Interactive Speaker Recognition

Применение обучения с подкреплением для решения задачи распознавания диктора

Головин Вячеслав Сергеевич
Шуранов Евгений Витальевич (руководитель)

Huawei CBG AI и ФКН ВШЭ СПб

07.06.2023

Задача распознавания диктора (Speaker Recognition)

Два типа задач:

1. **Идентификация** — по услышанной речи выбираем одного диктора из списка.
2. **Верификация** — по услышанной речи решаем, произнёс ли её конкретный диктор.

Фактически обе задачи сводятся к определению меры похожести между двумя наборами данных:

1. Векторы признаков, вычисленные из полученных ранее аудиозаписей речи (**эмбеддинги дикторов** или голосовые подписи).
   Обозначение: $G = [g^k]_{k=1}^{K},\ K \in \mathbb{N}$.
2. Векторы признаков аудиозаписей речи, полученных сейчас (**эмбеддинги** произнесенных **слов**).
   Обозначение: $X = [x^t]_{t=1}^{T},\ T \in \mathbb{N}$.

Область исследования

Зачем нам *Interactive* Speaker Recognition

Некоторые системы распознавания запрашивают у диктора произносимые фразы. Логично выбирать эти слова и фразы таким образом, чтобы

- точность распознавания была выше,
- количество запросов было меньше,
- они были разнообразными (боремся со спуфингом).

Исследуемый подход: использование нейросетевого RL-агента для выбора запрашиваемых слов.

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

Цель и задачи

Цель: повышение точности систем распознавания диктора при помощи выбора запрашиваемых у диктора слов.

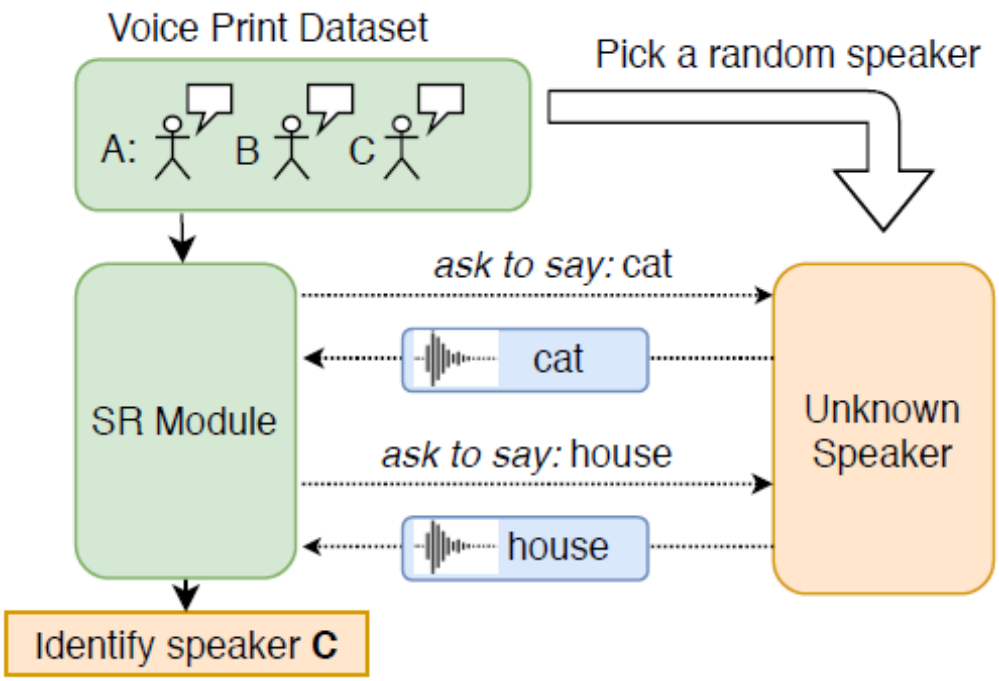
Задачи:

- Воспроизведение результатов, достигнутых в исходной статье.
- Улучшение и модификация изначальной системы:
  - Переход от идентификации к верификации.
  - Использование произвольного набора запрашиваемых слов.
  - Проверка работы при добавлении шума.
  - Использование других эмбеддингов.

Interactive Speaker Recognition

Здесь и далее изображения из *A Machine of Few Words — Interactive Speaker Recognition with Reinforcement Learning*, Mathieu Seurin et al., INTERSPEECH 2020, arXiv:2008.03127v1.

Использовался датасет TIMIT (630 дикторов, 20 слов).

