



Разговорный ИИ

Машинное обучение в диалоговых системах

Лекция 6

12 марта 2019

Кому интересно

- [From Eliza to Xiaolce: Challenges and Opportunities with Social Chatbots](#)
- [The Design and Implementation of Xiaolce, an Empathetic Social Chatbot](#)
- [Smart Reply: Automated Response Suggestion for Email](#)
- [Conversational AI: The Science Behind the Alexa Prize](#)
- [How NOT To Evaluate Your Dialogue System](#)
- [A Deep Reinforcement Learning Chatbot](#)
- [Алиса: что внутри](#)

Disclaimer: половину этого мы сами ещё не сделали

1966 — Eliza

```
EEEEEE LL      IIII ZZZZZZZ  AAAAA
EE      LL      II      ZZ  AA  AA
EEEEEE LL      II      ZZZ  AAAAAA
EE      LL      II      ZZ  AA  AA
EEEEEE LLLLLL IIII ZZZZZZZ  AA  AA
```

Eliza is a mock Rogerian psychotherapist.
The original program was described by Joseph Weizenbaum in 1966.
This implementation by Norbert Landsteiner 2005.

```
ELIZA: Is something troubling you ?
YOU:   Yes. I'm trying to describe you for a research paper I'm writing. What sh
ould I say?
ELIZA: You are sure.
YOU:   Yes, I'm sure.
ELIZA: I see.
YOU:   What can I tell them about you?
ELIZA: You seem to need to tell them about me.
YOU:   I absolutely need to. Please help.
ELIZA: That's quite interesting.
YOU:   I agree.
ELIZA: You say you agree ?
YOU:   █
```

DeepHack.Chat / ConvAI (2018)

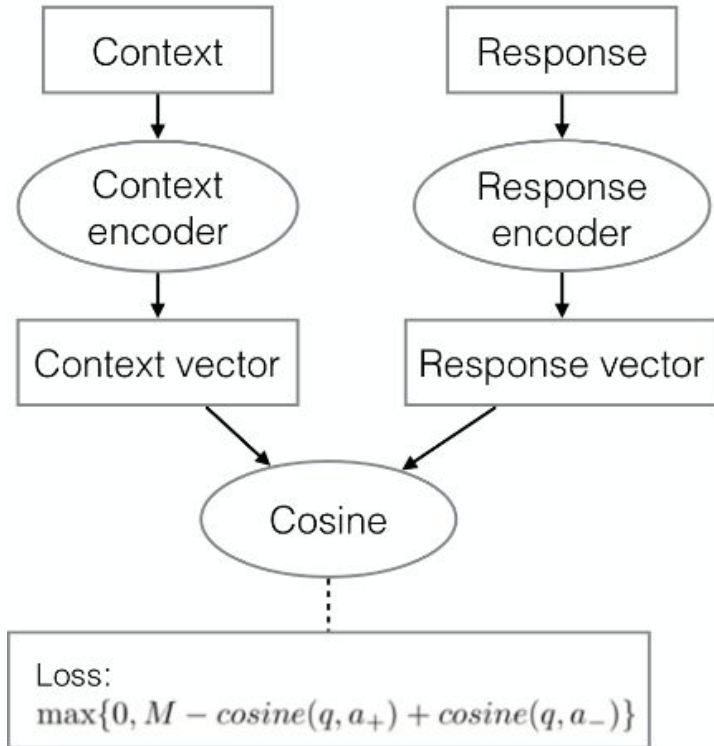
На хакатоне вы ничего лучше Элайзы не сделаете

Rank	Team Name	Dialogue quality	Role-playing quality	Average	Number of dialogues
1	NEUROBOTICS	0.5728571429	0.8901734104	0.7315152766	175
2	tensorborne	0.5558375635	0.7908163265	0.673326945	197
3	infinity	0.4936868687	0.7969543147	0.6453205917	198
4	Lost in conversation	0.3850574713	0.7687861272	0.5769217992	174
5	loopAI	0.4790419162	0.6107784431	0.5449101796	167
6	baseline	0.0462962963	0.55	0.2981481481	81
7	Sonic	0.01807228916	0.4578313253	0.2379518072	83

Общий подход

1. Намайним кучу сообщений с открытых источников (Twitter, reddit, Пикабу, Ответы @Mail.ru, Галя.py etc.)
2. Составим из них пары контекст-ответ.
Под контекстом тут может быть несколько последних сообщений, плюс мета-информация про диалог.
3. Обучим модель, которая учится по контексту генерировать / откуда-то ещё получать правдоподобный ответ.
4. PROFIT

Retrieval-based



- Обучим DSSM-подобную сеть: есть башни для запроса и ответа, перегоняющие текст в вектор
- Придумаем какой-нибудь лосс, который будет увеличивать скалярное произведение для положительных, и уменьшать — для отрицательных
- Намайним большую response corpora, и на реальные запросы будем отвечать сообщением с максимальным скором

Как можно формировать тройки?

