

Глубокое обучение и вообще

Ульянкин Филипп

25 мая 2022 г.

Посиделка 11: Идентификация объектов и обучение без учителя

Agenda

- Что мы уже знаем про обучение представлений
- Автокодировщики для картинок
- Автокодировщики для графов
- Автокодировщики для текстов

Что мы уже знаем про обучение
представлений (тексты)

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 отранжировал сериалы и тп

Свойства word2vec



Частотность слова

- Высокая Средняя Низкая

НКРЯ и Wikipedia

1. чай noun 0.56
2. пиво noun 0.56
3. самогон noun 0.56
4. лимонад noun 0.53
5. напиток noun 0.53



Частотность слова

- Высокая Средняя Низкая

НКРЯ и Wikipedia

1. преданность noun 0.38
2. доброта noun 0.37
3. нежность noun 0.37
4. упование noun 0.35
5. умиление noun 0.35

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

Свойства word2vec

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

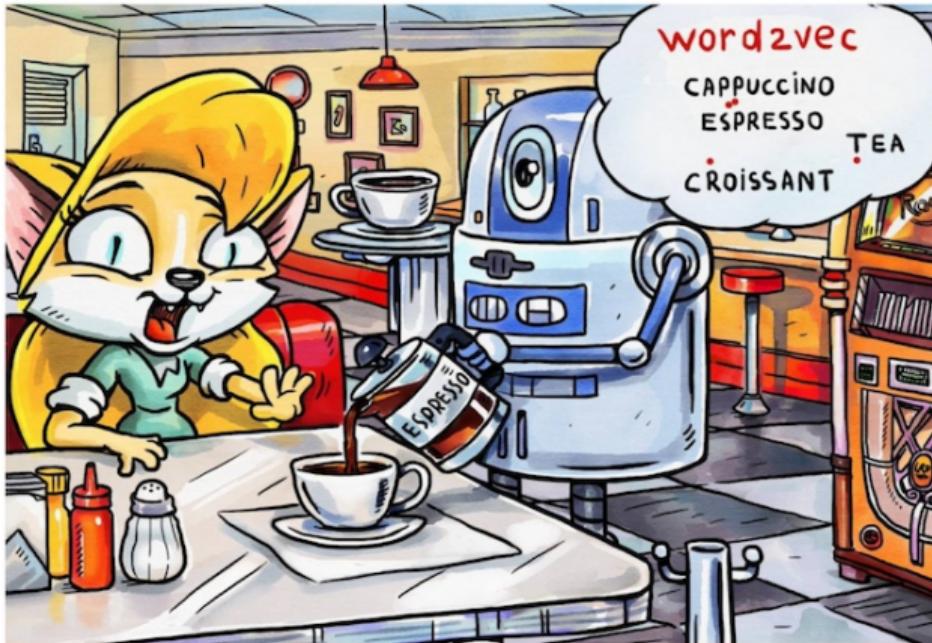
Википедия

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

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.

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

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 обобщить результат до текста

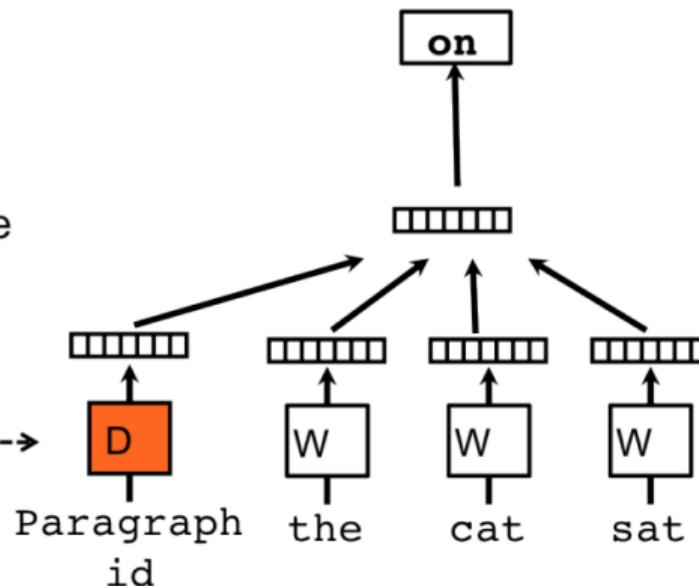
- Усреднить
- Засунуть матрицу в свёрточную сетку или RNN
- doc2vec

doc2vec (Distributed Memory)

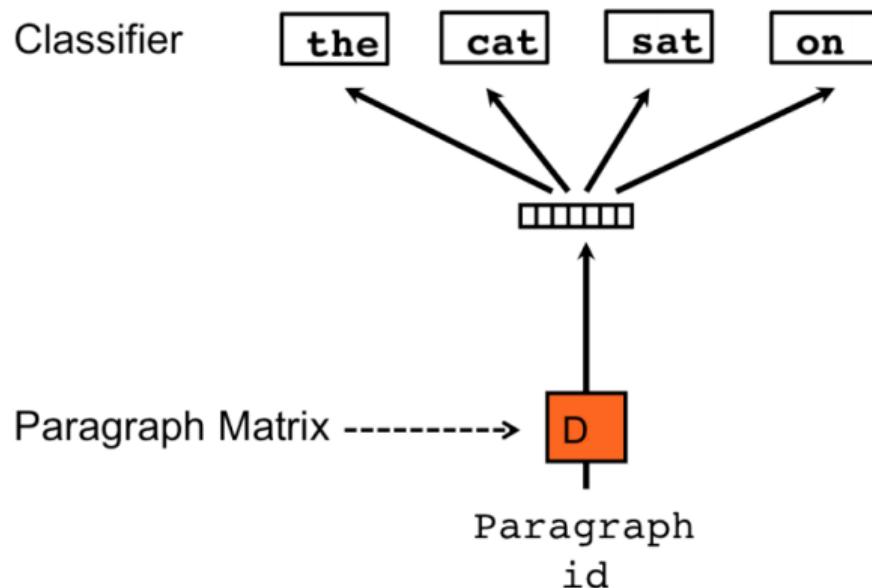
Classifier

Average/Concatenate

Paragraph Matrix----->



doc2vec (Distributed Bag Of Words)



<https://arxiv.org/abs/1405.4053>

doc2vec

- Инициализируем случайно дополнительный вектор для каждого параграфа или текста
- **Distributed Memory:** Предсказываем слово, используя вектор параграфа и контекст
- **Distributed Bag Of Words:** предсказываем пропущенные слова по вектору документа
- По-прежнему мешок слов, порядок не учитывается, новые слова и тексты не обрабатываются

