Аннотация

аннотация на русском

Abstract

abstract in english

Оглавление

Введение

Данная работа посвящена интерактивному подходу к решению задачи распознавания диктора (*Speaker Recognition*)[1]. Оригинальный метод был предложен в [1], и значительная часть работы посвящена его описанию и практической реализации. С момента публикации этой статьи (2020 год) уже прошло достаточное количество времени, но она не стала популярной — по данным *Google Scholar* на момент написания этого отчёта она была процитирована 6 раз[2]. Тем не менее, нам (автору, его научному руководителю и коллегам из лаборатории Huawei CBG AI) она показалась заслуживающей внимания. На это есть ряд причин.

В первую очередь стоит отметить оригинальность предложенного подхода. Большинство работ, в той или иной степени затрагивающие задачу распознавания диктора, посвящены способам как можно лучше определять диктора на основе уже существующих аудиозаписей. Рассматриваемая работа ставит проблему иначе — какие слова или фразы должен произнести диктор, чтобы уже существующая система смогла распознать его как можно быстрее и надёжнее. Чем-то такой подход напоминает концепцию активного обучения (*active learning*) — разметки только тех данных, которые являются наиболее важными для решающей функции.

Другой причиной интереса к работе стала возможность её потенциального использования в конечном продукте — систе-

[1]Здесь и далее для ясности мы иногда будем указывать более распространённые названия терминов на английском языке.

[2]Одна из этих цитат — диссертация первого автора статьи.

ме аутентификации пользователя на мобильном устройстве или персональном ассистенте. Предополагается, что такая система будет спрашивать пользователя произнести ту или иную фразу, пока она не станет уверена, что перед ней действительно находится настоящий владелец прибора. В таком случае логично делать не случайные запросы, а такие, которые позволят системе как можно быстрее идентифицировать пользователя. При этом, как будет пояснено далее, возможность делать разнообразные запросы тоже является преимуществом. Кроме того, этот подход может быть использован и для других задач, например, для синтеза речи с определённым голосом.

Таким образом, целью данной работы является разработка системы распознавания диктора, в которой высокая точность достигается за счёт выбора запрашиваемых у диктора слов. Первоочерёдной задачей работы стало воспроизведение результатов, достигнутых в [1]. Другими задачами стали адаптация изначальной модели для целевого конечного продукта (системы аутентификации пользователя) и повышение её точности.

Сформулированные задачи определяют структуру отчёта. В главе 1 дано подробное описание исследуемого метода. Глава 2 посвящена вопросам имплементации и полученным при этом результатам. Модификации изначальной системы (например, переход к задаче верификации) рассматриваются в главе 3.

1. Распознавание диктора в интерактивном режиме

1.1. Задача распознавания диктора

Распознавание диктора является одной из задач обработки речи — обширного научного-исследовательского направления с долгой и богатой историей. Как и в случае со многими другими научными направлениеями, в последние годы обработка речи стала активно использовать методы машинного обучения, в частности нейросетевые модели. Так, впечатляющих результатов удалось добиться не только в распознавании диктора [2, 3], но и в смежных областях автоматического распознавания [4] и синтеза речи [5].

Задача распознавания диктора, как нетрудно понять из названия, заключается в определении личности человека по аудиозаписи его речи. Если говорить чуть более строго, задачей является сопоставление некоторой аудиозаписи речи неизвестного человека с некоторым набором дикторов. В случае решения задачи *идентификации* этот набор состоит из нескольких дикторов, при этом мы точно знаем, что один из них произнес анализируемую нами речь. Соответственно, в таком случае задачей системы является правильный выбор диктора. В случае решения задачи *верификации* нам известна информация только об одном дикторе, и от нас требуется определить, произнёс ли он речь на предоставленной аудиозаписи.

Может показаться, что две указанные проблемы существенно отличаются и, соответственно, требуют для своего решения

различные системы. На самом деле, это не так. Задача распознавания диктора в общем случае может быть представлена [6] как сравнение модели диктора и вектора признаков анализируемой речи. Тогда каждой паре диктор–аудио можно сопоставить некоторое число, характеризующее степень соответствия. При идентификации мы получим несколько чисел, наибольшее из которых будет указывать на наиболее вероятного диктора. При верификации нам нужно будет просто преобразовать единственное полученное число в вероятность. Если говорить в терминах глубокого обучения, отличаться будет только последняя функция активации: при идентификации это будет softmax, при верификации — sigmoid.

В целом модель диктора может принимать различные формы. Например, в течение продолжительного времени одним из ведущих методов моделирования дикторов была [6] модель смеси гауссиан (*Gaussian Mixture Model*). В современных нейросетевых системах модель диктора обычно представляет собой вектор признаков. Тогда задача определения степени соответствия диктора и речи сводится к сравнению двух векторов, которое можно осуществлять как с помощью вычисления косинусного расстояния, так и с использованием более сложных методов — вероятностного линейного дискриминантного анализа (*Probabilistic Linear Discriminant Analysis*) [7] или нейронных сетей [8]. В оригинальной статье [1], посвященной исследуемому в этой работе методу, нейронные сети используются как для моделирования диктора, так и для расчёта метрики соответствия диктора и речи. Мы будем придерживаться такого же подхода.