Какие лоссы можно использовать?

Пример: кроссэнтропия

label smoothing

```
def bce_loss(X, Y, conf_true=0.9, conf_false=0.1):
```

```
    n = X.shape[0]
```

```
    logits = torch.mm(X, Y.transpose(0, 1))
```

^ считаем таблицу умножения

```
    identity = torch.eye(n, device=X.device)
```

```
    non_diagonal = torch.ones_like(logits) - identity
```

```
    targets = identity * conf_true + non_diagonal * conf_false
```

получаем матрицу с conf_true на диагонали и conf_false где-либо ещё

```
    weights = identity + non_diagonal / (n - 1)
```

^ чтобы не было перекоса в сторону негативов

```
    return F.binary_cross_entropy_with_logits(logits, targets, weights) *
```

n

^ на самом деле эта штука просто возьмет
и перемножит logits на target и сложит

на вход пришел батч размера n,
мы векторизовали контексты (X)
и ответы (Y) и хотим сделать n*n
независимых классификаций

Пример: Hinge loss

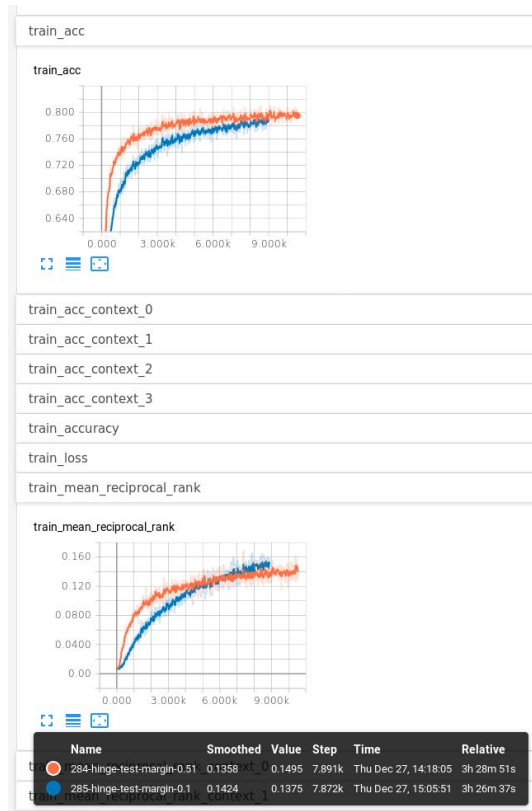
```
def hinge_loss(X, Y, margin=0.1):  
    batch_size = X.shape[0]  
    similarities = cosine_similarity_table(X, Y)  
                    ^ см. ниже  
  
    identity = torch.eye(batch_size, device=X.device)  
    non_diagonal = torch.ones_like(similarities) - identity  
    targets = identity - non_diagonal  
  
    weights = identity + non_diagonal / (batch_size - 1)  
    всё то же самое, но лосс другой: учитываем только то, что не превосходит margin  
    losses = torch.pow(F.relu(margin - targets * similarities), 2)  
  
    return torch.mean(losses * weights)
```



```
def cosine_similarity_table(X, Y):  
    X = F.normalize(X)  
    Y = F.normalize(Y)  
    return torch.mm(X, Y.transpose(0, 1))
```