<https://arxiv.org/abs/1405.4053>

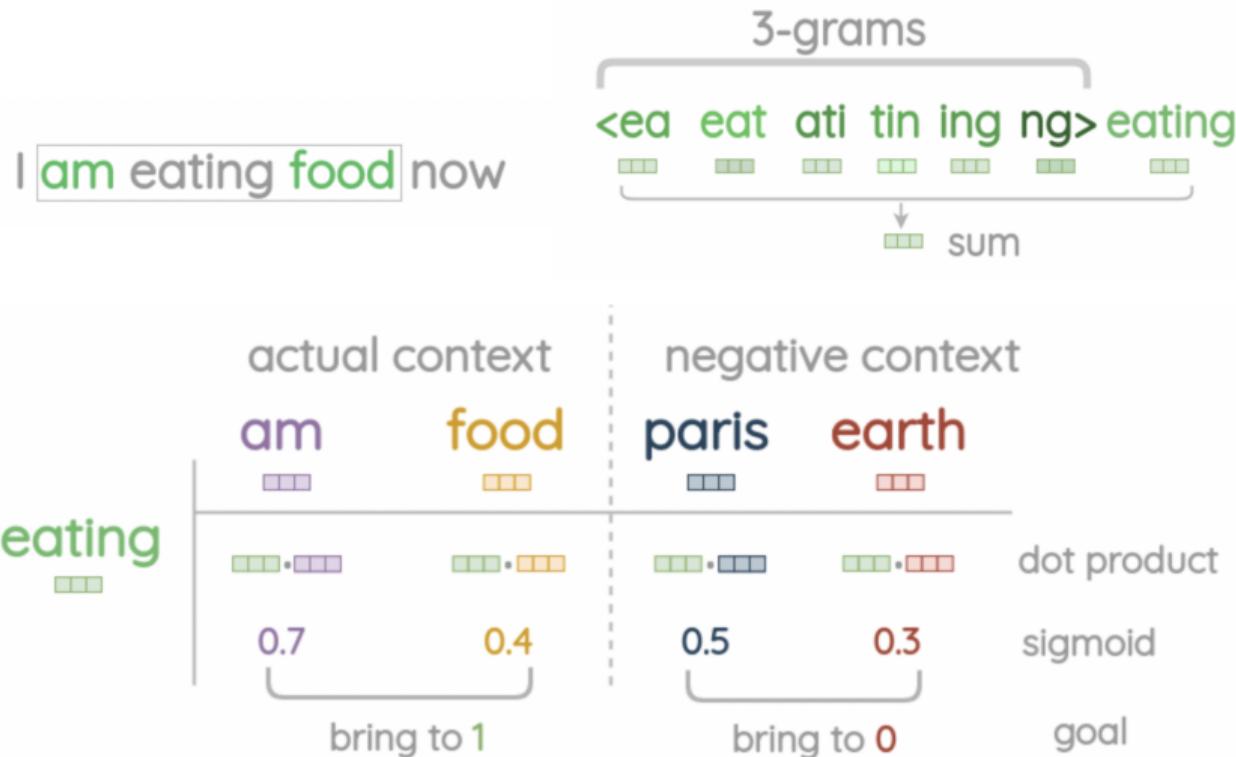
Fasttext

- Усовершенствование word2vec дополнительными токенами, отвечающими за буквенные n-граммы, это решает проблему OOV (out of the vocabulary)
- Например, триграмы для eating: <ea, eat, ati, tin, ing ng>
- Символы < и > это спец-символы, говорящие о том, что слово кончилось или началось
- При расчёте для слова итогового вектора будем усреднять эмбединг для слова и эмбединги для n-грамм

<https://arxiv.org/pdf/1607.04606v1.pdf>
<https://arxiv.org/pdf/1607.01759.pdf>

<https://fasttext.cc/>
<https://fasttext.cc/docs/en/crawl-vectors.html>

Fasttext

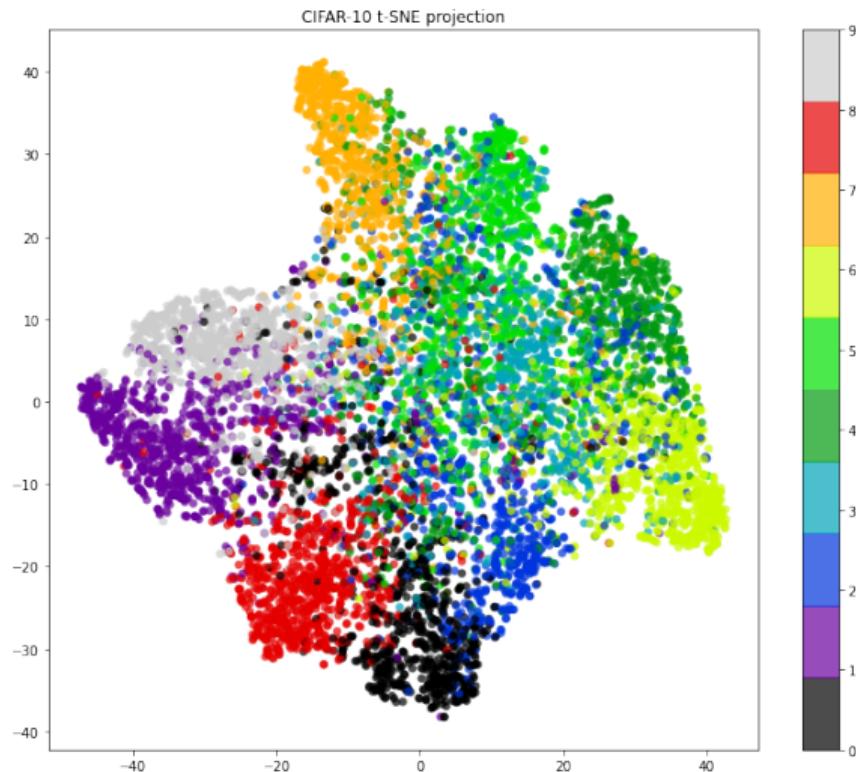


Что мы уже знаем про обучение
представлений (картинки)

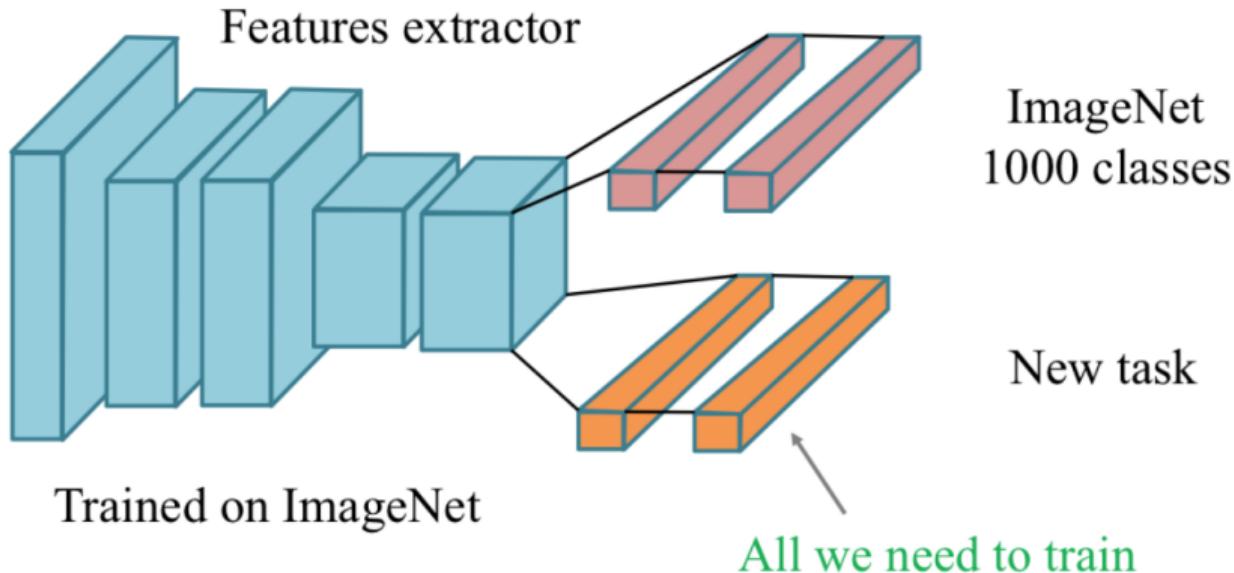
Представления с последних слоёв

- Выходы с последних слоёв свёрточных нейросетей являются хорошими признаковыми описаниями изображений
- Такие векторные представления называют **эмбеддингами (embeddings)**
- У эмбеддингов нет чёткой интерпретации, цифры в них говорят о наличии каких-то паттернов, на которые настроилась нейросетка
- Эмбеддинги картинок оказываются полезными во многих задачах

TSNE для CIFAR-10 (предпоследний слой простой свёрточной сетки)



Представления изображений



Дообучение (Transfer learning)