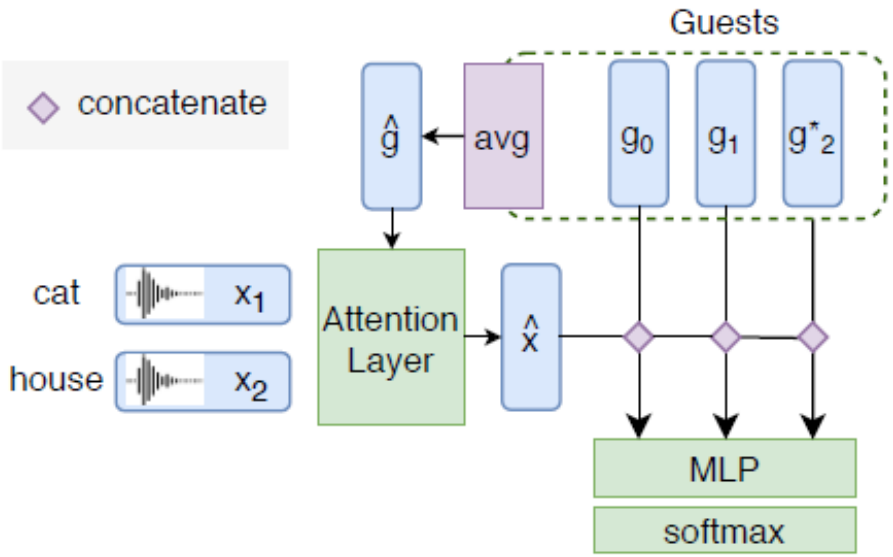


Важные особенности статьи:

1. только идентификация
2. фиксированный набор слов
3. разные нейронные сети для запроса слов (`Enquirer`) и идентификации (`Guesser`)

Архитектура Guesser

Пытаемся угадать диктора



Входные данные:

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

Выходные данные:

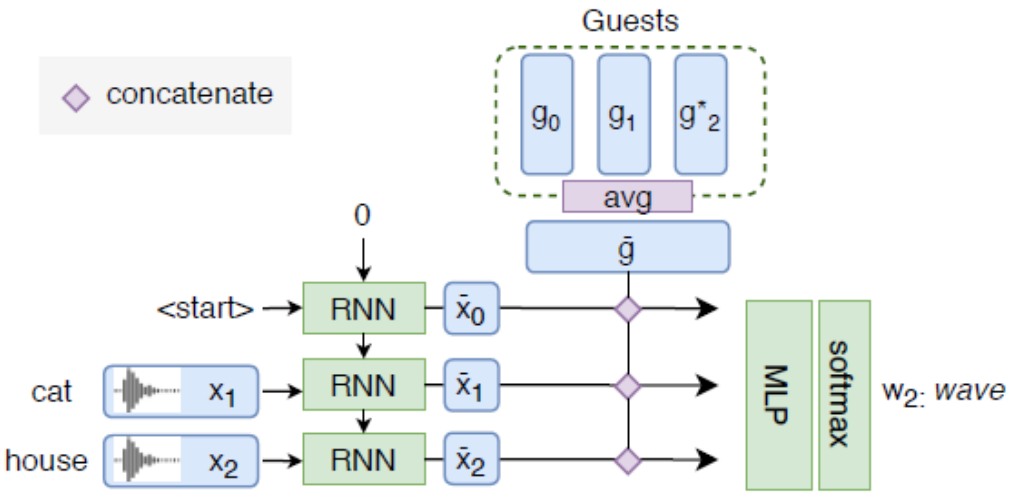
- вероятности $\{P(g_i = g^*) \mid i = 1..K\}$

Обозначения

K	количество гостей / дикторов
T	количество запрашиваемых слов

Архитектура Enquirer

Выбираем, какое слово мы спрашиваем у диктора



Входные данные:

- среднее эмб. дикторов $\hat{g} = \frac{1}{K}\sum_{i=1}^{K} g_k$
- эмбеддинги слов $X = [x_1; x_2; \ldots; x_t]$

Выходные данные:

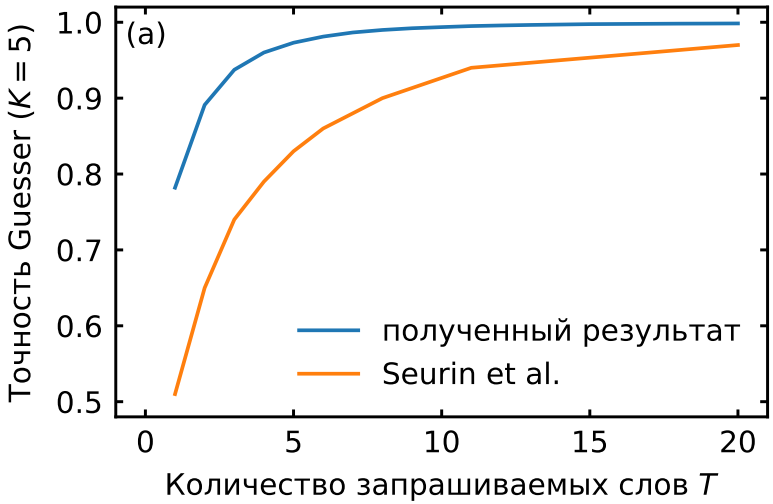
- вероятность выбрать каждое из слов

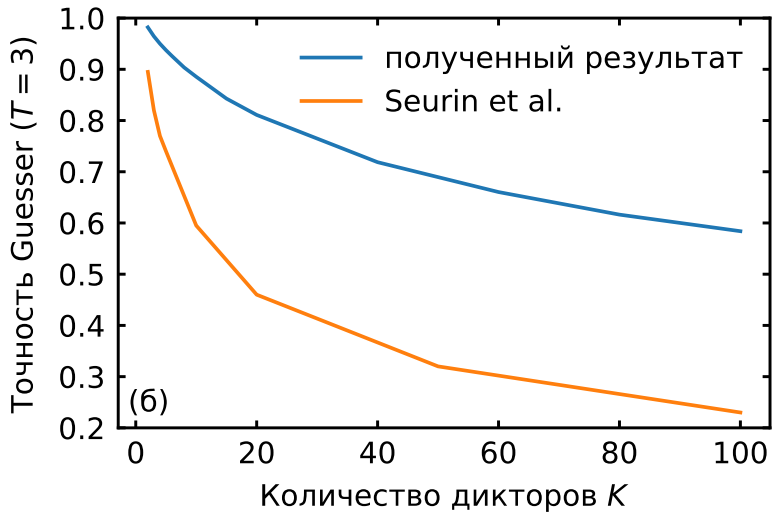
Обозначения

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

Обучение и тестирование Guesser

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



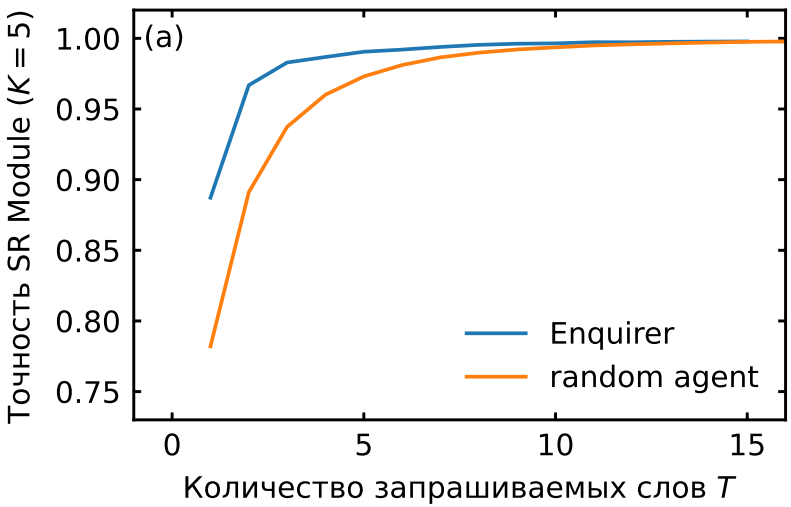


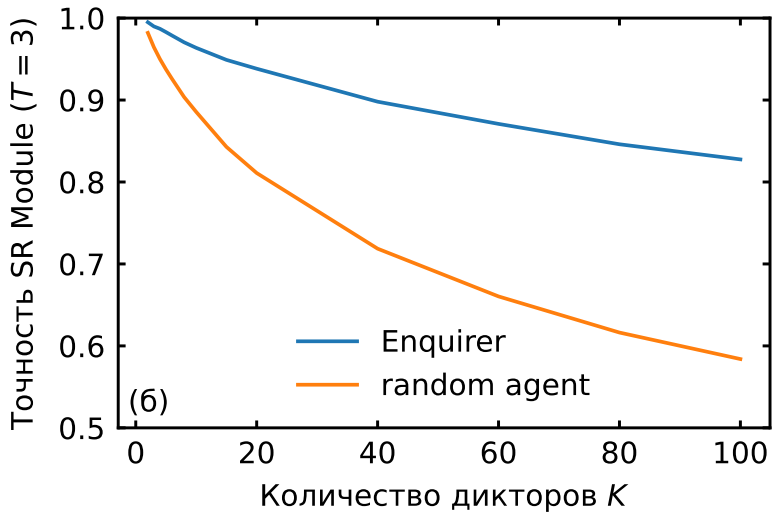
В качестве эмбеддингов использовались **x-vectors** из фреймворка **Kaldi**.

