln \frac{\exp(\mathbf{w}_0^T \mathbf{w}_c)}{\sum_i \exp(\mathbf{w}_i^T \mathbf{w}_0)} = -\mathbf{w}_0^T \mathbf{w}_c + \ln \sum_i \exp(\mathbf{w}_i^T \mathbf{w}_c)$$



Один шаг обучения в деталях

Функция потерь:

$$J = -\ln P(w_0 \mid w_c) = -\ln \frac{\exp(w_0^T w_c)}{\sum_i \exp(w_i^T w_0)} = -w_0^T w_c + \ln \sum_i \exp(w_i^T w_c)$$

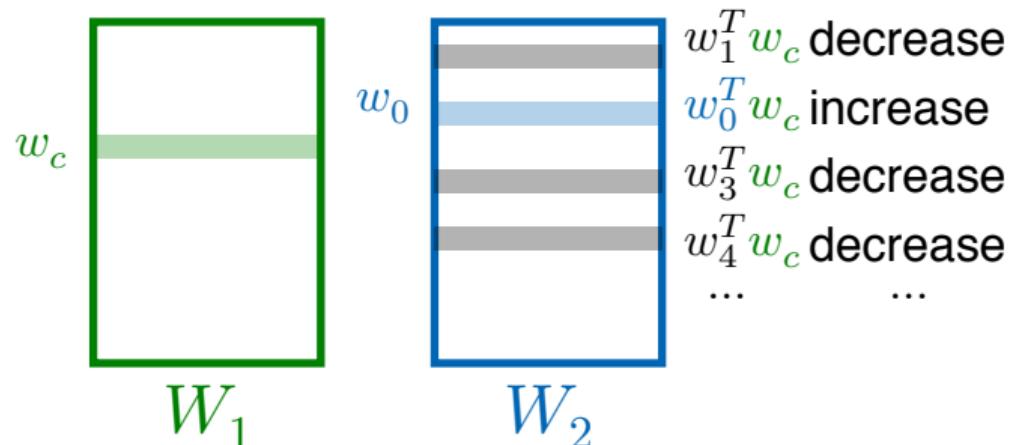
Шаг градиентного спуска:

$$\begin{aligned} w_c &= w_c - \gamma \cdot \frac{\partial J}{\partial w_c} \\ w_i &= w_i - \gamma \cdot \frac{\partial J}{\partial w_i} \quad \forall w_i \in W \end{aligned}$$

https://lena-voita.github.io/nlp_course.html#whats_inside

Один шаг обучения в деталях

$$J = -\ln P(w_0 \mid w_c) = -\ln \frac{\exp(w_0^T w_c)}{\sum_i \exp(w_i^T w_0)} = -w_0^T w_c + \ln \sum_i \exp(w_i^T w_c)$$