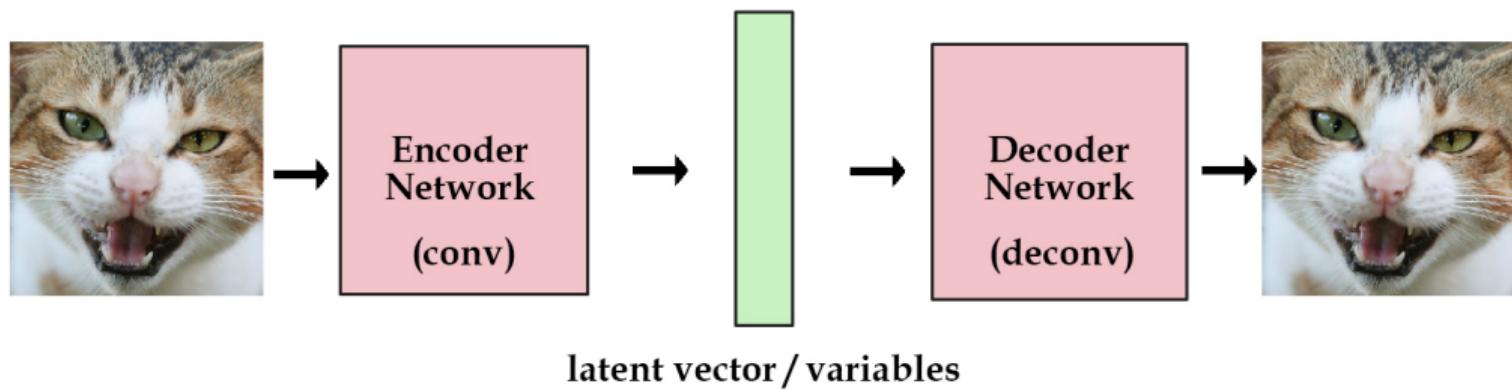
- Если данных совсем мало
- Берём модель из другой задачи
- Заменяем последний слой на слой с нужным числом выходов
- Обучаем только его
- По сути, это обучение линейной модели на эмбединге картинки
- Иногда приходится дообучать часть экстрактора эмбедингов

Представления изображений

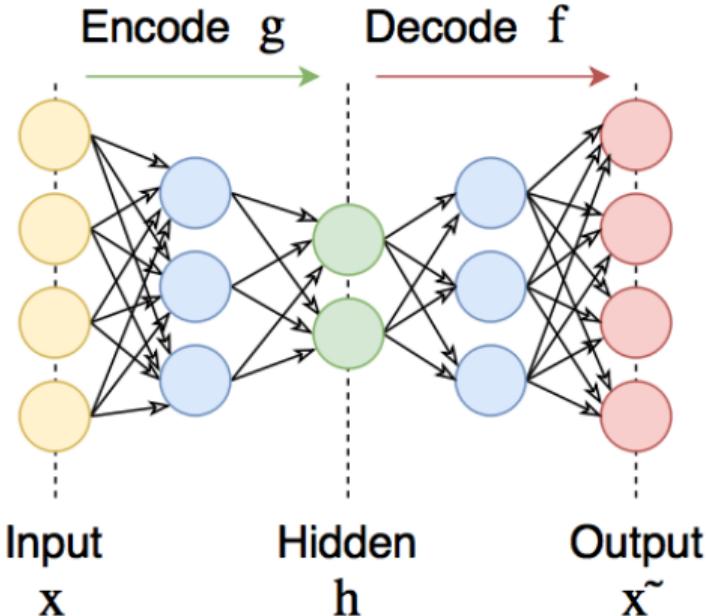
- Выход предпоследнего полносвязного слоя — хорошее представления картинки
- Но для его обучения нужны изображения с разметкой
- Может, получится строить такие представления и без разметки?
- Ведь для текстов получилось, мы собрали w2v
- Нужна какая-то фейковая задача

Автокодировщики для картинок

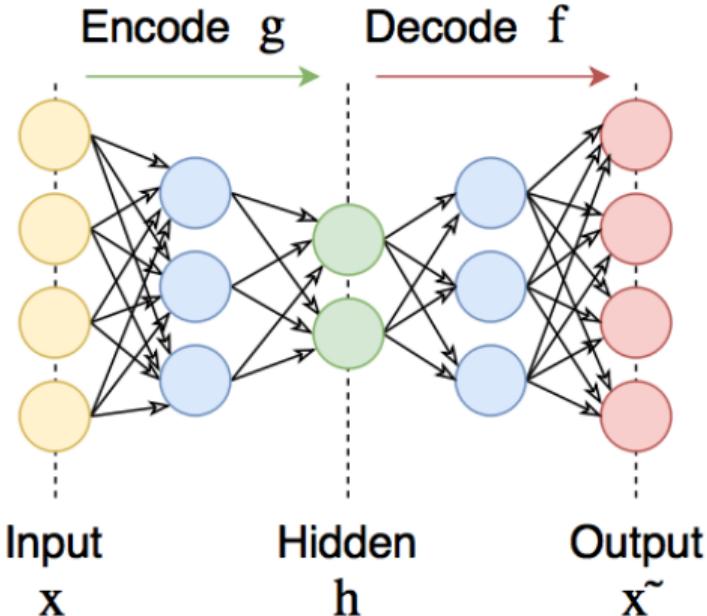
Автокодировщики



Автокодировщики

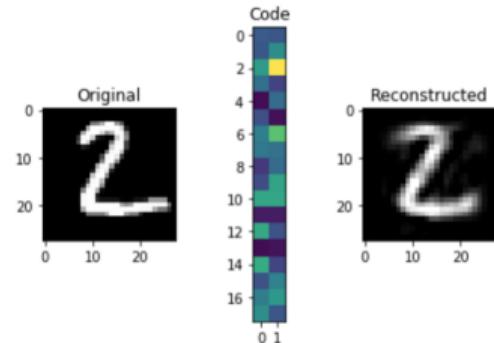
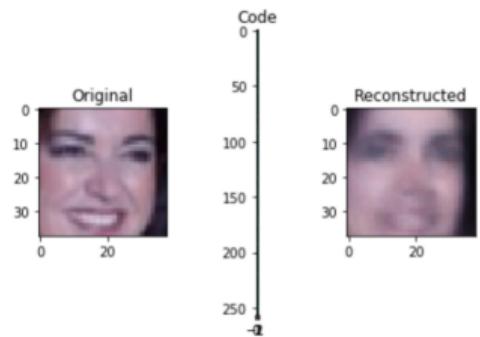
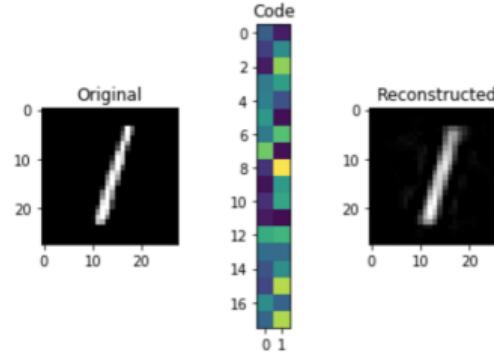
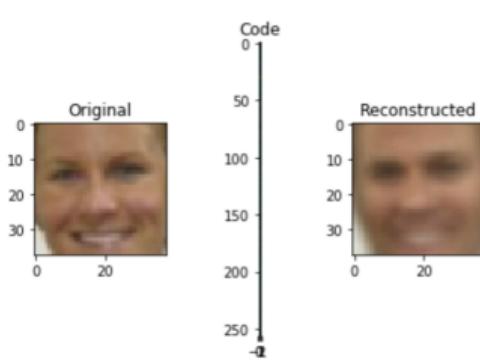


