

Interactive Speaker Recognition

Применение обучения с подкреплением для решения задачи распознавания диктора

Вячеслав Головин
Евгений Шуранов (руководитель)

Huawei CBG AI и ФКН ВШЭ СПб

16.05.2023

Цель: повышение точности систем верификации / идентификации диктора при помощи выбора запрашиваемых у диктора слов. Такая система может, например, быть использована для подтверждения личности на мобильных устройствах.

Требования к системе:

- короткие запросы (не раздражаем пользователя),
- разнообразные запросы (не боимся спуфинга),
- высокая точность (без комментариев).

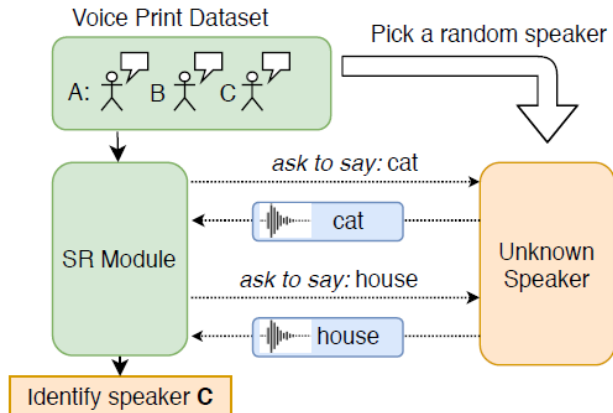
Предлагаемое решение: использование RL-агента для выбора запрашиваемых слов.

Новизна дипломной работы:

- переход от идентификации к верификации,
- более гибкая система для выбора слов,
- исследование влияния шума.

Interactive Speaker Recognition

Метод был предложен в статье *A Machine of Few Words — Interactive Speaker Recognition with Reinforcement Learning*, Mathieu Seurin et al., INTERSPEECH 2020, arXiv:2008.03127v1.

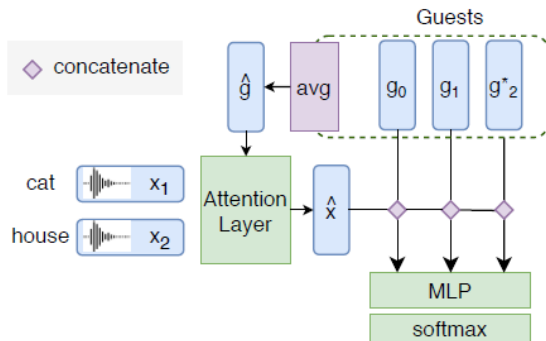


Важные особенности:

- 1 Рассматривается только задача идентификации.
- 2 Набор слов строго фиксирован.
- 3 Разные нейронные сети для двух задач SR Module — запроса слов (Enquirer) и идентификации диктора (Guesser).

Блок Guesser

Архитектура



Входные данные:

- эмбеддинги дикторов
 $G = [g_1; g_2; \dots g_K]$
- эмбеддинги слов
 $X = [x_1; x_2; \dots x_T]$

Выходные данные:

- вероятности
 $\{P(g_i = g^*) \mid i = 1..K\}$

Обозначения

- K количество гостей / дикторов
 T количество запрашиваемых слов

Блок Guesser

Псевдокод 1 итерации обучения

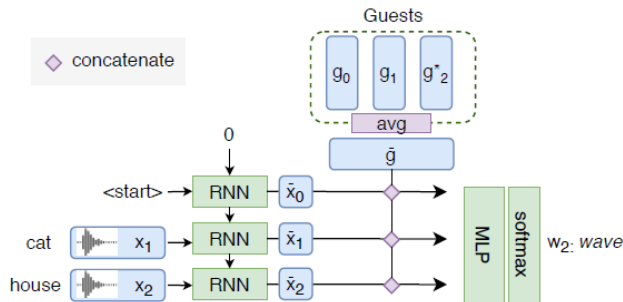
```
speaker_ids = speakers.sample(size=K)
G = voice_prints.get(speaker_ids)
target = randrange(0, K)
word_inds = randrange(0, V, size=T)
X = word_vocab.get(speaker=speaker_ids[target],
                  words=word_inds)
probabilities = guesser.forward(G, X)
loss = cross_entropy(probabilities, target)
```