Вероятно, главная причина расхождения результатов — увеличение размерности эмбеддингов (512 вместо 128). Как и зачем в статье производилось понижение размерности неизвестно.

Обучение и тестирование Enquirer

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





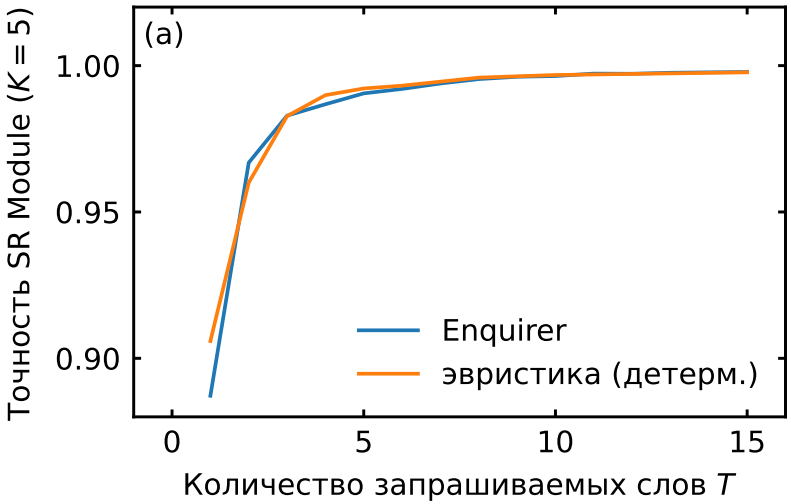
Для обучения использовался **PPO**. Выбор слова при обучении и тестировании проводился по-разному:

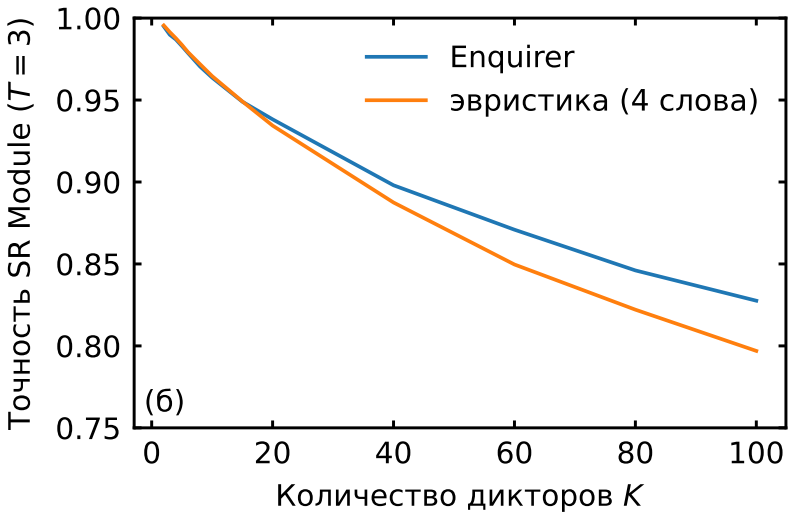
- `train` — сэмплирование из распределения,
- `test` — arg max по не использованным ранее словам.

Награда: 1.0 — `Guesser` угадал диктора, 0.0 — иначе.