Автокодировщики



$$\frac{1}{n} \sum_{i=1}^n L(x_i, f(g(x_i))) \rightarrow \min$$

Пример сжатия



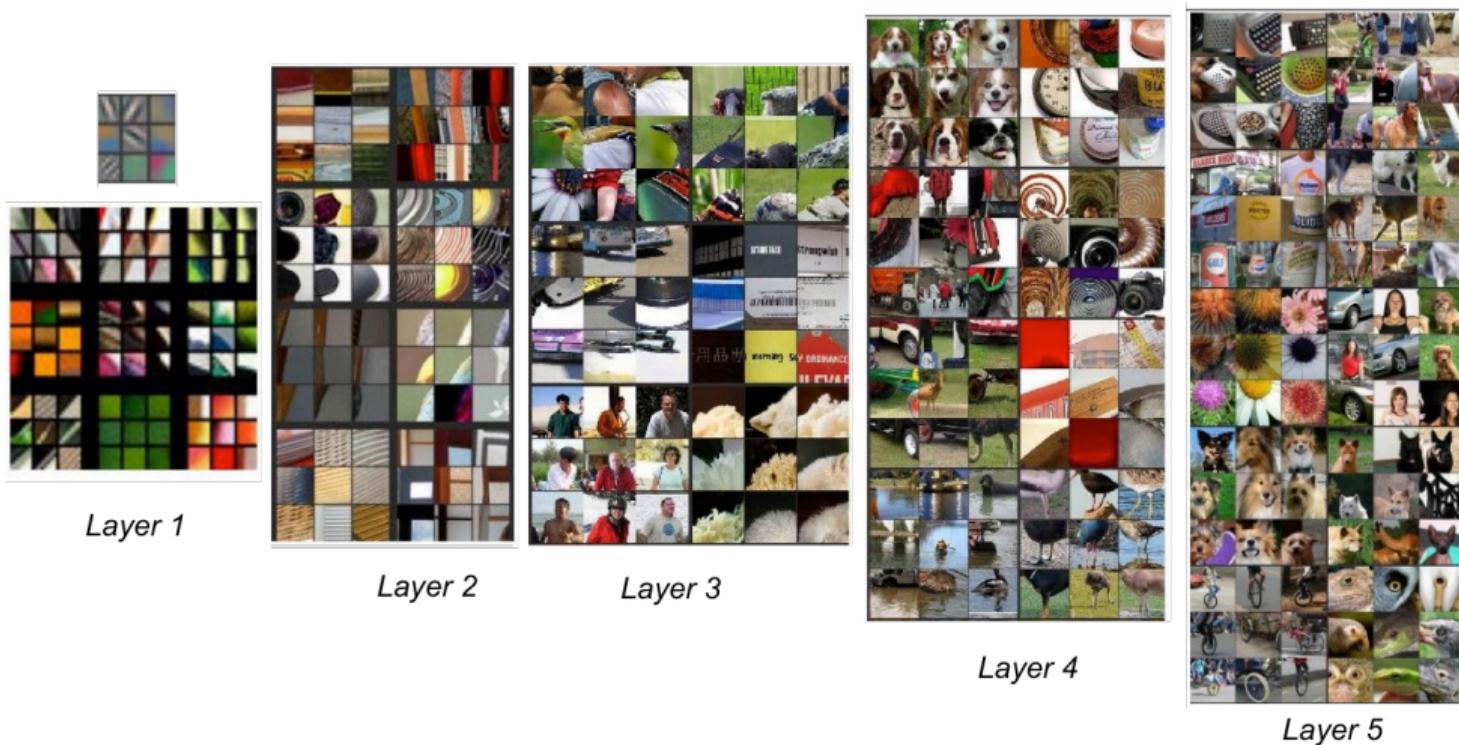
Зачем это всё?

- Сжатие данных (нелинейных аналог РСА)
- Очистка картинки от шума
- Поиск похожих изображений
- Трансформация изображений
- Генерация изображений

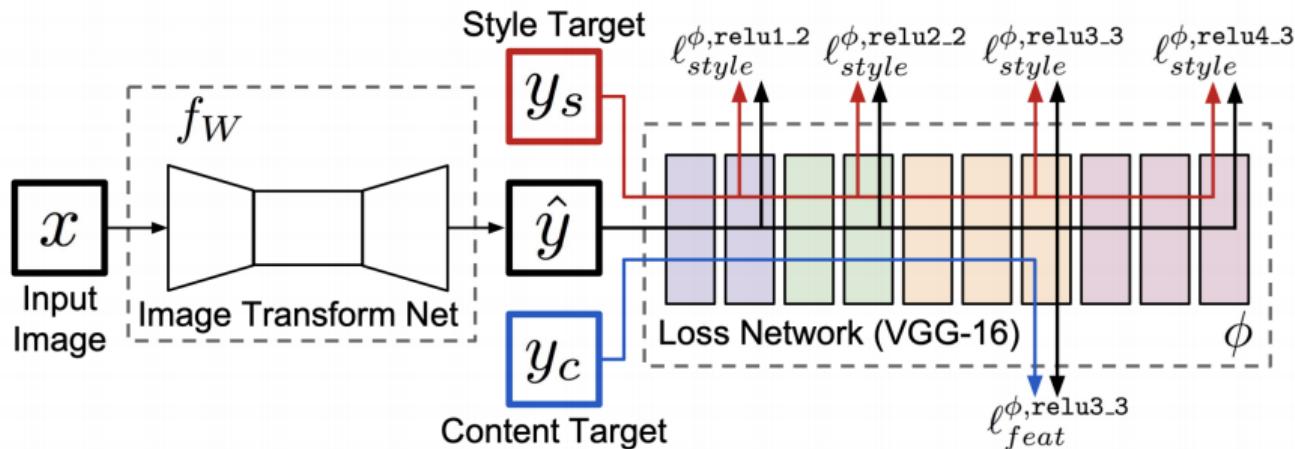
Автокодировщики

- Понижают размерность, исходная картинка восстанавливается с потерями
- Основная ценность — эмбединг из булочного горлышка, хочется чтобы он получился максимально качественным
- Наша архитектура сильно переобучается под выборку, для новых лиц работает хуже
- Нужна регуляризация, которая позволит извлекать смысл, а не восстанавливать пиксели в точности (фон тоже запоминается)

Что находит нейросеть... опять ...



Perceptual loss



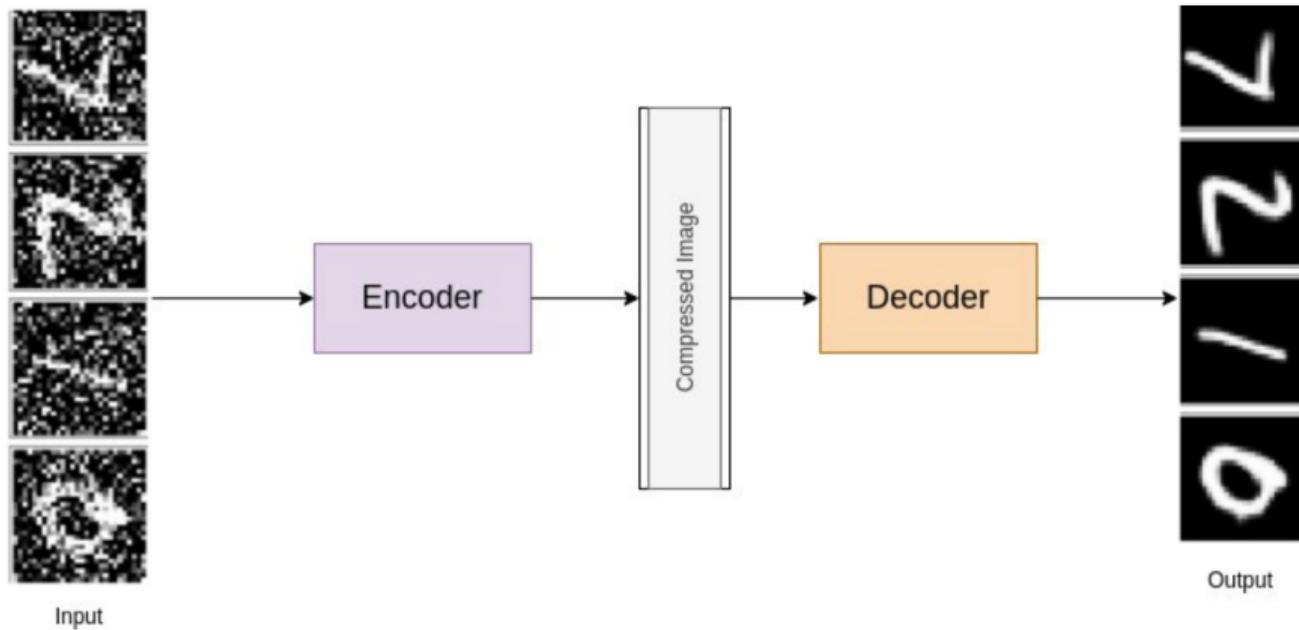
<https://arxiv.org/pdf/1603.08155.pdf>

Perceptual loss

- Чем глубже слой, тем осмысленнее образы
- Ранние слои во многом определяют стиль картинки
- Можно измерять расстояния между выходами свёрточных слоёв картинок
- Это поможет сохранить смысл и сделать эмбединг более осмысленным