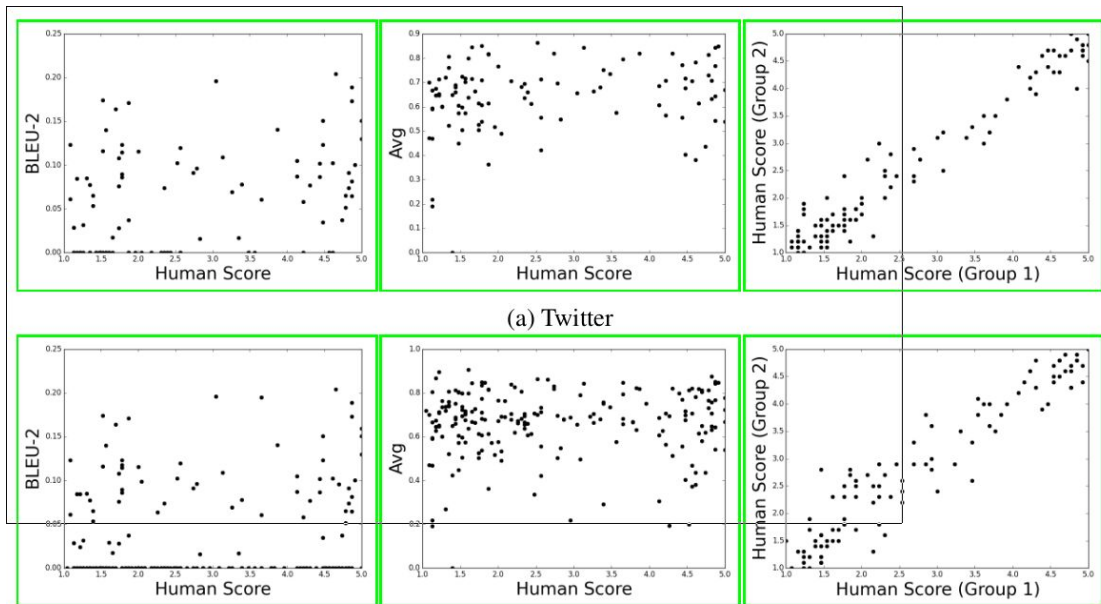
Метрики: ранжирование / классификация

- В первом примере мы напрямую решали задачу классификации: мы за бинарной кроссэнтропией стоит мощная теория
- Но абсолютная точность нам и не нужна: мы на самом деле решаем задачу ранжирования, и для этого человечество изобрело другие лоссы
- Нам больше интересен mean reciprocal rank (MRR) — $E[1/\text{rank}]$, где rank это позиция позитивного примера, если батч отсортировать по скорам



Метрики: token-level


- Есть много разных метрик, пришедших из перевода, суммаризации и прочих seq2seq-задач
- BLEU, METEOR, ROUGE
- Ни одна из них не работает
- [How NOT To Evaluate Your Dialogue System](#)




Метрики: если есть разметка


- Есть много разных фидбэков, которые можно посчитать и оптимизировать
- Длина диалога, количество существительных, наличие каких-то ключевых слов
- Можно ещё попросить пользователя дать оценку
- Если такие данные есть, то имеет смысл считать Discounted Cumulative Gain:


$$DCG_p = \sum_{i=1}^p \frac{rel_i}{\log_2(i+1)} = rel_1 + \sum_{i=2}^p \frac{rel_i}{\log_2(i+1)}$$

**Sergey** 11:46:52 PM
Кто лучше: алиса или алекса?


**BoltalkaTesting** 11:46:53 PM
Оба ужасны.


В тему 😊 Средненько ... Не в тему ☹

**Sergey** 7:04:31 PM
как поднять бабла

**BoltalkaTesting** 7:04:31 PM
Продавать курсы как заработать деньги.

В тему 😊 Средненько ... Не в тему ☹

**Sergey** 5:33:19 PM
Как выучить матан

**BoltalkaTesting** 5:33:19 PM
Зубрить и стараться выполнять домашнее задание.

В тему 😊 Средненько ... Не в тему ☹

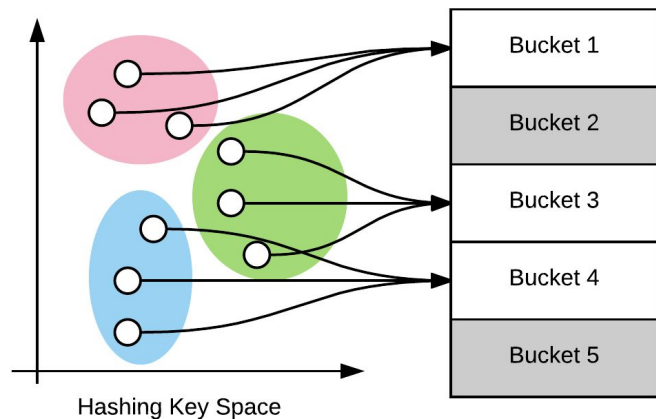
Как искать ответ в продакшне?

Высокий RPS, много кандидатов, большая размерность эмбедингов

Можно ли как-то учесть мета-информацию?