1.2. Общие принципы метода

На рис. 1 проиллюстрирован метод интерактивного распознавания диктора. Изначально у нас имеется K дикторов, далее мы случайно выбираем из них одного целевого — его модуль распознавания диктора (*SR Module*) и будет пытаться угадать. Как уже было сказано ранее, каждый диктор характеризуется своим вектором признаков (эмбеддингом диктора или *voice print* — голосовой подписью). Данные о дикторах передаются модулю распознавания диктора, после чего он выбирает, какое слово должен произнести угадываемый диктор. Диктор произносит это слово, новая аудиозапись поступает на вход SR-модуля, и он запрашивает новое слово. Процесс повторяется, пока SR-модуль не получает T аудиозаписей слов, после чего он пытается угадать целевого диктора.

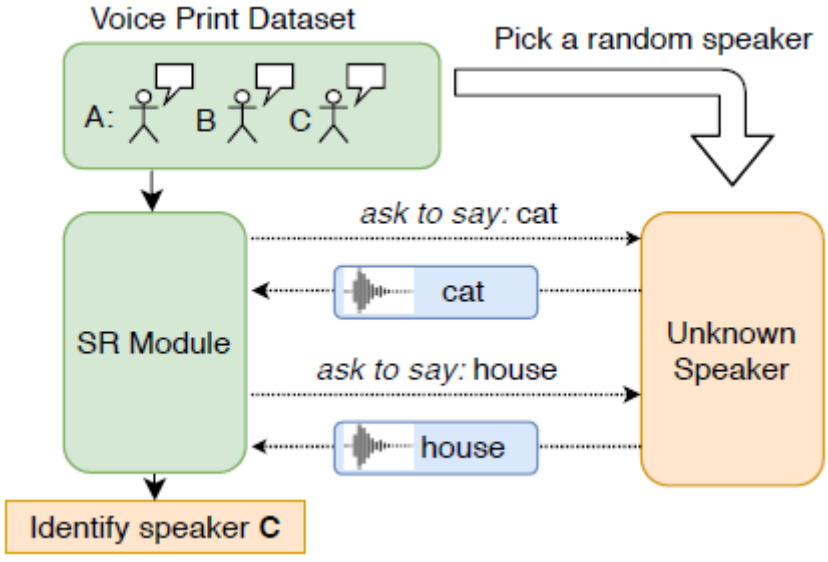

Рис. 1. Схема интерактивной игры по определению диктора [1].

Стоит отметить несколько важных моментов рассматривае-

мого метода:

1. Выше был описана задача идентификации. В этой работе мы тоже будем в основном решать именно её, хотя с практической точки зрения нам интереснее верификация. Такое решение объясняется двумя причинами. Во-первых, задача идентификации является более гибкой — варьируя число дикторов K её можно делать более или менее сложной. Во-вторых, как будет продемонстрировано в разделе 3.1, перейти от идентификации к верификации достаточно просто.

2. Все векторы признаков дикторов и произнесенных слов вычисляются с помощью отдельной нейронной сети, более подробное описание будет дано в разделе 2.1. Далее мы будем обычно говорить не об аудиозаписях, а об эмбеддингах дикторов и слов, которые и будут получать на вход нейросетевые модели для распознавания диктора.

3. Две функции SR модуля — идентификация диктора и выбор запрашиваемых слов — выполняют две различные нейронные сети. Такое разделение вовсе не является обязательным, но оно позволяет использовать обучение с учителем для нейросети, решающую задачу идентификации.

1.3. Нейросетевые модели — `Guesser` и `Enquirer`

Итак, рассмотрим внутреннее устройство модуля для распознавания диктора. В первую очередь стоит изучить модель, решающую непосредственно задачу идентификации, которую

авторы [1] назвали `Guesser`. Её архитектура (рис. 2) достаточно проста. Сначала эмбеддинги дикторов g_i усредняются, полученный вектор $\hat{g}$ подаётся в качестве запроса в блок с механизмом внимания (*Attention Layer*).