<https://arxiv.org/pdf/1603.08155.pdf>

Denoising autoencoder

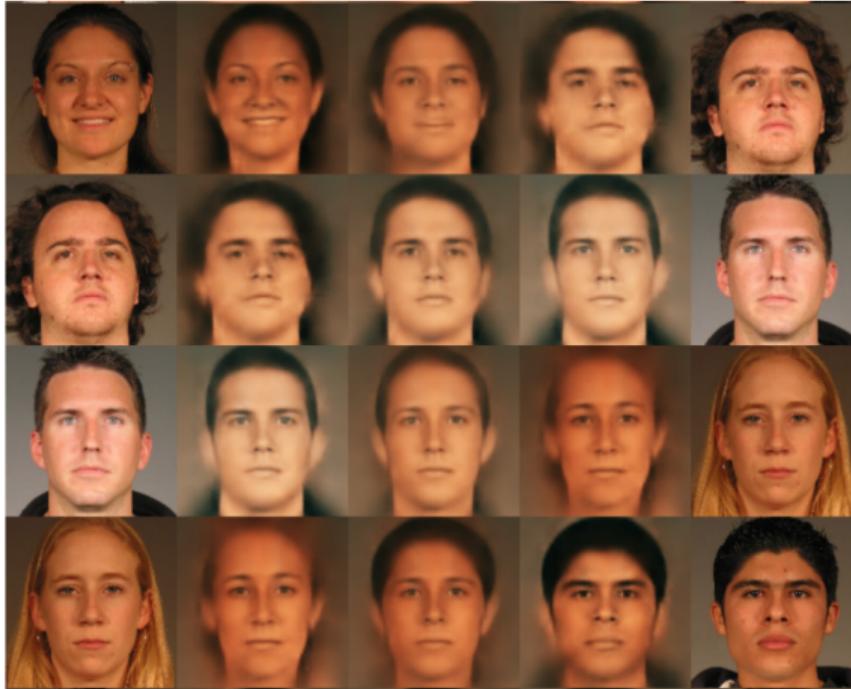


<https://medium.com/@garimanishad/>

reconstruct-corrupted-data-using-denoising-autoencoder-python-code-aeaff4b0958e

<https://www.cs.toronto.edu/~larocheh/publications/icml-2008-denoising-autoencoders.pdf>

Morphing faces



http://essay.utwente.nl/81372/1/Heuver_MA_EEMCS.pdf

Morphing faces

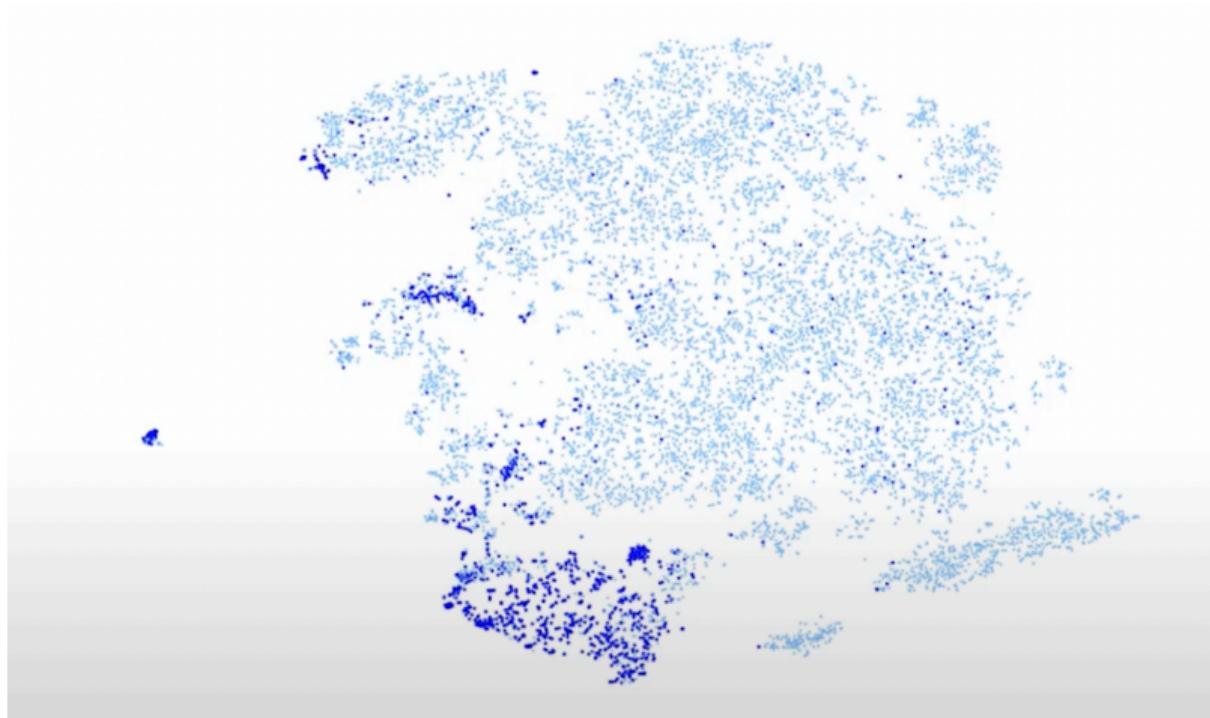


<https://www.echevarria.io/blog/lego-face-vae/index.html>

Саморегуляция выборки



Поиск фрода



Автокодировщики в антифроде Яндекса: <https://www.youtube.com/watch?v=aBckDgtG0Zs>

Обучение представлений для графов

Задачи на графах

- **Node-focused tasks:** изучаем структурные свойства вершины графа относительно других вершин
- классификация вершин (поиск хабов в транспортных сетях), предсказание связей (предсказания взаимодействия между белками), рекомендация вершин (поиск возможных друзей в социальных сетях)
- **Graph-focused tasks:** изучаем граф целиком или подграф
- классификация графов (предсказание токсичности молекулы), генерация графов (генерация молекул белков с желаемыми свойствами)

<https://github.com/esokolov/ml-course-hse/blob/master/2021-spring/seminars/sem17-graph.pdf>

Представления для вершин

- Обучаем векторное представление вершин графа, хотим чтобы вектор включал в себя информацию о структурной роли вершина графа
- Граф: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- Множество вершин: $\mathcal{V} = \{v_i\}_{i=1}^{|\mathcal{V}|}$
- Ребра с неотрицательными весами: $\mathcal{E} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}_{\geq 0}$
- Функция похожести между вершинами: $s_{\mathcal{G}} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}$

Примеры функции похожести между вершинами

- Если вершины соединены ребром $s_{\mathcal{G}}(v_i, v_j) = 1$ и $s_{\mathcal{G}}(v_i, v_j) = 0$ иначе
- Формула, зависящая от расстояния между вершинами

$$s_{\mathcal{G}}(v_i, v_j) = \exp\left(-\frac{d_{\mathcal{G}}(v_i, v_j)}{\tau}\right),$$

где $d_{\mathcal{G}}(v_i, v_j)$ — расстояние между вершинами v_i и v_j , а τ — гиперпараметр.