Enquirer против эвристики

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



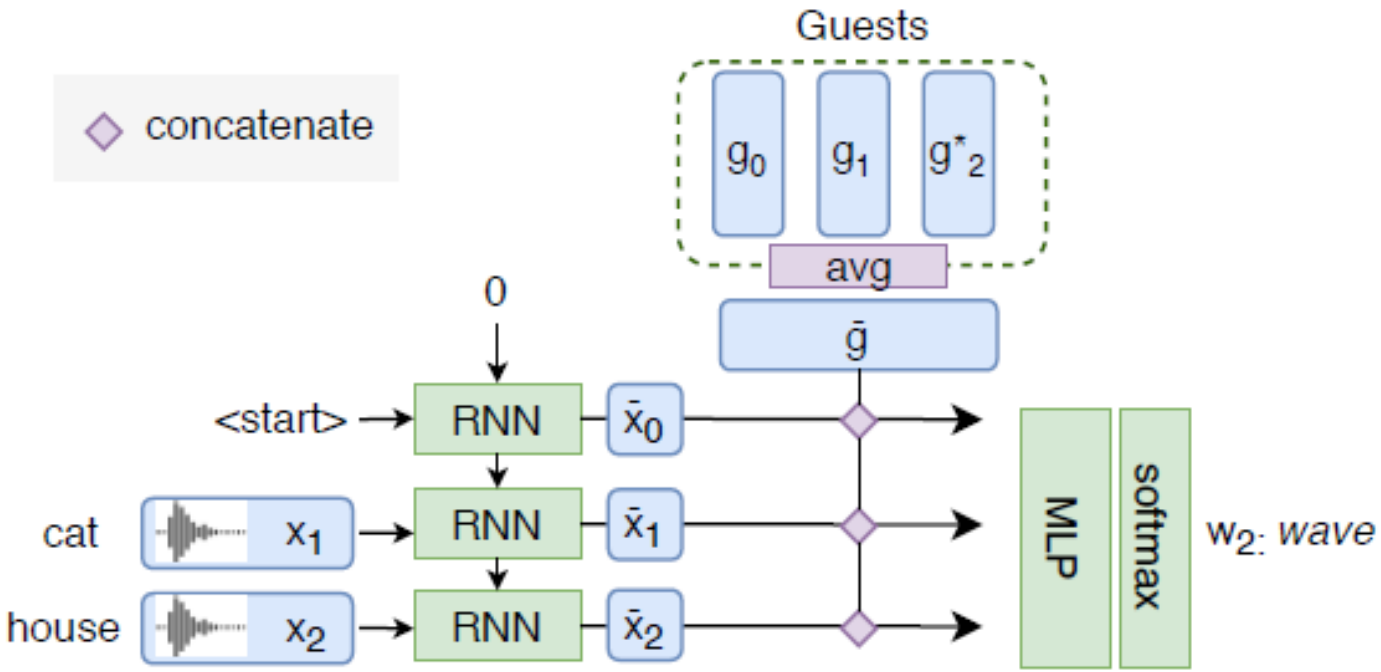


Эвристический агент не обращает внимание на контекст и (практически) всегда запрашивает одни и те же слова.

Для выбора слов используется средняя точность `Guesser` на валидационной выборке при использовании этого слова.

От идентификации к верификации

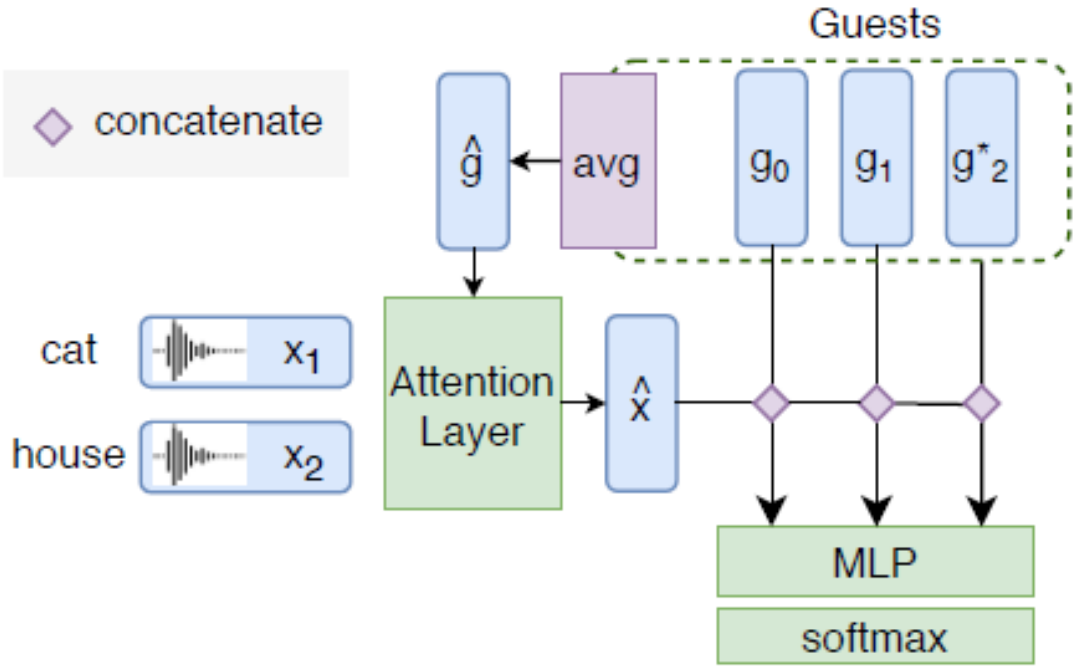
$T = 3$ слова



- `Enquirer`: не меняем ничего (даже веса)