Обозначения

- K количество гостей / дикторов
- T количество запрашиваемых слов
- V размер словаря — число доступных для запроса слов

Блок Enquirer

Архитектура



Входные данные:

- среднее эмб. дикторов

$$\hat{g} = \frac{1}{K} \sum_{i=1}^K g_k$$

- эмбединги слов

$$X = [x_1; x_2; \dots; x_t]$$

Выходные данные:

- вероятность выбрать каждое из слов

Обозначения

- K количество гостей / дикторов
- T количество запрашиваемых слов
- t количество запрошенных слов, $0 \leq t \leq T$

Блок Enquirer

Псевдокод 1 эпизода ISR-игры

```
speaker_ids = speakers.sample(size=K)
G = voice_prints.get(speaker_ids)
target = randrange(0, K)

g_hat = G.mean(dim=0)
x_i = start_tensor
X = []
for i in range(T):
    probs = enquirer.forward(g_hat, x_i)
    if training:
        word_inds = multinomial(probs).sample()
    else:
        word_ind = argmax(probs)
    x_i = word_vocab.get(speaker=speaker_ids[target], word=word_ind)
    X.append(x_i)

prediction = guesser.predict(G, X)
reward = 1 if prediction == target else 0
```

Входные данные

Для обучения использовался датасет **TIMIT**:

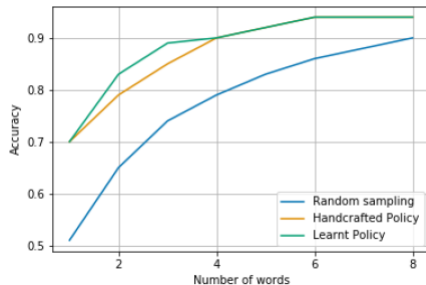
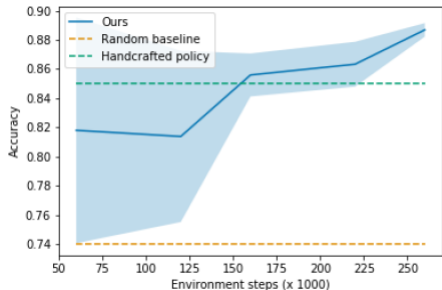
- 630 дикторов из США, 8 акцентов;
- каждый диктор произносит 10 предложений: 8 уникальных и 2 общих.

В качестве эмбеддингов использовались **x-vectors**, полученные с помощью нейронной сети, обученной на аугментированных датасетах для распознавания диктора *Switchboard*, *Mixer 6* и *NIST*.

- Эмбеддинги дикторов g получались с помощью усреднения эмбеддингов 8 уникальных предложений.
- Эмбеддинги слов x извлекались с помощью 2 общих предложений, т.е. сначала вырезались записи одиночных слов, которые затем пропусклись через нейронную сеть.

Результаты из статьи

$K = 5$ дикторов и $T = 3$ слова

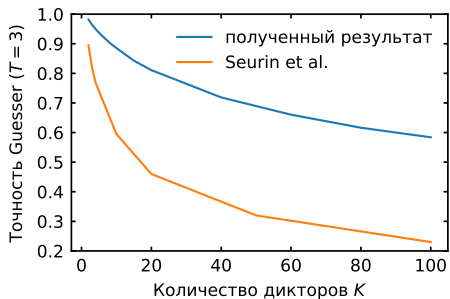
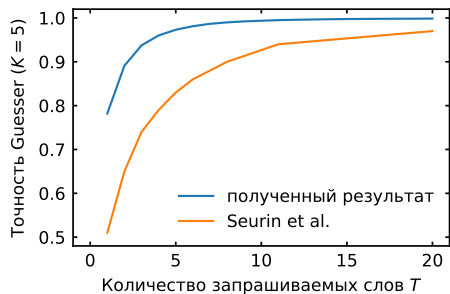


RL-агент при выборе запрашиваемых слов учитывает контекст — он опережает не только случайного агента, но и эвристического, выбирающего из подмножества “лучших” слов.

Преимущество RL-агента невелико и проявляется только при небольшом числе запрашиваемых слов.