- Оценить вероятность перехода из одной вершины в другую случайным блужданием по графу

Энкодер и декодер для графов

- Энкодер преобразует данные в вектор размерности d :

$$\text{ENC} : \mathcal{V} \rightarrow \mathbb{R}^d, \text{ENC}(v_i) = z_i$$

- Декодер принимает два векторных представления и пытается приблизить функцию похожести соответствующих вершин:

$$\text{DEC} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}, \text{DEC}(z_i, z_j) \approx s_{\mathcal{G}}(v_i, v_j)$$

- Для обучения берём удобную функцию потерь и минимизируем по параметрам декодера и энкодера:

$$\sum_{v_i, v_j \in \mathcal{V}} \mathcal{L}\left(\text{DEC}\left(\text{ENC}(v_i), \text{ENC}(v_j)\right), s_{\mathcal{G}}(v_i, v_j)\right) \rightarrow \min_{\text{ENC}, \text{DEC}}$$

Пример:

- В качестве энкодера в простейшем случае можем взять матрицу $|\mathcal{V}| \times d$
- В роли декодера можно использовать обычное скалярное произведение $\text{DEC}(z_i, z_j) = z_i^T z_j$

node2vec

- В качестве функции похожести используем вероятность случайного блуждания попсть из вершины v_i в вершину v_j , $p(v_j|v_i)$
- Будем обучать вектора так, чтобы

$$p(v_j|v_i) \approx \frac{e^{z_j^T z_i}}{\sum_k e^{z_k^T z_i}} = \text{DEC}(z_i, z_j)$$

- Энкодер тут матрица, а декодер Softmax от скалярных произведений

node2vec

- В качестве функции похожести используем вероятность случайного блуждания попсть из вершины v_i в вершину v_j , $p(v_j|v_i)$
- Будем обучать вектора так, чтобы

$$p(v_j|v_i) \approx \frac{e^{z_j^T z_i}}{\sum_k e^{z_k^T z_i}} = \text{DEC}(z_i, z_j)$$

- Энкодер тут матрица, а декодер Softmax от скалярных произведений
- Для оценки вероятностей будем запускать по графу случайные блуждания из разных вершин какой-то длины

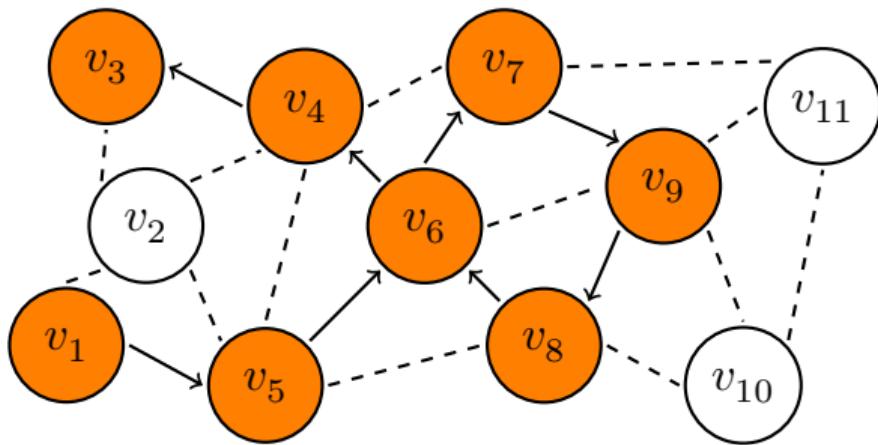
node2vec

- В качестве функции похожести используем вероятность случайного блуждания попсть из вершины v_i в вершину v_j , $p(v_j|v_i)$
- Будем обучать вектора так, чтобы

$$p(v_j|v_i) \approx \frac{e^{z_j^T z_i}}{\sum_k e^{z_k^T z_i}} = \text{DEC}(z_i, z_j)$$

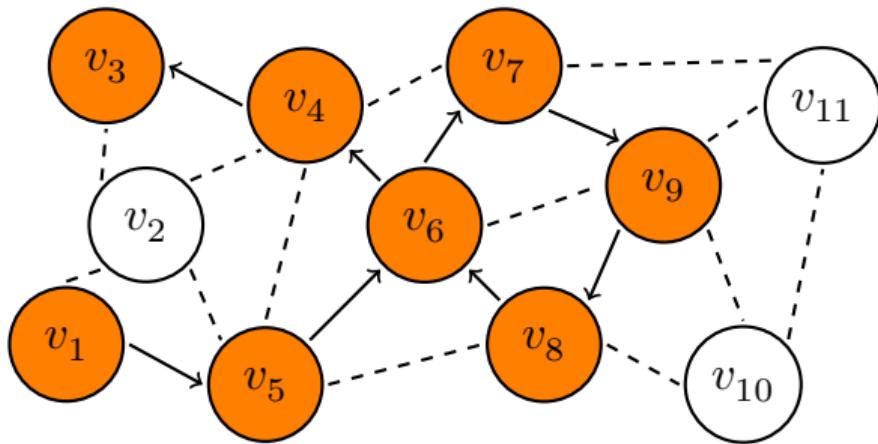
- Энкодер тут матрица, а декодер Softmax от скалярных произведений
- Для оценки вероятностей будем запускать по графу случайные блуждания из разных вершин какой-то длины
- Узнали word2vec ??? Фактически это skip-gram word2vec, где словами являются вершины графа, а предложениями случайные пути в графе

node2vec



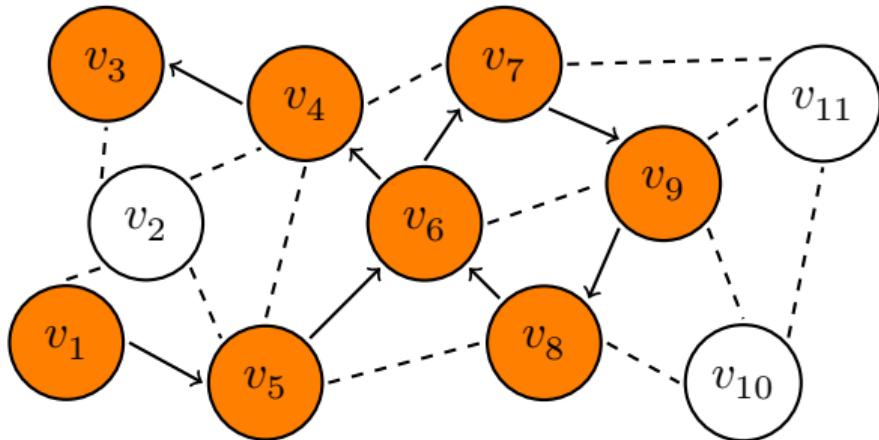
- Случайный путь: $S = [v_1, v_5, v_6, v_7, v_9, v_8, v_6, v_4, v_3]$
 - Пусть размер контекста равен 2
 - Для v_1 получаем $[v_1, v_5, v_6]$
 - Для первого вхождения v_6 получим $[v_1, v_5, v_6, v_7, v_9]$

node2vec



- Для каждого окна выписываем пары (центр окна, другое упоминание вершины в окне)
- Для $[v_1, v_5, \underline{v_6}, v_7, v_9]$ получаем пары $(v_6, v_1), (v_6, v_5), (v_6, v_7), (v_6, v_9)$.

node2vec



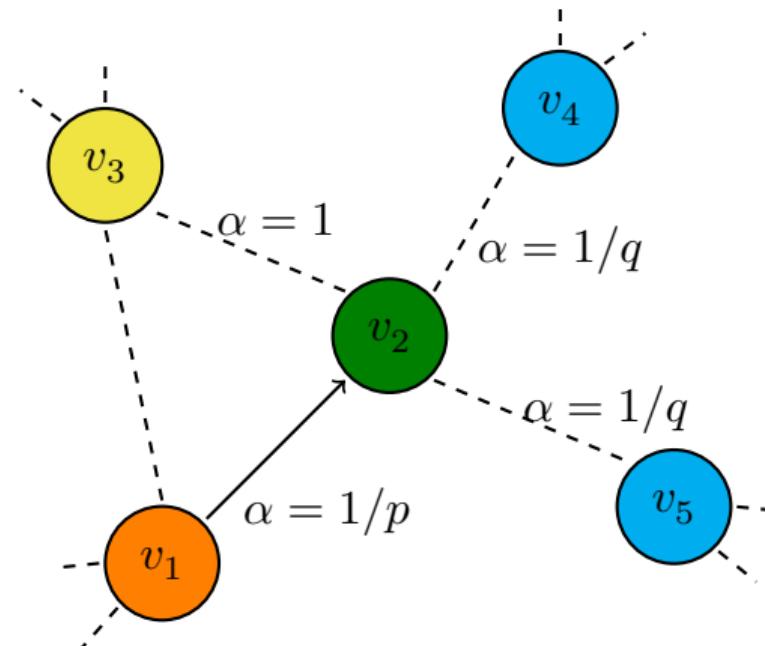
- Минимизируем функционал:

$$\mathcal{L} = \sum_{(v_i, v_j)} -\log (\text{DEC}(z_i, z_j)) = \sum_{(v_i, v_j)} \left(-z_j^T z_i + \log \sum_k e^{z_k^T z_i} \right) \rightarrow \min_z$$

- Вместо софтмакса обычно делают негативное сэмплирование

node2vec

- Итоговые векторы будут содержать информацию о близости вершин с точки зрения случайного блуждания, его можно параметризовать



node2vec

Допустим, на очередном шаге блуждания мы попали из вершины v_1 в вершину v_2 . Чтобы определить вероятности перехода в соседние вершины, зададим веса ребрам α :

- Для ребра, ведущего обратно, положим $\alpha = 1/p$.
- Для общих соседей предыдущей и текущей вершины положим $\alpha = 1$.
- Для всех остальных вершин положим $\alpha = 1/q$.
- Что контролирует p ? Что контролирует q ?