От идентификации к верификации

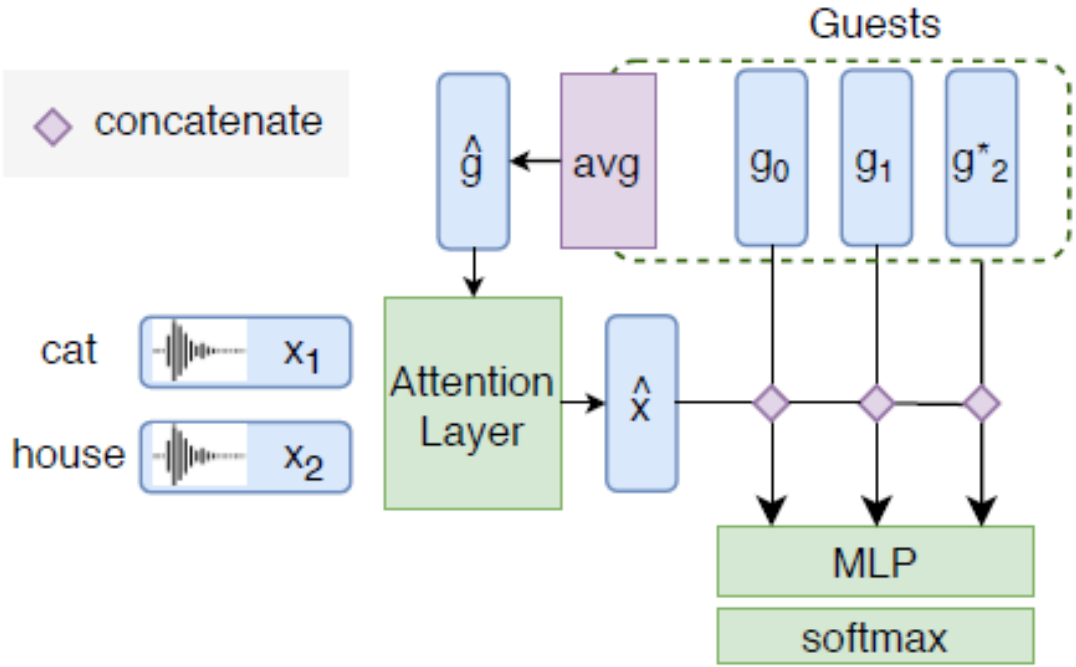
$T = 3$ слова



- `Enquirer`: не меняем ничего (даже веса)
- `Guesser`: меняем `softmax` на `sigmoid`

От идентификации к верификации

$T = 3$ слова



- `Enquirer`: не меняем ничего (даже веса)
- `Guesser`: меняем `softmax` на `sigmoid`

Выбор слов	Точность
случайный	0.895
`Enquirer`	0.933
эвристика	0.917

Другие эксперименты

1. Подбор режима обучения.
   - Для улучшения точности можно обучать модели в более тяжелых режимах (например, $K = 20$, $T = 2$).
2. `CodebookEnquirer` — гибкая система выбора слов.
   - Модифицируем `Enquirer` таким образом, что выбор осуществляется не из фиксированного набора слов, а из списка эмбеддингов.
   - Работает (небольшое падение точности), даже если мы обучаем и тестируем модель на разных наборах слов.
3. Добавление шума
   - Добавляем к аудиозаписям слов 6 видов шума из MUSAN.
   - Не меняем тип шума в течение игры.
   - Результаты принципиально не изменяются.
4. Альтернативные эмбеддинги
   - Вместо **x-vector** используем нейронную сеть, обученную с помощью контрастного прогнозирующего кодирования (**CPC**).
   - Существенное увеличение точности, `Enquirer` обучается.

Выводы

- Исследованный подход работает — точность идентификации существенно повышается при добавлении выбирающего слова агента.
- Модель можно сделать практически полезной: легко перейти от идентификации к верификации и от фиксированного набора слов к произвольному.
- В большинстве режимов (очень) простая эвристика оказывается не хуже нейросетевого агента для выбора слов (`Enquirer`).