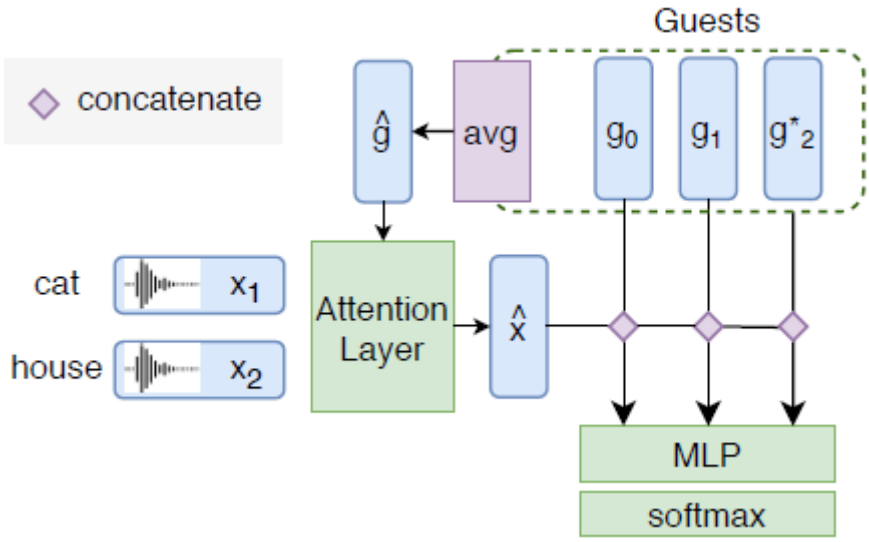

Рис. 2. Архитектура нейросети `Guesser` [1].

Другие входные данные — эмбеддинги произнесённых диктором слов x_i — используются в Attention слое в качестве ключей и значений. В [1] приведены следующие формулы для расчёта вектора $\hat{x}$:

$$e_t = \text{MLP}([x_t, \hat{g}]); \quad \alpha = \text{softmax}(e); \quad \hat{x} = \sum_t \alpha_t x_t;$$

где MLP — многослойный перцептрон, а $[.,.]$ — операция конкатенации. Как мы видим, в данном случае используется аддитивная версия механизма внимания.

Далее вектор $\hat{x}$ конкатенируется к каждому эмбеддингу диктора g_i, результат пропускается через многослойный перцеп-

трон, рассчитывающий метрику соответствия. Для превращения этих K чисел в вероятностное распределение используется операция softmax.

Здесь и далее многослойный перцептрон имеет 1 скрытый слой, использует в качестве функции активации ReLU, а также применяет на скрытом слое операцию Dropout с $p = 0.5$.

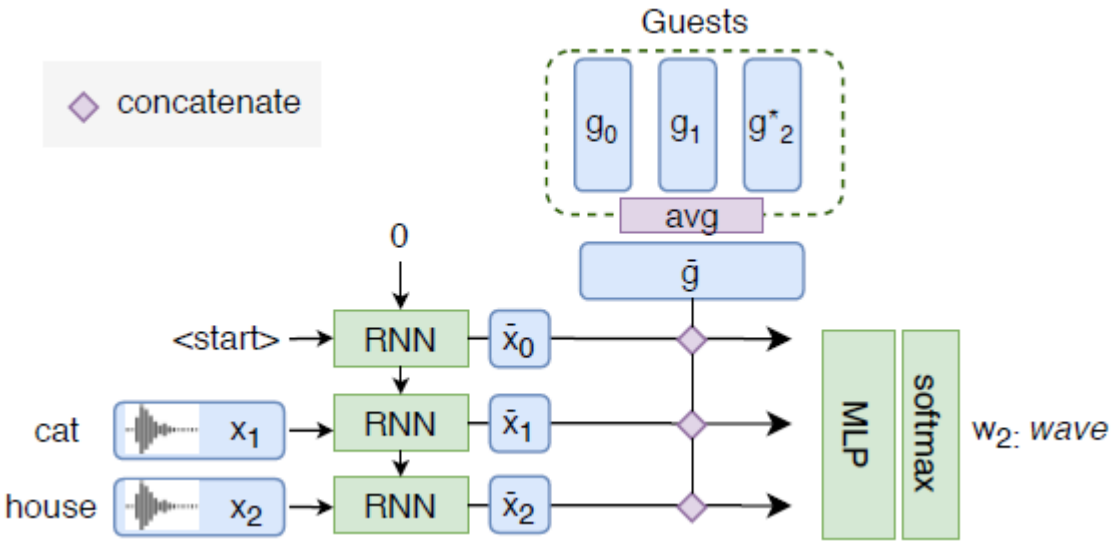

Рис. 3. Архитектура нейросети `Enquirer` [1].

Архитектура нейросети `Enquirer` (рис. 3), выполняющей функцию выбора запрашиваемого слова, тоже относительно проста. В ней для аггрегации информации о запрошенных (и услышанных) ранее словах используется рекуррентная нейронная сеть (*RNN*), если быть точнее — BiLSTM. Далее к последнему скрытому состоянию BiLSTM конкатенируется усреднённый эмбеддинг дикторов, результат пропускается через MLP. Выходная размерность равна V — числу слов в словаре.

1.4. Мотивация

Теперь, когда мы разобрались с устройством предлагаемой системы распознавания диктора, возникает логичный вопрос — есть ли у неё какие-либо преимущества относительно существующих решений? И какую цель преследует добавление модели для выбора запрашиваемых слов?