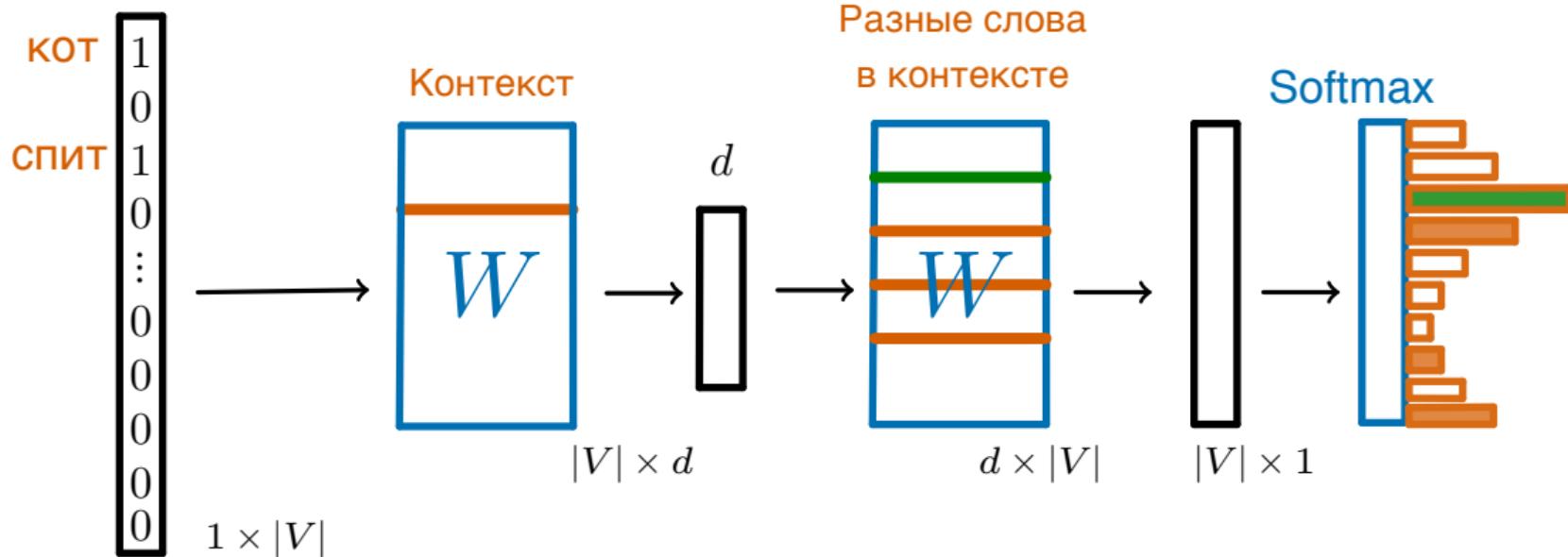
Обновляем один w_c и каждый w_i , то есть всего $|V| + 1$ параметр

https://lena-voita.github.io/nlp_course.html#whats_inside

Negative Sampling

- Нужно обновлять много параметров \Rightarrow медленное обучение
- Многие слова вместе не встречаются, поэтому большая часть вычислений избыточная
- Давайте максимизировать вероятность типичного контекста и минимизировать вероятность нетипичного контекста

Negative Sampling



Negative Sampling

- Придётся изменить функцию потерь
- Пусть $D = 1$, если слово вошло в контекст, тогда вероятность угадать слово по контексту:

$$P(D = 1 \mid \mathbf{w}_0, c) = \sigma(\mathbf{w}_0^T \cdot \mathbf{w}_c) = \frac{1}{1 + e^{-\mathbf{w}_0^T \cdot \mathbf{w}_c}}$$

$$P(D = 0 \mid \mathbf{w}_j, c) = 1 - \sigma(\mathbf{w}_j^T \cdot \mathbf{w}_c) = \frac{1}{1 + e^{\mathbf{w}_j^T \cdot \mathbf{w}_c}}$$

- Новая функция потерь:

$$-\ln \sigma(\mathbf{w}_0^T \cdot \mathbf{w}_c) + \sum_{w_j \in \{\mathbf{w}_1, \dots, \mathbf{w}_K\}} \ln(\sigma(-\mathbf{w}_j^T \cdot \mathbf{w}_c))$$

Гиперпараметры

- Размер эмбединга (исторически 300, но варианты 100, 500 тоже возможно)
- Число наблюдений для негативного сэмплирования (для маленьких датасетов 15 – 20, для больших 2 – 5)
- Окно контекста (обычно 5 – 10),
- Очень большое окно — похожесть топиков, можно вводить w2v через матричные разложения

https://lena-voita.github.io/nlp_course/word_embeddings.html

Полезные мысли про обучение

- В tensorflow довольно легко собрать свой собственный w2v и обучить его, но не стоит делать это. Ваша реализация не будет такой эффективной, как уже существующие специализированные реализации. За последние годы алгоритмы для обучения w2v претерпели существенную эволюцию.
- Реализация w2v из пакета gensim зачастую работает быстрее, чем модели, написанные в стандартных для нейросеток бэкэндах. Это происходит из-за многопоточности и разных умных оптимизаций тонких мест в обучении.