Особенности node2vec

- Непонятно что делать с новыми вершинами, появляющимися в графе
- Никак не учитываем мета-информацию о вершинах
- Для вершин нет общих параметров, можем переобучится

Neighborhood autoencoder

- Поставим в соответствие каждой вершине вектор из её похожестей на остальные (вектор будет очень длинный)

$$v_i \mapsto s_i, s_{ij} = s_{\mathcal{G}}(v_i, v_j)$$

- Сжимаем с помощью автокодировщика

$$z_i = \text{ENC}(s_i), \text{DEC}(z_i) \approx s_i$$

- Обучать можно, например, с помощью MSE:

$$\sum_i \left\| \text{DEC}(\text{ENC}(s_i)) - s_i \right\|^2 \rightarrow \min_{\text{ENC}, \text{DEC}}$$

Особенности Neighborhood autoencoder

- Параметры модели будут общими для всех вершин
- Всё ещё не умеем генерировать эмбеддинги для новых вершин
- Всё ещё не используем метаданные

Neighborhood aggregation

- Для каждой вершины будем агрегировать признаки её соседей, в том числе метаданные (можно их усреднять)
- Соединяем вектор агрегированных признаков с признаками исходной вершины
- Обучаем на них автокодировщик

Представления для графов

- Усреднить эмбеддинги вершин:

$$z_{\mathcal{G}} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} z_v$$

- Обрабатывать последовательность из эмбеддингов вершин рекуррентными нейросетями
- Изучить более продвинутые методы :)

ELMO

Embeddings from Language Models (ELMo)

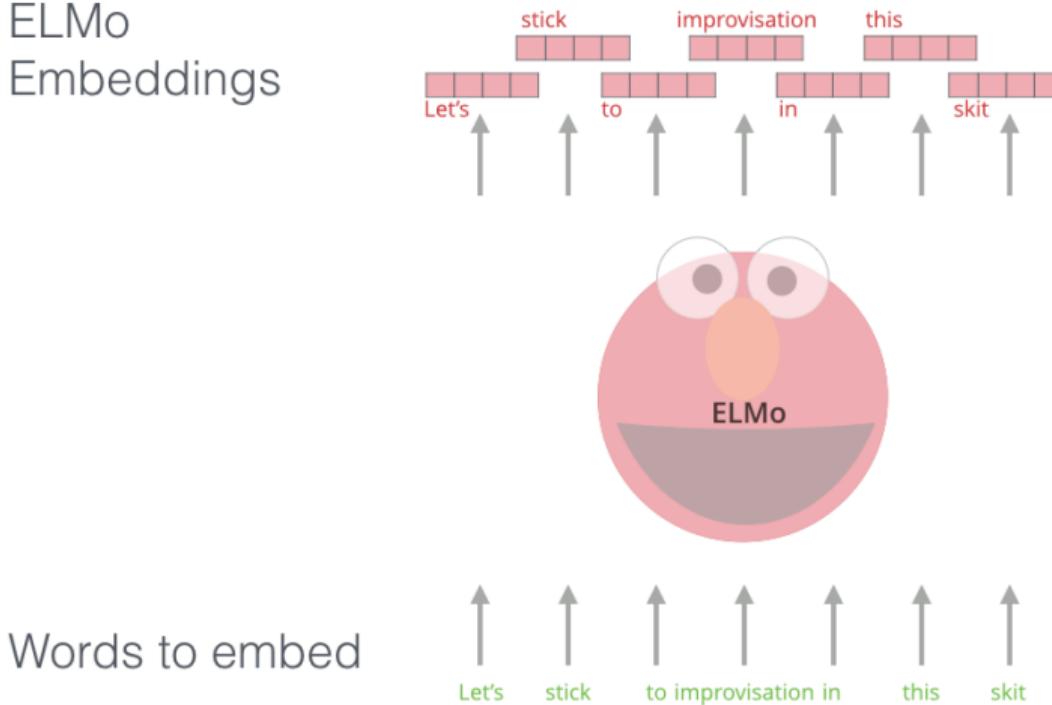
- Усложним наши модели для текстов: начнём учитывать порядок слов
- Для этого соберём рекуррентный автокодировщик

Embeddings from Language Models (ELMo)

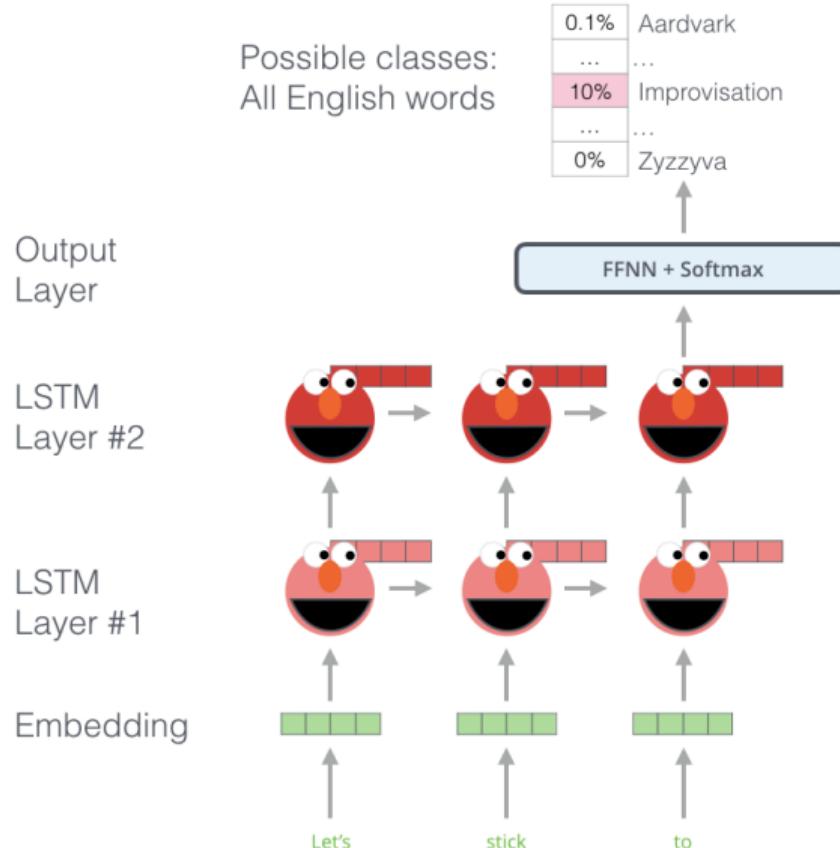


Embeddings from Language Models (ELMo)