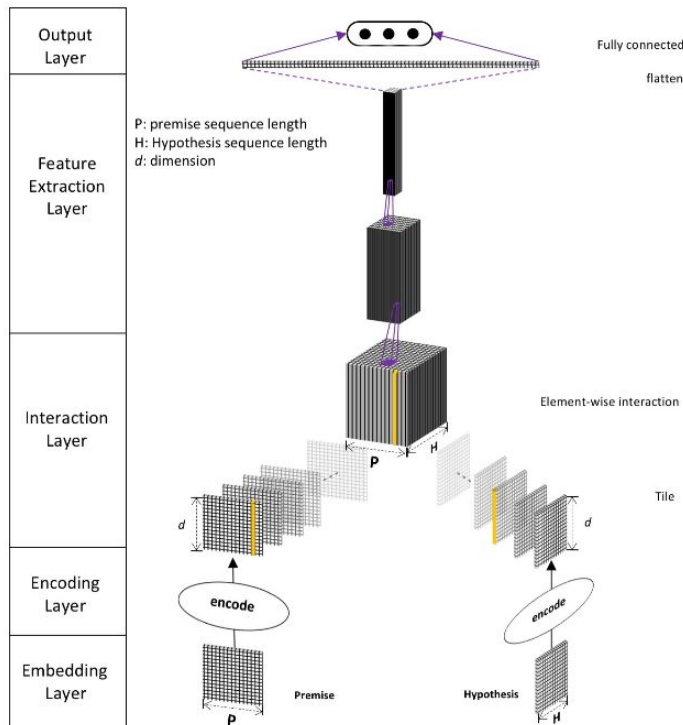
Мы хотим взять всю информацию о пользователе и засунуть в условный LightGBM

IR-системы: NMSLIB, Lucene, etc.



- Косинусное расстояние — это на самом деле bottleneck; нельзя вместить всю необходимую информацию в один вектор
- Высоконагруженные ранжирующие системы обычно имеют несколько слоев, каждый следующий тяжелее предыдущего
- Ранжирование по cosine similarity можно использовать как первый слой, потому что можно быстро найти несколько сотен хороших кандидатов⁶

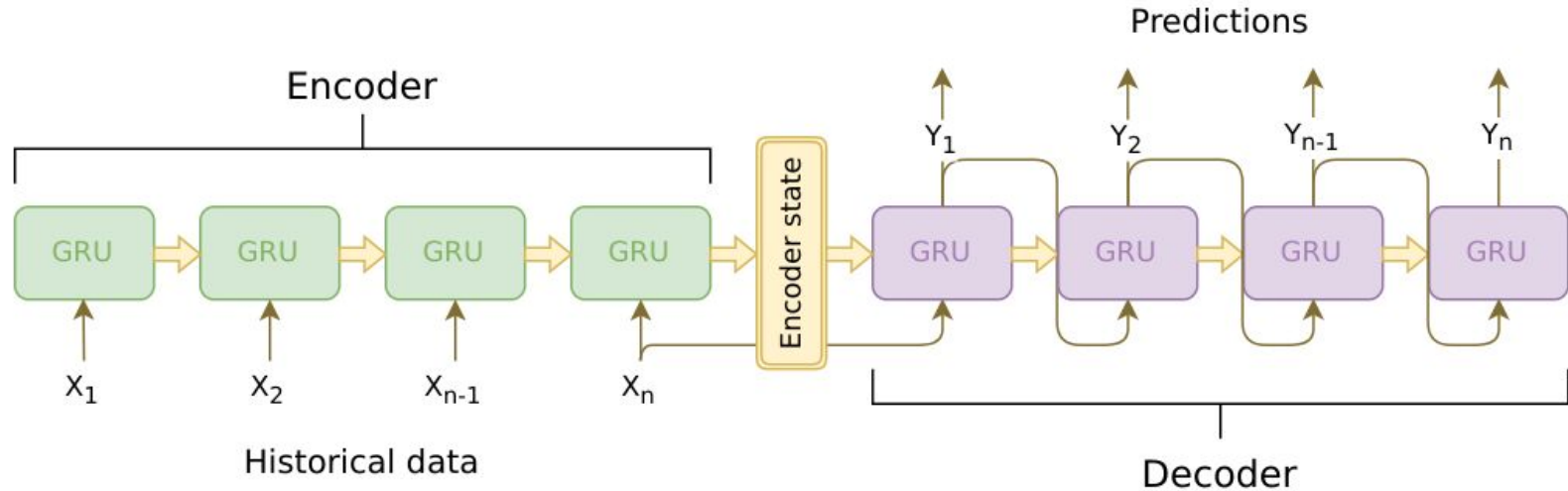
Более тяжелые модели



- Когда кандидатов осталось немного, их можно переранжировать какой-нибудь другой моделью, не ограниченной косинусным произведением в конце
- Последним слоем любых моделей не будет лишним наstack бустинг, куда можно сложить вообще всё, что мы знаем про пользователя
- [Natural Language Inference over Interaction Space](#)