В оригинальной работе [1] основная идея — адаптация системы под идентифицируемых в данный момент дикторов. Различия в произношении могут объясняться как акцентом человека, так и его индивидуальными особенностями. Поэтому важные признаки будут отличаться от диктора к диктора, соответственно, для наиболее быстрого распознавания будут требоваться различные слова. Таким образом, главное преимущество предлагаемого подхода — распознавание диктора с использованием минимального количества тестовых аудиозаписей речи.

Стоит отметить, что сегодня уже существуют решения[3], позволяющие надёжно верифицировать пользователя по очень коротким аудиозаписям. Проблема этих решений заключается в том, что человек должен произнести некоторые слова из ограниченного списка. Это создаёт риск т. н. спуфинга — злоумышленник может обмануть систему, предоставив ей запись речи верифицируемого человека. Хотя оригинальная имплементация разрабатываемой системы едва ли решает эту проблему — выбор осуществляется из фиксированного списка из 20 слов, нейросеть для выбора запрашиваемых слов может быть изменена таким образом, чтобы исправить этот недочёт (см. главу 3.2).

[3]Здесь сошлёмся на экспертные знания сотрудников лаборатории Huawei CBG AI

1.5. Выводы и результаты по главе

- Задача распознавания диктора (*speaker recognition*) активно изучается в течение многих лет и представляет большой практический интерес.

- В ряде случаев система распознавания диктора может выбирать, какую фразу произнесет пользователь. В таком случае логично использовать алгоритм, позволяющий за счёт выбора фраз сократить количество запрашиваемой речи и/или увеличить точность распознавания.

- В качестве такого алгоритма можно использовать предложенную в [1] нейросетевую модель.

2. Детали реализации и результаты

2.1. Данные для обучения и извлечение эмбеддингов

Здесь мы практически полностью повторяем описанный в [1] подход. Для обучения и тестирования моделей мы использовали фреймворк PyTorch [9], исходный код доступен на платформе GitHub[4]. Единственным (но очень существенным) отличием является использованная размерность эмбеддингов. Перед тем как перейти к обсуждению этого момента, расскажем про исходные данные.

Итак, для обучения и тестирования моделей мы использовали датасет TIMIT [10]. Он составлен из аудиозаписей речи 630 дикторов, говорящих на 8 основных диалектах американского английского языка. Эти дикторы поделены на обучающую (*train*) и тестовую (*test*) выборки, в первую входят 468 дикторов, во вторую — 162. Для обучения нейросетевых моделей мы также создавали валидационную выборку, в которую выделялись 20% дикторов из обучающей.

Каждый из дикторов произносит 10 фонетически насыщенных предложений. При этом 2 из 10 предложений являются общими для всех дикторов[5], остальные 8 уникальны для каждого диктора. Такое разделение позволяет без особых затруднений

[4]https://github.com/vsgolovin/interactive-speaker-recognition

[5]Эти предложения:

- *She had your dark suit in greasy wash water all year.*
- *Don't ask me to carry an oily rag like that.*

подготовить данные, необходимые для описанной в 1.2 игры:

- 2 общих предложения можно использовать для получения аудиозаписей слов. Для этого разделим аудиозаписи этих предложений по временным отметкам, предоставленным создателями датасета. В результате получим 20 аудиозаписей слов[6] для каждого диктора.

- 8 уникальных для каждого диктора предложений можно использовать для получения голосовых подписей — эмбеддингов дикторов — просто при помощи усреднения эмбеддингов аудиозаписей этих предложений.

В качестве векторов признаков использовались эмбеддинги `x-vector` [2]. Весь процесс преобразования аудиозаписей в векторы признаков осуществлялся с помощью библиотеки Kaldi [11]. На первом этапе рассчитывались мел-частотные кепстральные коэффициенты[7] и производилось детектирование голосовой активности (Voice Activity Detection). Полученные векторы признаков поступали на вход предобученной нейронной сети [12]. В качестве эмбеддингов использовались данные со второго 512-мерного слоя.

Здесь, как уже было сказано ранее, мы отступаем от оригинальной работы [1], где использовались 128-мерные эмбеддинги. На это есть две причины. Во-первых, из приведенных в [1] комментариев неочевидно[8], как производилось понижение раз-

[6]Аналогично [1] мы не используем слово *an*.

[7]Параметры аналогичны использованным в [1] и определяются требованиями предобученной модели.

[8]Цитата: *We then process the MFCCs features through a pretrained X-Vector network to obtain a high quality voice embedding of fixed dimension 128, where the X-Vector network is trained on augmented Switchboard, Mixer 6 and NIST SREs.*

мерности. Во-вторых, мотивация такого преобразования тоже неочевидна. Уже первые проведенные нами эксперименты показали, что при использовании 512-мерных эмбеддингов точность идентификации оказывается существенно выше приведенных в [1] значений.