Приложение

Псевдокод 1 итерации обучения Guesser

```
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

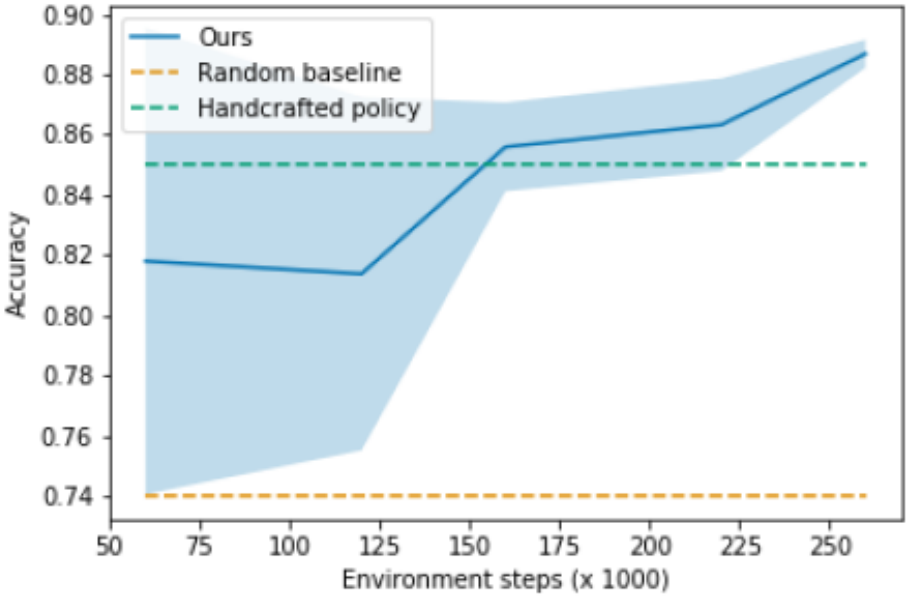
K	количество гостей / дикторов
T	количество запрашиваемых слов
V	размер словаря — число доступных для запроса слов

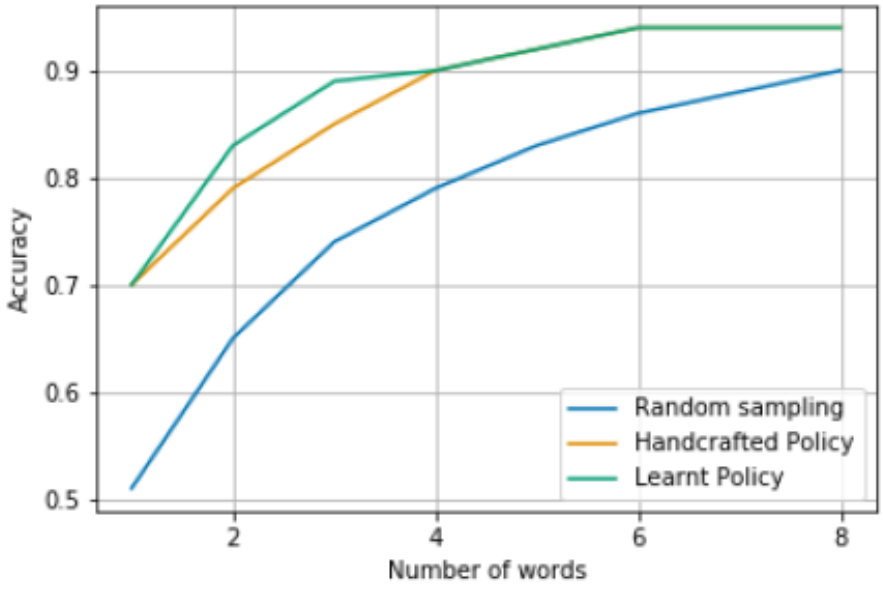
Псевдокод 1 эпизода ISR-игры