Полезные мысли про обучение

- При достаточно большом корпусе текстов можно не делать лемматизацию. Сетка сама поймёт по контексту, что слова близки и присвоит им похожие вектора.
- Если у вас специфическая задача, в которой встречается специфическая лексика, возьмите предобученную на большом корпусе сетку и дообучите её под свои нужды.
- w2v может выдавать эмбединги только для слов из заданного при обучении словаря. Этот минус можно попытаться побороть и получить другую модель, fasttext

Проблемы word2vec

- Не умеем работать с новыми словами, которых не было в нашем словаре при обучении
- Не закладываем никакой априорной информации о разных формах одного слова
- Обучаемся на контекст слова, но всё ещё действуем в парадигме мешка слов, никак не учитываем порядок слов
- Не учитываем структуру слов, не умеем обрабатывать опечатки

Как это использовать и где
раздобыть?

Как это использовать

- Можно искать похожие слова
- Можно менять формы слов
- Можно искать определённые отношения
- Можно использовать как признаки для моделей
- Обучение w2v — аналог transfer learning, но обучение идёт на фиктивную задачу, разметка, сделанная вручную, не нужна

Something2vec

- Эмбеддинги (embeddings) — это сопоставление произвольной сущности (например, узла в графе или кусочка картинки) некоторому вектору.
- Любую последовательность можно представить в виде эмбединга
- Последовательность банковских транзакций
- Веб-сессии (последовательность перехода по сайтам)
- Графы взаимосвязей между пользователями
- Любая категориальная переменная: порядок, в котором турист посещал города; порядок, в котором юзер отректировал сериалы и тп

something2vec



-



+



=



Где взять уже готовое (Google)

Google Code Archive

Search this site

Projects Search About

Project word2vec

Source

Issues Tool for computing continuous distributed representations of words.

Wikis

Downloads

Introduction

This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research.

Quick start

- Download the code: svn checkout <http://word2vec.googlecode.com/svn/trunk/>
- Run 'make' to compile word2vec tool
- Run the demo scripts: `./demo-word.sh` and `./demo-phrases.sh`
- For questions about the toolkit, see <http://groups.google.com/group/word2vec-toolkit>

Project Information

The project was created on Jul 30, 2013.

- License: Apache License 2.0
- 945 stars
- svn-based source control

Labels:

NeuralNetwork MachineLearning
NaturalLanguageProcessing WordVectors
Google

Гуглловская модель для английского языка: <https://code.google.com/archive/p/word2vec/>

Где взять уже готовое (проект RusVectōrēs)

Модели

В настоящий момент вы можете скачать следующие модели (жирным выделены модели, доступные для использования в веб-интерфейсе):

Таблицу можно (и нужно) пролистывать по горизонтали!

Уникальный идентификатор ▾	Скачать ▾	Корпус ▾	Размер корпуса ▾	Объём словаря	Частотный порог ▾	Таргет ▾	Алгоритм ▾	Размерность вектора ▾	Размер окна ▾
a_upos_skipgram_300_2_2018	331 Мбайт	Тайга	почти 5 миллиардов слов	237 255	200	Universal Tags	Continuous Skipgram	300	2
корpora_upos_skipgram_300_5_2018	191 Мбайт	НКРЯ	260 миллионов слов	195 071	20	Universal Tags	Continuous Skipgram	300	5
ikiuruscorpora_upos_skipgram_300_2_2018	376 Мбайт	НКРЯ и Википедия за декабрь 2017	600 миллионов слов	384 764	40	Universal Tags	Continuous Skipgram	300	2
rs_upos_cbow_600_2_2018	547 Мбайт	Русскоязычные новости, с сентября 2013 до ноября 2016	почти 5 миллиардов слов	289 191	200	Universal Tags	Continuous Bag-of-Words	600	2
araneum_upos_skipgram_300_2_2018	192 Мбайта	Araneum	около 10 миллиардов слов	196 620	400	Universal Tags	Continuous Skipgram	300	2
araneum_none_fasttextcbow_300_5_2018	1 Гбайт	Araneum	около 10 миллиардов слов	195 782	400	Нет	fastText CBOW (3.-5-граммы)	300	5
araneum_none_fasttextskipgram_300_5_2018	675 Мбайт	Araneum	около 10 миллиардов слов	195 782	400	Нет	fastText Skipgram /2-граммы/	300	5