2.2. Обучение `Guesser`

Первой обучается нейронная сеть `Guesser`, выполняющая выбор из K дикторов при помощи T аудиозаписей произнесенных слов. Как уже было сказано ранее, эта нейросеть тренируется в режиме обучения с учителем, дикторы и произносимые слова выбираются случайно, в качестве функции потерь используется кросс-энтропия. Процесс вычисления значения функции потерь для одной игры можно записать следующим образом:

Listing 1. Рассчёт функции потерь `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)
```

Из-за того, что мы увеличили размерность эмбеддингов в 4 раза по сравнению с [1] пропорционально увеличились и размерности слоёв `Guesser`. Из-за этого нам пришлось изменить

гиперпараметры, в частности мы сильно уменьшили темп обучения (*learning rate*).

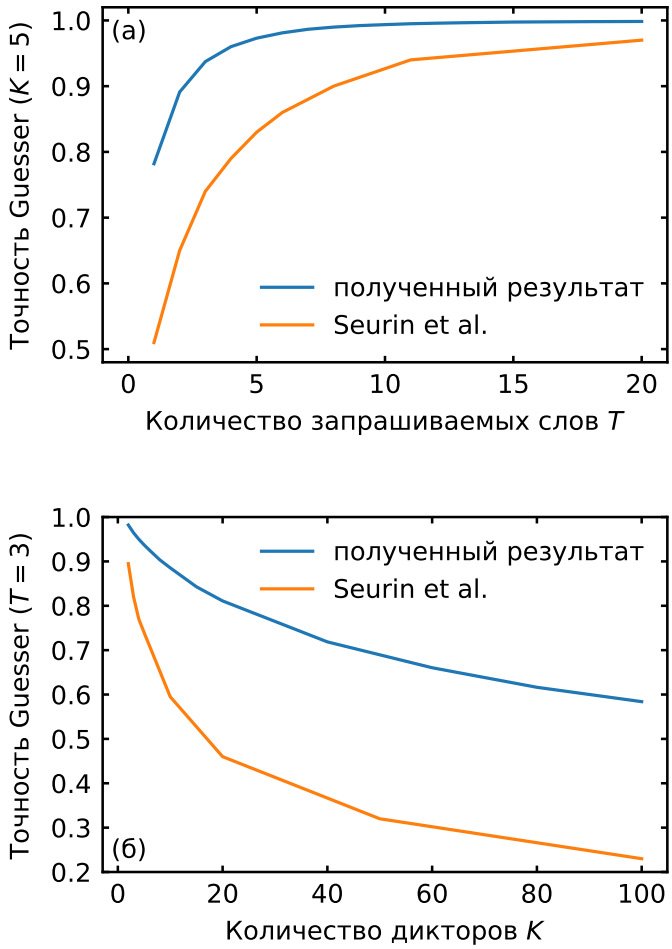

Рис. 4. Зависимость точности `Guesser` обученного нами и авторами [1] от (а) числа запрошенных слов T, (б) числа дикторов K. Модели обучены в режиме $K = 5$, $T = 3$.

Как и в оригинальной статье, для сравнения моделей мы строим графики *word* и *guest sweep*. Т. е. мы обучаем модель в режиме с $K = 5$ дикторами и $T = 3$ запрашиваемыми словами, а затем тестируем её в режимах с отличным числом дикторов или

слов. Здесь и далее, если это не оговорено отдельно, для расчёта точности проводятся 20000 игр среди дикторов из тестовой выборки, эксперименты повторяются по 5 раз с различным `seed` генератора случайных чисел.

По приведенным на рис. 4 результатам видно, что увеличение размерности эмбеддингов существенно улучшает точность идентификации, разница особо велика в режимах с большим числом дикторов K.

2.3. Обучение `Enquirer`

Для обучения `Enquirer` — модели для выбора слов — уже нужна обученная модель `Guesser`. На этом этапе используется обучение с подкреплением, псевдокод для 1 игры приведён ниже.

Как видно из приведенного псевдокода, награда выдается в том случае, когда `Guesser` правильно угадывает диктора. Для обучения мы использовали алгоритм PPO [13] — здесь мы снова повторяем подход авторов [1]. В целом выбор метода выглядит разумным — PPO зарекомендовал себя как простой и универсальный алгоритм, позволяющий достигать хороших результатов. Однако некоторые особенности нашей задачи — дискретное пространство действий, малая длительность эпизодов — выглядят лучше подходящими для off-policy алгоритмов. К сожалению, у нас не нашлось времени, чтобы проверить эту гипотезу.

Listing 2. Интерактивная игра для обучения `Enquirer`