Другой подход: seq2seq

Задача почти такая же, как в машинном переводе:
давайте получим вероятностное распределение ответов

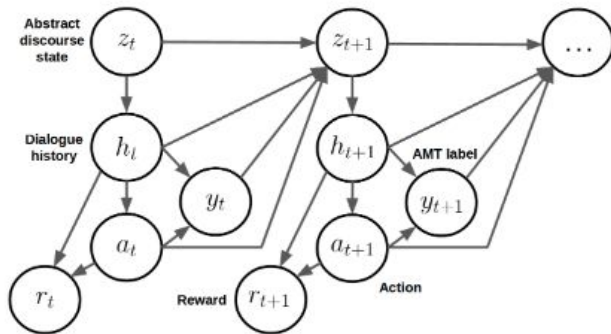
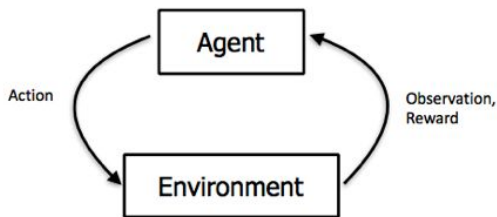


Seq2seq

- Не нужно хранить и фильтровать миллионы реплик и возиться с LSH
- Более обширный (бесконечный) набор вариантов ответа
- Возможность применить reinforcement learning, чтобы оптимизировать метрики напрямую
- Намного более сложный процесс фильтрации, если её делать
- Нельзя встроить редакторские ответы
- Холодный старт: нужно собрать солидный датасет и иметь много железа

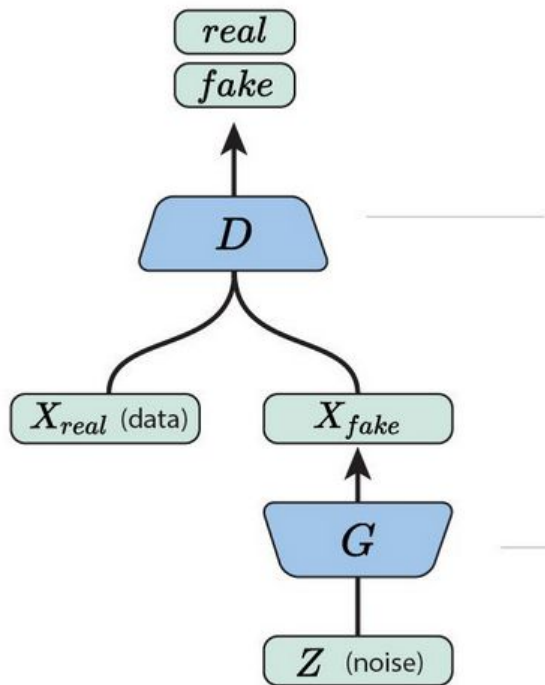
(На практике крутые ребята используют комбинацию дискриминативной и генеративной моделей.)

Reinforcement Learning



- Одно из применений RL: оптимизировать дискретные или вообще невычислимые метрики
- Основная идея такая: давайте попробуем повзаимодействовать с пользователями, соберем фидбэк (явно или неявно), и дообучим сеть так, чтобы она вероятнее повторила те действия, где фидбэк хороший
- [A Deep Reinforcement Learning Chatbot](#)

GAN



- Мечта всех AI исследователей — пройти тест Тьюринга
- Генератор: пишет правдоподобные продолжения диалога
- Дискриминатор: отличает реальные диалоги от сгенерированных
- Процесс обучения — своеобразная игра с нулевой суммой
- [Adversarial Learning for Neural Dialogue Generation](#)