Обучение и тестирование Guesser

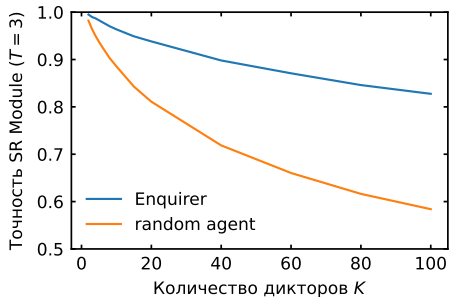
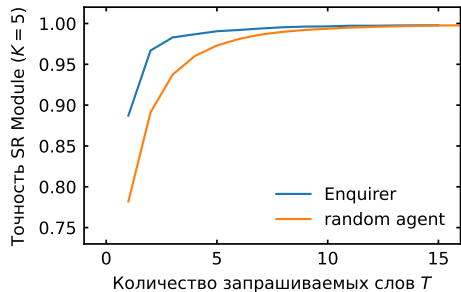
$K = 5$ дикторов и $T = 3$ слова при обучении



Вероятно, главная причина расхождения результатов — увеличение размерности эмбедингов (512 вместо 128 в статье). Неизвестно, как и зачем в статье производилось понижение размерности.

Обучение и тестирование Enquirer

$K = 5$ дикторов и $T = 3$ слова при обучении

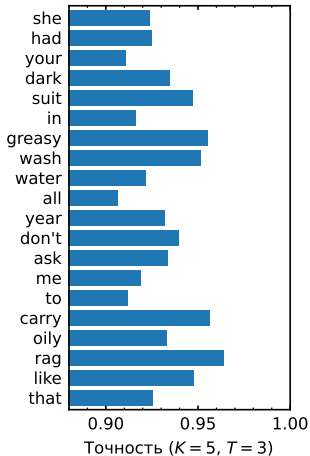


Для обучения использовалась **PPO**. Выбор слова при обучении и тестировании проводился по-разному:

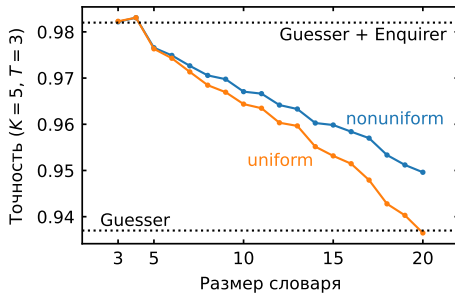
- train — сэмплирование из распределения,
- test — $\arg \max$ по не использованным ранее словам.

Эвристический агент

Алгоритм работы

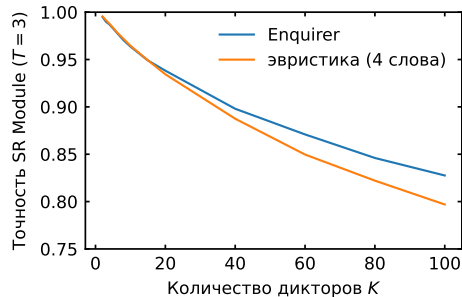
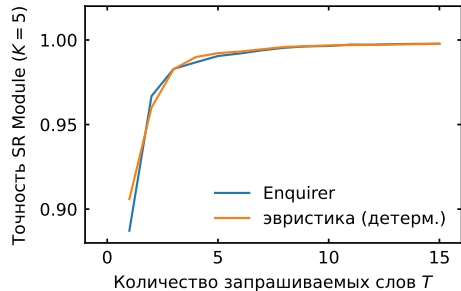


- 1 Рассчитываем точность на валидационной выборке.
- 2 Сэмплируем из слов с самой высокой точностью.



Эвристический агент

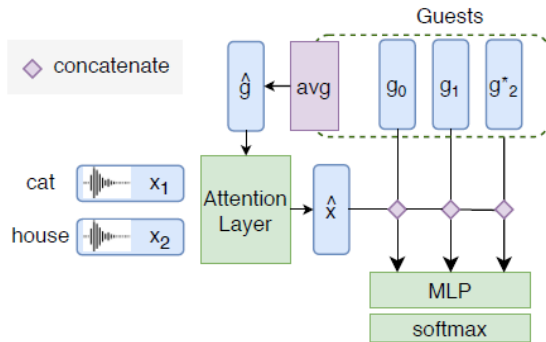
Сравнение с Enquirer



- Обучение в режиме с $K=5$ дикторами и $T=3$ запрашиваемыми словами.
- При $T=1$ Enquirer работает плохо. Можно обучать на 1 слове, но такая система плохо работает в других режимах.