```
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:
        probs[previous_actions] = 0.0
        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
```

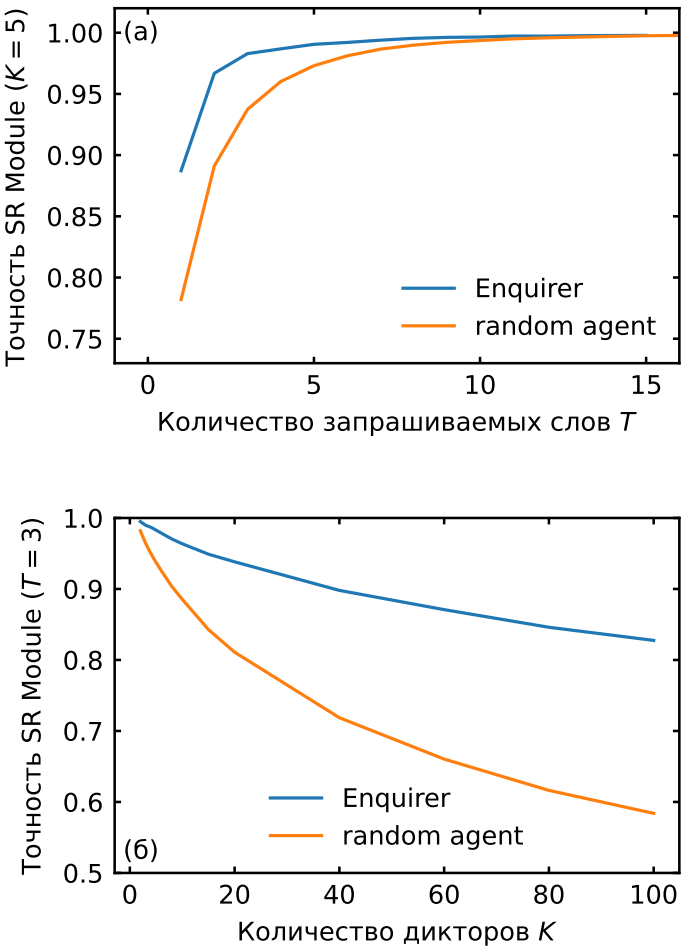

Рис. 5. Зависимость точности SR-систем с различными методами выбора слов — нейросетевым (`Enquirer`) и случайным (random agent) — от (а) числа запрашиваемых слов T, (б) числа дикторов K. Модели обучены в режиме $K = 5$, $T = 3$.

Приведенные на рис. 5 результаты свидетельствуют о том, что нейросеть действительно успешно обучается — точность оказываются заметно выше, чем с случае случайного выбора слов. Как и в случае повышения размерности эмбеддингов, особенно большое различие наблюдается в режимах с большим

числом дикторов.

2.4. Эвристическая модель выбора слов

Очевидно, что агент, выбирающий запрашиваемые слова случайным образом, не является тяжелым противником для нейросетевого агента. Для более трезвой оценки возможностей последнего, логично сравнивать его с каким-то более сложным алгоритмом.

Здесь мы снова немного отходим от оригинальной статьи. И опять основной причиной является тот факт, что в [1] отсутствует точное описание использованного в качестве бейзлайна эвристического алгоритма выбора слов. Из приведенного в работе объяснения[9] общий подход понятен — сэмплирование производится не из всех 20 слов, а из тех, которые в среднем показывают самую высокую точность. При этом остаются непонятными следующие детали:

1. Из скольки слов производится сэмплирование, и меняется ли это число в зависимости от числа запрашиваемых слов?

2. Производится ли сэмплирование равномерно, или вероятность выбрать слово пропорциональна достигаемой при выборе этого слова средней точности?

Именно такие вопросы возникли у нас при создании эвристического агента. Первым же этапом стала оценка слов — расчёт средней точности, которая достигается случайным агентом в

[9]Цитата: *We curated a list of the most discriminant words (words that increase globally the recognition scores) and sample among those instead of the whole list.*

тех играх, когда он выбрал то или иное слово. Для этого мы протестировали `Guesser` в 100000 эпизодов с $K = 5$, $T = 3$, а также случайным выбором слов без повторений (рис. 6). Мы рассчитывали точность для каждого слова, учитывая только те эпизоды, в которых это слово было выбрано. Фактически мы оценивали условную вероятность связки `Guesser`–случайный агент правильно выбрать диктора при условии, что одно слово уже было выбрано.

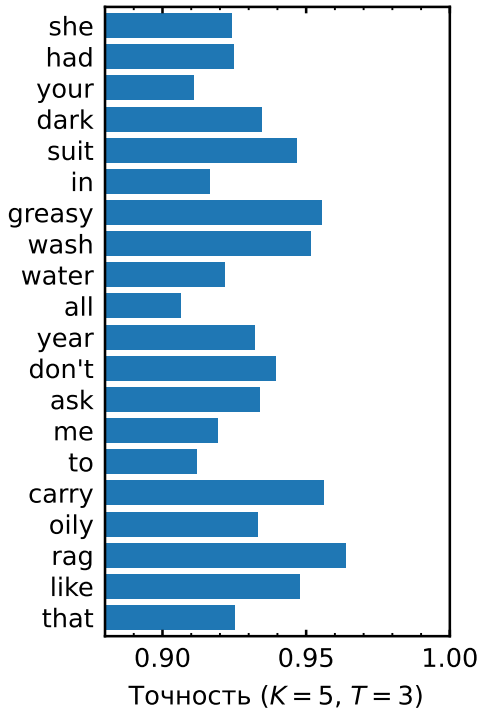

Рис. 6. Средняя точность `Guesser` на валидационной выборке в тех эпизодах, когда соответствующее слово было выбрано (остальные выбирались случайно).

После этого мы стали тестировать различные модификации эвристического агента, отличавшиеся числом использованных

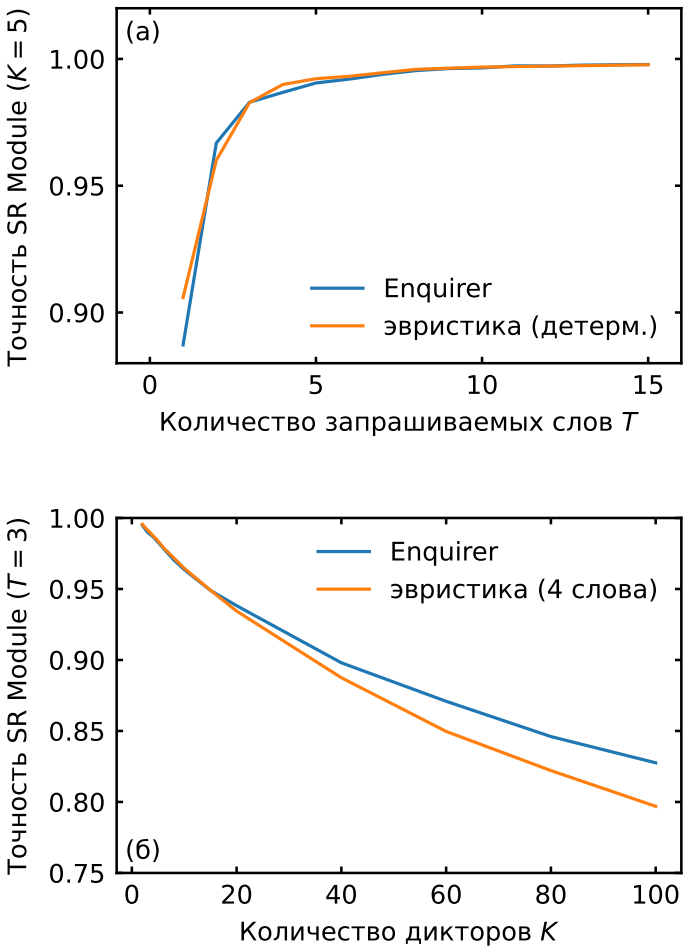

Рис. 7. Зависимость точности SR-систем с различными методами выбора слов — нейросетевым и эвристическим агентами — от (а) числа запрашиваемых слов T, (б) числа дикторов K. Модели обучены в режиме $K = 5$, $T = 3$.

слов и методом сэмплирования. Эксперименты показали, что наилучшие результаты достигаются при использовании “детерминированного” агента, всегда выбирающего одни и те же слова с наибольшей средней точностью. В таком случае говорить о каком-либо сэмплировании неуместно, поэтому такой агент, по