Куча разных моделей для русского языка: <https://rusvectores.org/ru/models/>

Свойства word2vec



Частотность слова

- Высокая Средняя Низкая

НКРЯ и Wikipedia

- чай NOUN 0.56
- пиво NOUN 0.56
- самогон NOUN 0.56
- лимонад NOUN 0.53
- напиток NOUN 0.53



Частотность слова

- Высокая Средняя Низкая

НКРЯ и Wikipedia

- преданность NOUN 0.38
- доброта NOUN 0.37
- нежность NOUN 0.37
- упование NOUN 0.35
- умиление NOUN 0.35

<https://rusvectores.org/ru/calculator>

Свойства word2vec

Результат обучения векторных представлений сильно зависит от коллекции документов. **Могут возникать неожиданные артефакты.**

Википедия

`most_similar(россия)`

российский 0.5653642416

рф 0.523694574833

украина 0.492026507854

ссср 0.473026573658

финляндия 0.464367419481

`most_similar(тролль)`

муметь 0.717674195766

гоблин 0.559770524502

великан 0.557757973671

злобный 0.55741250515

гном 0.554968833923

Луркоморье

`most_similar(россия)`

беларусь 0.645048737526

европа 0.622894406319

украина 0.622316598892

рашка 0.619276404381

германия 0.609378278255

`most_similar(тролль)`

троллинг 0.725703835487

троль 0.660580933094

лжец 0.582996308804