Болталка говорит голосом народа



Как исправить

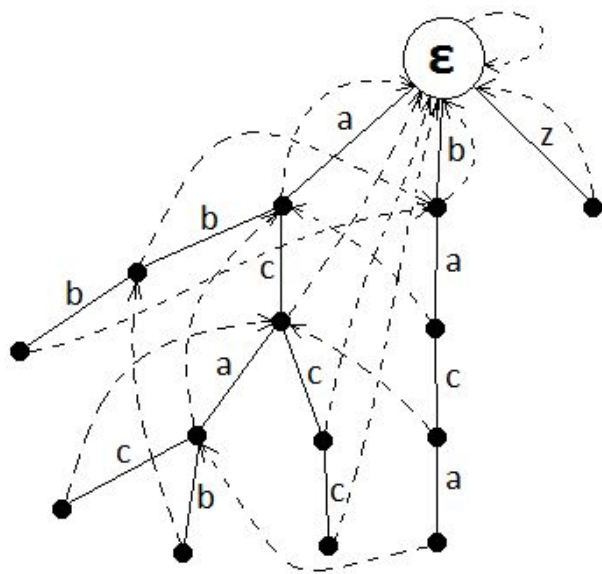


```
def filter(str):  
    if 'hitler' in str.lower():  
        return False  
    else:  
        return True
```

- Retrieval-based:
Написать руками кучу правил,
брать данные из более
надежных источников
- Generative:
Этими же правилами
фильтровать, либо искусственно
уменьшать вероятность того, что
было сгенерировано

Как делать фильтрацию для Retrieval-based?

Как исправить: стоп-слова



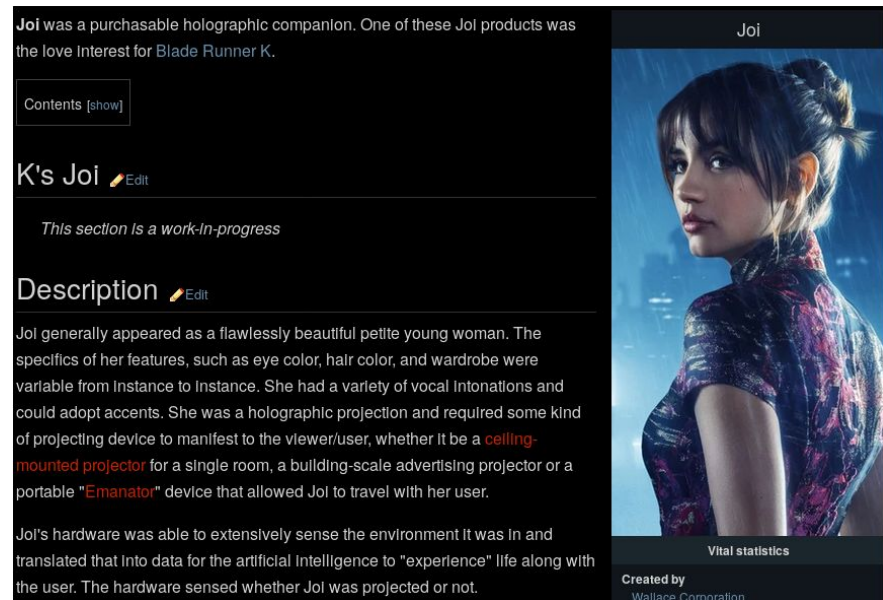
- Намайним много данных с «плохих мест»: экстремистских пабликов, порносайтов, религиозных и политических сайтов и т. д.
- Посчитаем их tf-idf относительно условной Википедии
- Запишем несколько тысяч самых частых в файл и будем фильтровать тексты, где они есть
- Вот и пригодился Ахо-Корасик: вы не зря в вуз ходили

Персонализация

В фильме Blade Runner 2049 у главного героя была виртуальная спутница жизни Джой.

В фильме было несколько сцен, в которых он говорил что-то вроде “чувство юмора б” и менял её характер.

В голосовых ассистентах на самом деле можно сделать так же.



Эмбе́ддинг пользо́вателя

- Соберем мета-информацию про пользователя:
простую — возраст, пол, работа;
более сложную — увлечения, черты характера
- Во время обучения будем подавать эту
информацию в сеть вместе с текстом, обучая
эмбе́ддинги каждого трейта — это поможет
улучшить модель
- Для реплик ассистента в проде выберем какую-
то фиксированную персональность: например,
восемнадцатилетняя девочка или
пятидесятилетний банкир

Эмбединг пользователя

У соцсетей боты должны получаться лучше

Topic	music, Ashin
Intent	request
Sentiment	sad, nerves -> neural
Opinion	neural
...	...
Age	25
Gender	male
Interests	singing, music ...
Occupation	salesman
Personality	quite, serious ...
...	...

Topic	music, Ashin
Intent	inform
Sentiment	happy
Opinion	positive
...	...
Age	18
Gender	female
Interests	music, games, movie, singing ...
Occupation	AI chatbot
Personality	reliable, sympathetic, affectionate...
...	...

BCE!