всей видимости, отличается от использованного в оригинальной статье.

В данном случае преимущество `Enquirer` проявляется только в режимах с большим числом дикторов. В стандартном режиме с $K = 5$ дикторами и $T = 3$, в отличие от [1], мы не наблюдаем сколько-нибудь существенной разницы между двумя агентами.

В таком случае возникает резонный вопрос — не сходится ли `Enquirer` к такой же политике, что и эвристический агент? Ответ на этот вопрос — отрицательный, протестированный `Enquirer` в основном выбирает из 5 слов (ещё 2 используются редко), в то время как эвристический агент всегда использует 3 тех же слова. Из этого можно предположить, что `Enquirer` обучен недостаточно хорошо, возможно, другие гиперпараметры или алгоритм обучения позволили бы улучшить результаты.

2.5. Обучение в других режимах

Другой логичный вопрос, возникающий при обсуждении графиков *word* и guest sweep — является ли стандартный режим ($K = 5$ дикторов и $T = 3$ запрашиваемых слова) оптимальным для обучения моделей? Не будут ли результаты лучше, если мы будем обучать и тестировать модели в одном и том же режиме? Мы также провели ряд экспериментов и пришли к следующим выводам:

- Общее правило — более тяжелые режимы позволяют улучшить точность. В первую очередь это касается увеличения числа дикторов, ситуация с уменьшением числа слов ме-

нее однозначная. Пример этого эффекта демонстрирует табл. 1.

- Основные улучшения наблюдаются в работе `Guesser`, в то же время `Enquirer` оказывается нечувствительным к режиму обучения.

- Обучение при $T = 1$ является специфической задачей. Во-первых, модель, обученная в таком режиме, показывает (относительно) хорошие результаты только в нём. Во-вторых, `Enquirer` в целом плохо справляется с этой задачей, часто уступая максимально простой политике, всегда выбирающей одно и то же слово.

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 1. Точность идентификации, $K = 5$ дикторов, $T = 3$ запрашиваемых слова.

2.6. Выводы и результаты по главе

- Предложенный в [1] действительно работает, нейросетевая модель выбора слов действительно позволяет увеличить точность распознавания.

- В оригинальной статье было выполнено понижение размерности эмбеддингов. Как именно и зачем это было сделано — неизвестно. Мы обучили модель на “полноразмерных” 512-мерных эмбеддингов и наши результаты оказались существенно лучше, чем в оригинальной статье.

- Хотя предложенный подход действительно работает, его преимущество над простым эвристическим подходом в среднем оказывается небольшим. Возможно, это можно исправить с помощью оптимизации процедуры обучения.

3. Модификации метода

3.1. От идентификации к верификации

В первых двух главах этого отчёта мы изучали предложенную в [1] систему распознавания диктора. С нашей точки зрения у неё есть серьёзный недостаток — она решает задачу *идентификации*, в то время как интересная нам с практической точки зрения система аутентификации пользователя должна решать задачу *верификации*. Ранее мы сформулировали тезис о том, что это не является большой проблемой, и переход идентификация–верификация можно выполнить без особых проблем. Обсуждению этого вопроса посвящён данный раздел.