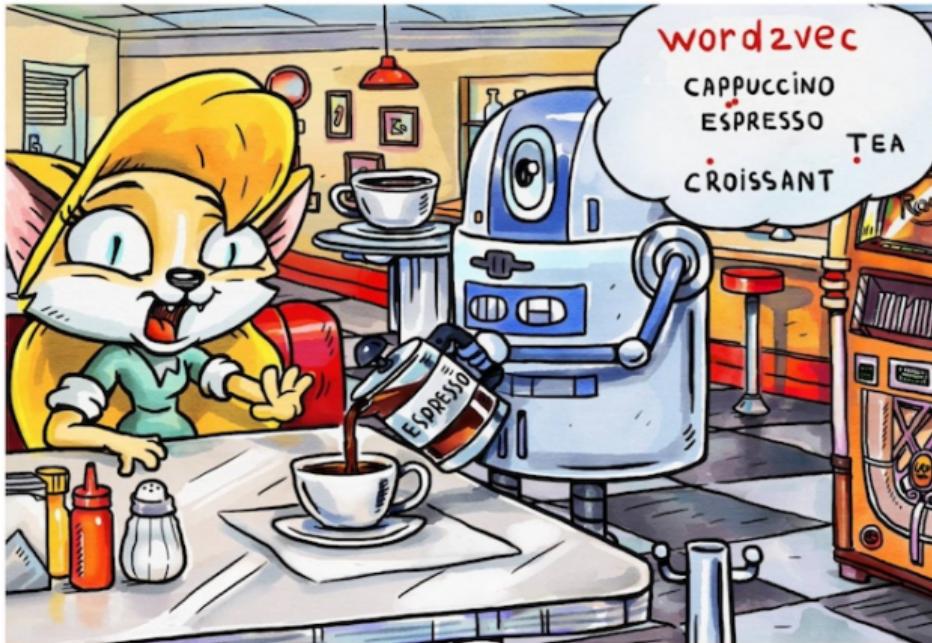
проводокатор 0.57004237175

толстый 0.568691492081

Модель - сексист

```
model.most_similar(u'интеллектуал')
[('моралист', 0.7139864563941956),
('теоретик', 0.6941959857940674),
('литератор', 0.6819325089454651)]  
  
model.most_similar(u'интеллектуалка')
[('бездельница', 0.6617184281349182),
('бунтарка', 0.6578608751296997),
('дилетантка', 0.6419748663902283)]  
  
  
model.most_similar(positive=[u'интеллектуал', u'женщина'], negative=[u'мужчина'])
[('знаменитость', 0.49774518609046936),
('поп-звезда', 0.4860984981060028),
('элита', 0.48200151324272156)]  
  
model.most_similar(positive=[u'ум', u'женщина'], negative=[u'мужчина'])
[('любовь', 0.43659064173698425),
('душа', 0.4330841302871704),
('внешность', 0.4280041456222534)]  
  
model.most_similar(positive=[u'гений', u'женщина'], negative=[u'мужчина'])
[('букашка', 0.4793989062309265),
('химера', 0.4589369595050812),
('душонка', 0.4547439217567444)]
```

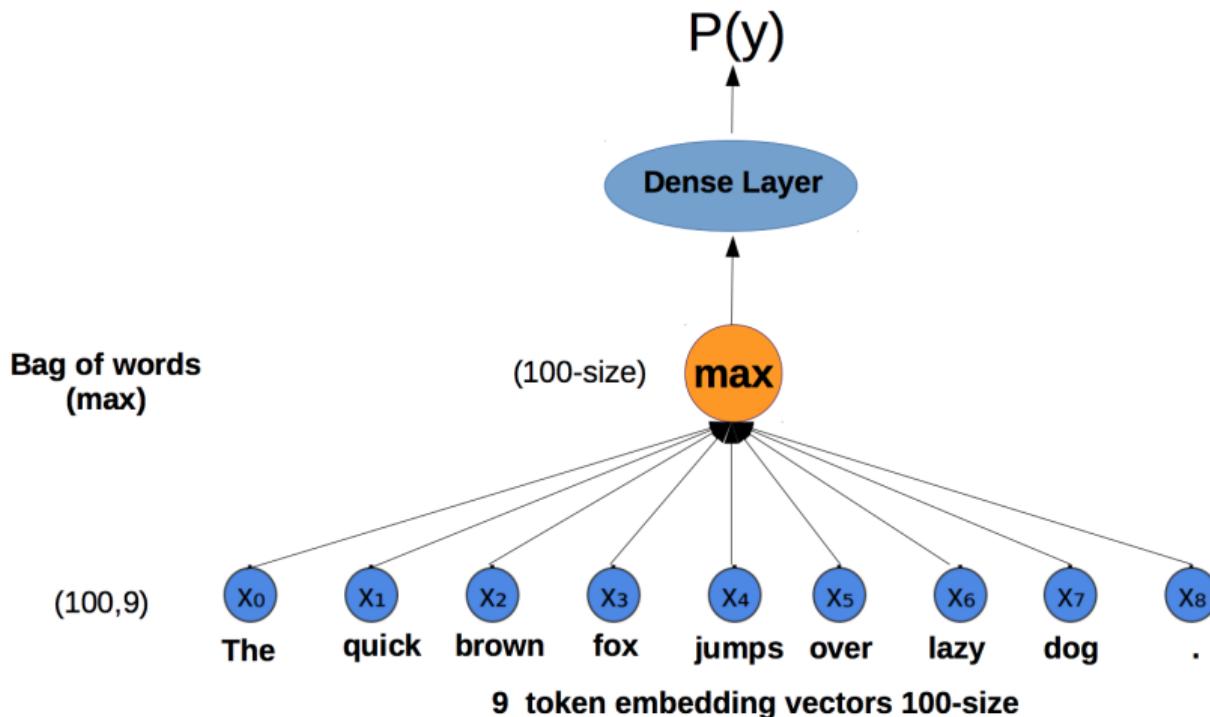
Word2vec всего лишь модель



- Espresso? But I ordered a cappuccino!
- Don't worry, the cosine distance between them is so small that they are almost the same thing.