Верификация

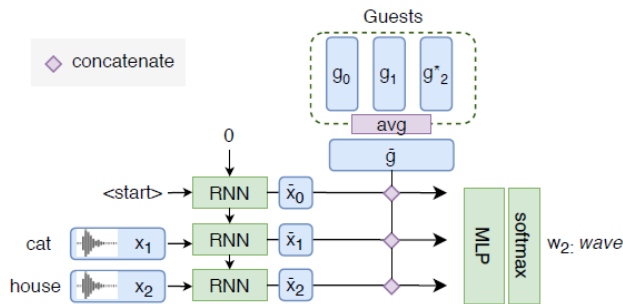
Как преобразовать Guesser



- только 1 диктор $\implies \hat{g} = g_0$
- softmax \rightarrow sigmoid
- при $T = 3$ точность 0.91
(\sim как в классификации с $K = 7$ гостями)
- MLP \rightarrow CosineSimilarity тоже работает, но хуже

Верификация

Как преобразовать Enquirer



- только 1 диктор $\implies \hat{g} = g_0$
- архитектуру менять не нужно
- можно взять веса для идентификации

Результаты ($T = 3$ слова):

Выбор слов	Точность
случайный	0.895
Enquirer	0.933
эвристика	0.917

Обучение в более тяжелом режиме

Выбор слов	Режим обучения	Точность
случайный	$T = 3$	0.895
Enquirer		0.933
эвристика		0.917
случайный	$T = 2$	0.913
Enquirer		0.947
эвристика		0.945

Таблица: Точность верификации, $T = 3$ запрашиваемых слова

Обучение в более тяжелом режиме

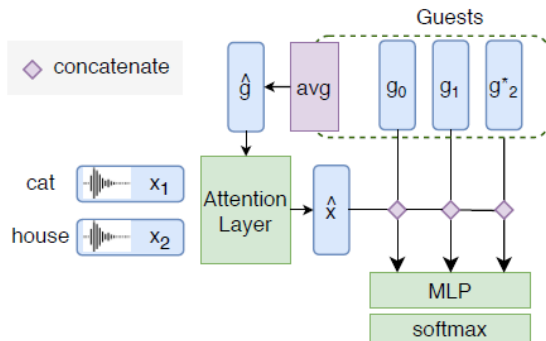
Выбор слов	Режим обучения	Точность
случайный	$K = 5$ $T = 3$	0.937
Enquirer		0.982
эвристика		0.984
случайный	$K = 20$ $T = 2$	0.951
Enquirer		0.989
эвристика		0.988

Таблица: Точность идентификации, $K = 5$ дикторов, $T = 3$ запрашиваемых слова

CodebookEnquirer

Мотивация и принцип работы

Очевидный недостаток архитектуры Enquirer — строго фиксированный набор слов, при любом его изменении нужно обучать заново или делать fine-tuning.



Предлагаемые изменения:

- MLP возвращает эмбединг слова, а не вероятности;
- добавляется Codebook — набор (фиксированных) эмбедингов слов;
- вероятность выбрать слово из Codebook обратно пропорциональна расстоянию между эмбедингами.

CodebookEnquirer

Результаты

Выбор слов	Режим обучения	Точность
случайный		0.937
Enquirer	$K = 5$	0.982
CodebookEnquirer	$T = 3$	0.964
CodebookEnquirer (половина слов)		0.970
случайный		0.951
Enquirer	$K = 20$	0.989
CodebookEnquirer	$T = 2$	0.990
CodebookEnquirer (половина слов)		0.980

Таблица: Точность идентификации, $K = 5$ дикторов, $T = 3$ запрашиваемых слова

Добавление шума

- 6 различных вариантов шума из датасета MUSAN для каждого слова: rain, car, crowd, typing, hum, white — а также исходная чистая аудиозапись.
- SNR 3 dB.
- Тип шума не меняется в течение эпизода.

Модель	Идентификация	Верификация
Guesser	0.887	0.895
Guesser + Enquirer	0.946	0.934
Guesser + эвристика (3 лучших)	0.957	0.938

Таблица: Точность идентификации и верификации в стандартных режимах ($T = 3$ слова, $K = 5$ гостей при идентификации)