ELMo
Embeddings

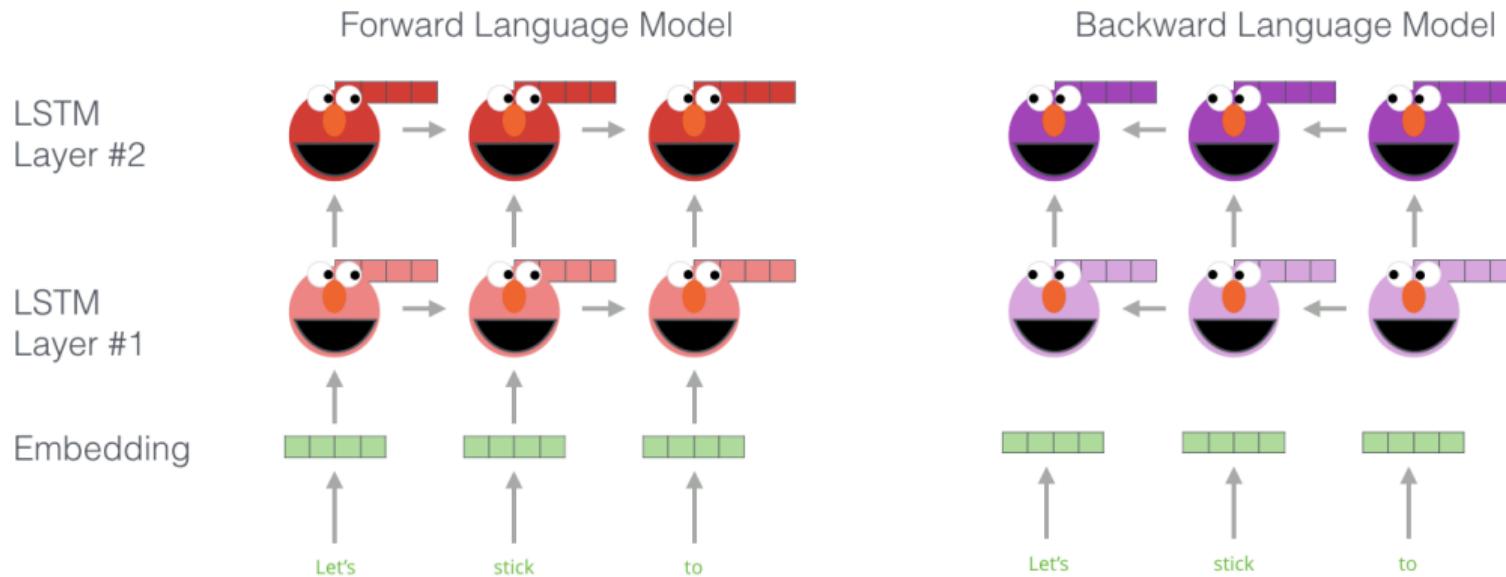


Embeddings from Language Models (ELMo)



Embeddings from Language Models (ELMo)

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



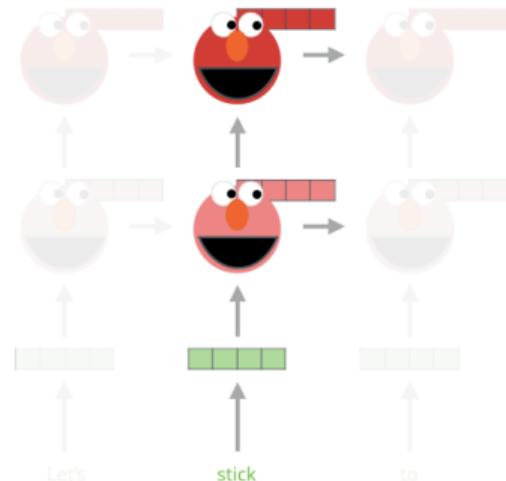
Embeddings from Language Models (ELMo)

Embedding of “stick” in “Let’s stick to” - Step #2

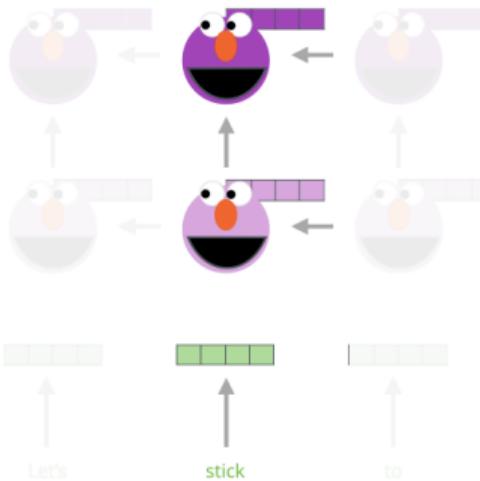
1- Concatenate hidden layers



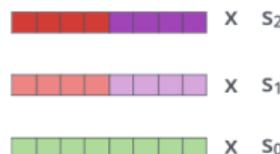
Forward Language Model



Backward Language Model



2- Multiply each vector by a weight based on the task

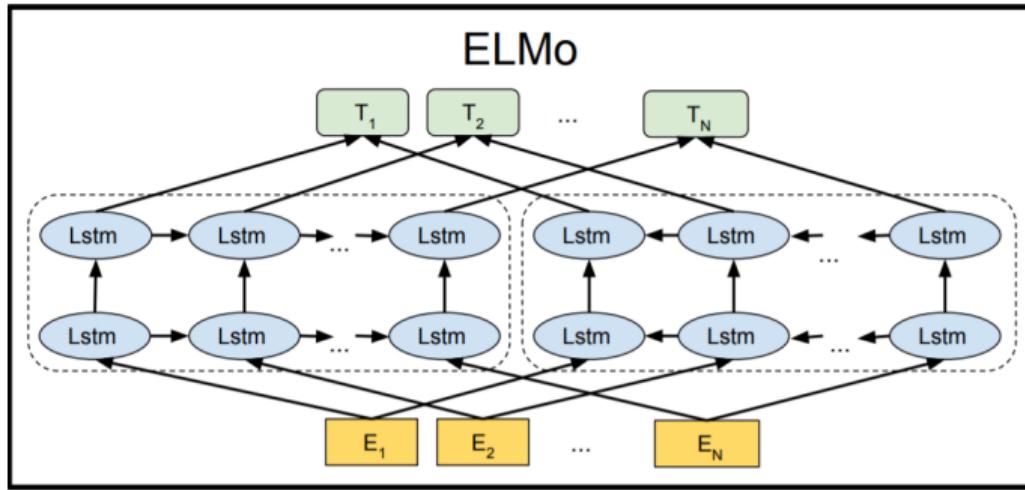


3- Sum the (now weighted) vectors



ELMo embedding of “stick” for this task in this context

Embeddings from Language Models (ELMo)



В качестве эмбеддинга используется вектор $[T, h_l, h_r]$, где T - токены, которые сетка выплёвывает наружу, h_l - итоговое скрытое состояние ячеек при проходе слева направо, h_r - справа налево.

<https://arxiv.org/pdf/1802.05365.pdf>

DSSM

Рекомендательные системы

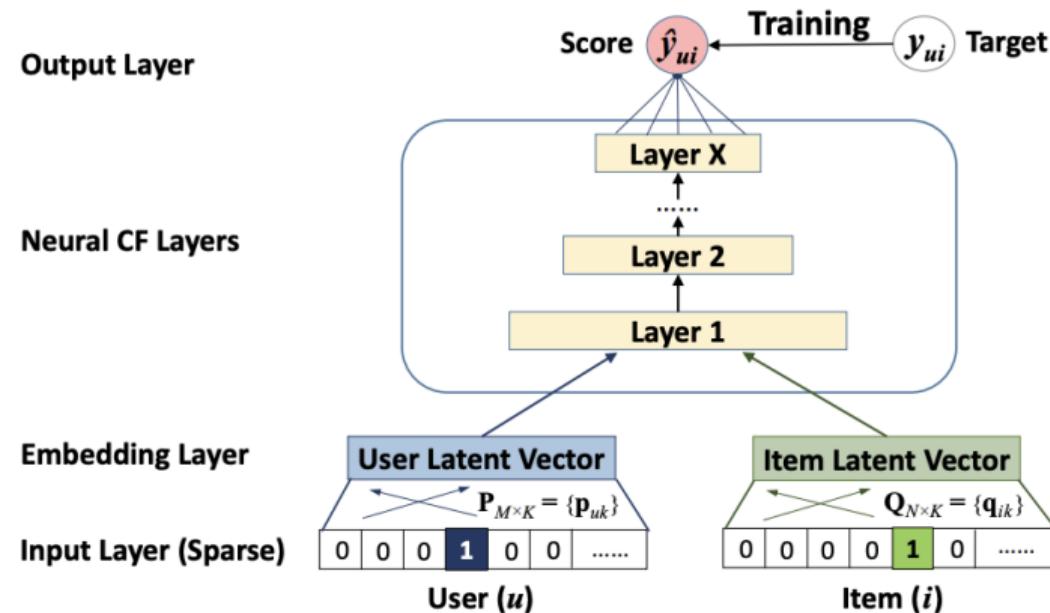


Figure 2: Neural collaborative filtering framework