Свёрточные нейросети для текстов

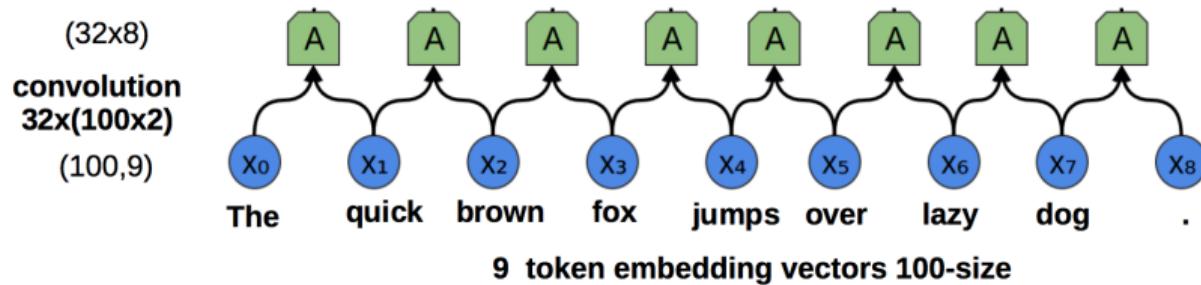
Нейросеть для текстов



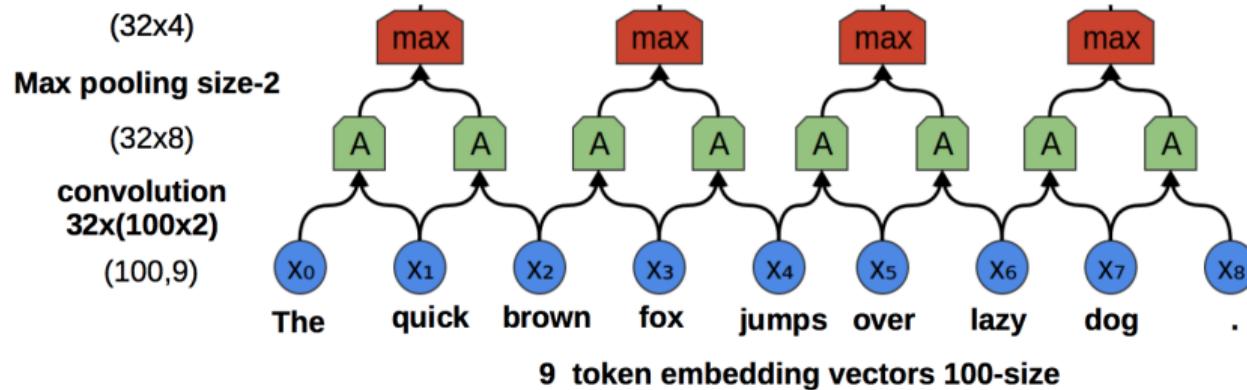
Проблемы

- Теряем информацию о порядке слов
- Решить эту проблему можно обучив эмбединги для биграм, трigram и тд, но это расширит пространство признаков
- Можно решить эту проблему с помощью рекурентных нейросеток, о них мы будем говорить в следующий раз
- Можно решить эту проблему с помощью свёрточного слоя