```
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
```

Результаты из статьи

$K = 5$ дикторов и $T = 3$ слова



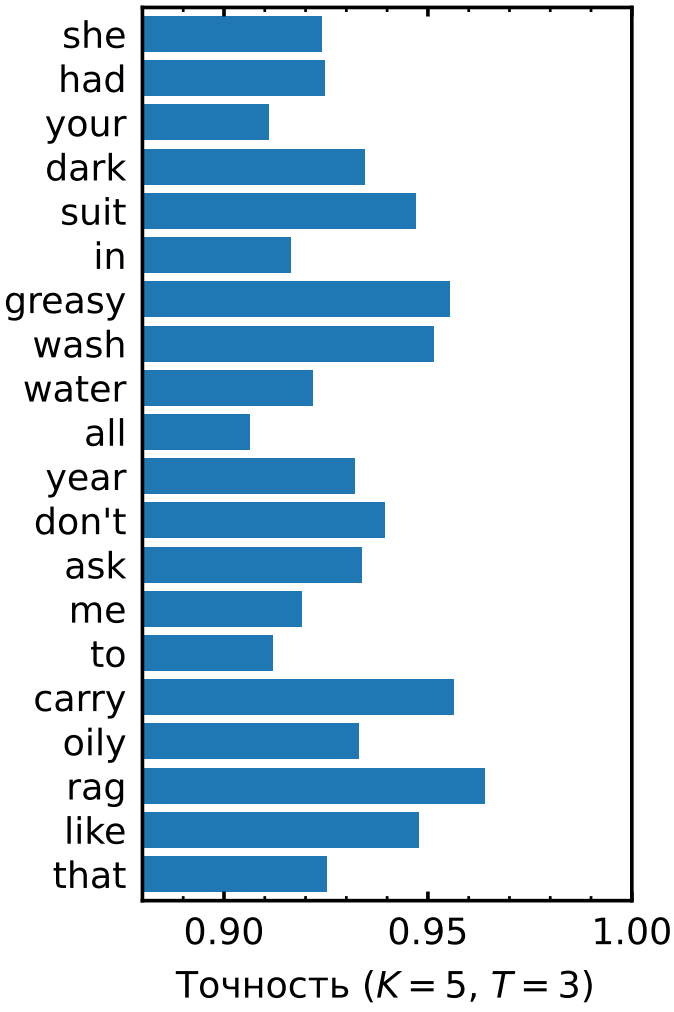


RL-агент при выборе запрашиваемых слов учитывает контекст — он опережает не только случайного агента, но и эвристического, выбирающего из подмножества “лучших” слов.

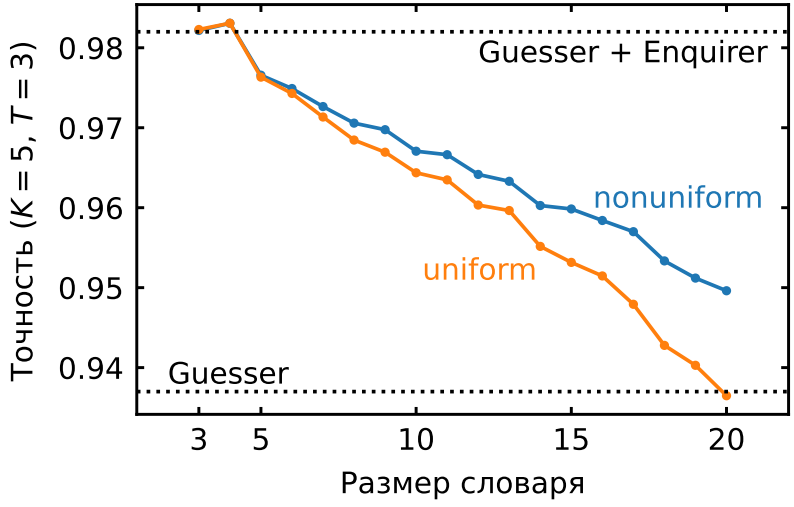
Преимущество RL-агента невелико и проявляется только при небольшом числе запрашиваемых слов.

Эвристический агент

Алгоритм работы



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



Обучение в более тяжелом режиме

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

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

Обучение в более тяжелом режиме

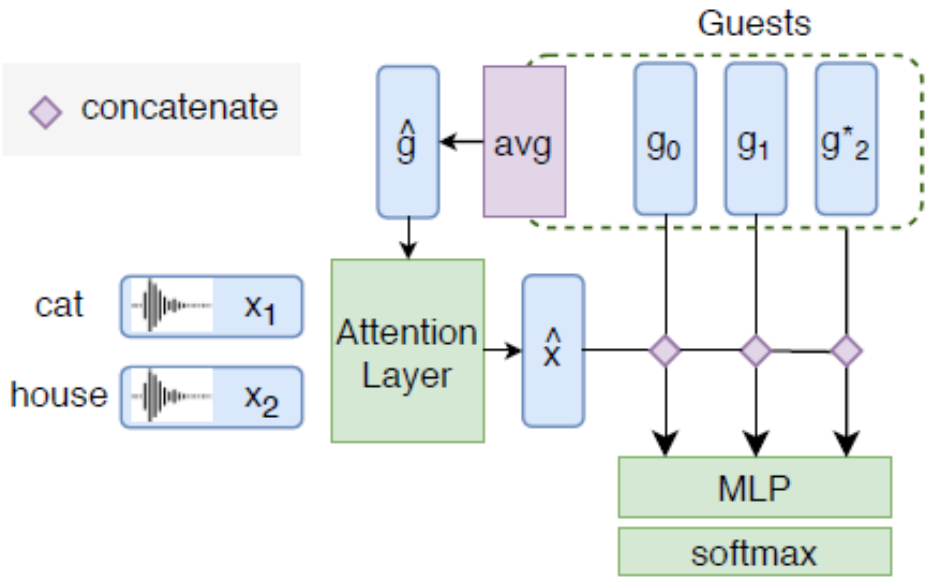
Выбор слов	Режим обучения	Точность
случайный	$K = 5$	0.937
`Enquirer`	$T = 3$	0.982
эвристика		0.984
случайный	$K = 20$	0.951
`Enquirer`	$T = 2$	0.989
эвристика		0.988

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 эмбеддинг слова, а не вероятности;
- добавляется Codebook — набор (фиксированных) эмбеддингов слов;
- вероятность выбрать слово из Codebook обратно пропорциональна расстоянию между эмбеддингами.

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)