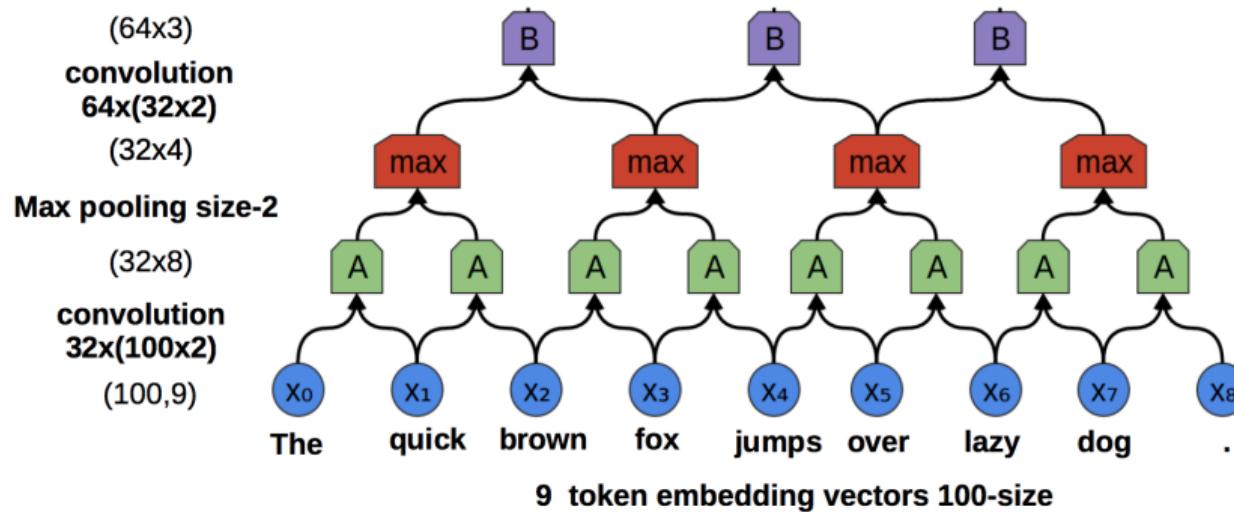
Свёрточная сеть



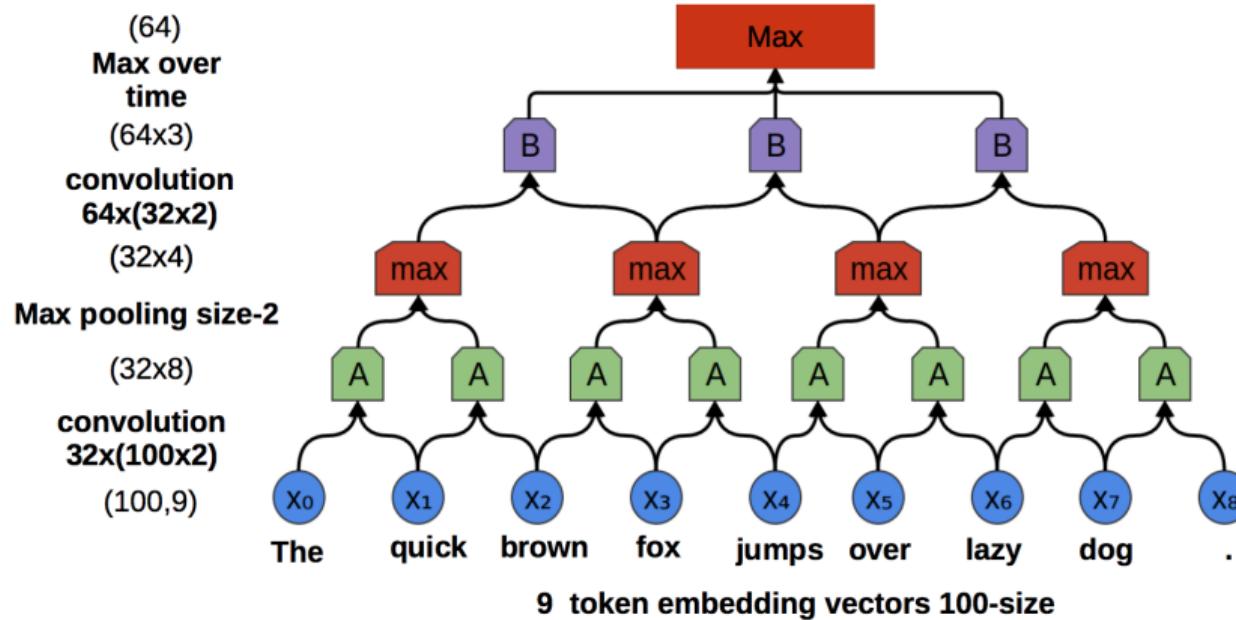
Свёрточная сеть



Свёрточная сеть



Свёрточная сеть



Свёрточная сеть