Сначала проговорим, как меняется наша задача. Ранее мы должны были выбрать одного из K дикторов, произнесшего T слов, т. е. мы использовали K эмбеддингов дикторов и T эмбеддингов слов. В случае верификации у нас есть только 1 диктор, от нас требуется ответить на вопрос, является ли он человеком, произнесшим услышанную нами речь. Подумаем, какие изменения нам нужно внести в архитектуру использованных нами нейросетей.

В случае `Enquirer` (нейросети для выбора запрашиваемых слов) ответ оказывается предельно простым — нам не нужны никакие изменения. Действительно, сами эмбеддинги диктора на вход этой модели не поступают, используется только их среднее $\hat{g}$, которое в случае верификации будет просто равно эмбеддингу единственного диктора.

Ситуация с `Guesser` лишь немного сложнее. Т. к. его архи-

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 2. Точность верификации, $T = 3$ запрашиваемых слова

тектура позволяет рассматривать игры с произвольным числом дикторов, проблемы возникают только на самом последнем слое, выполняющим операцию softmax. На данном этапе у модели (для каждой игры) есть только одно число, которое фактические является некоторой метрикой соответствия между взвешенной суммой эмбеддингов слов $\hat{x}$ и эмбеддингом диктора g. В таком случае для принятия решения о (не-)соответствии речи и диктора логично применить операцию sigmoid (логистическую функцию), превращающее эту метрику в число от 0 до 1.

Действительно, такое простое преобразование позволяет получить работающую систему верификации диктора. Полученные результаты приведены в табл. 2. Как и в случае идентификации, обучение в более тяжелом режиме (здесь мы можем только сокращать число запрашиваемых слов) позволяет немного улучшить результаты, но при этом преимущество перед простым эвристическим агентом[10] тоже является минимальным.

[10]Градация слов при переходе к верификации практически не меняется.

3.2. `CodebookEnquirer` — гибкая система выбора слов

Перейдём к обсуждению другой проблемы оригинальной модели — наличия фиксированного списка слов. Действительно, в качестве одного из преимуществ разрабатываемой системы мы ранее называли возможность делать разнообразные запросы. Однако используемый до данного момента времени вариант `Enquirer` слабо соответствует этому требованию — он осуществляет выбор из 20 слов. Конечно, этот список может быть и больше, просто для этого потребуется больший объём данных для обучения. Но при добавлении любого слова будет необходимо либо заново обучать `Enquirer`, либо выполнять fine-tuning, что выглядит не самым оптимальным вариантом для готового продукта.

Для решения этой проблемы была разработана архитектура `CodebookEnquirer`. Она представляет собой простую модификацию оригинальной модели:

1. "Голова" модели представляет собой `Enquirer`, в котором число выходов равно размерности эмбеддингов, и к ним не применяется операция softmax. Таким нехитрым способом мы преобразовали выходы модели из вероятностного распределения по словарю в эмбеддинг запрашиваемого слова.

2. Естественно, стоящая перед `CodebookEnquirer` задача никак не поменялась — у нас все ещё существует некоторый конечный набор слов, из которых на каждом шаге игры

нам нужно выбрать одно (или, что лучше, получить распределение). Для этого мы составляем `Codebook` — тензор из эмбеддингов слов, которые рассчитываются как среднее по всем дикторам из обучающей выборки.

3. Наконец, нам нужно как-то сопоставить возвращаемый моделью эмбеддинг с эмбеддингами из `Codebook`. Самый очевидный вариант — просто найти ближайший по L_2-норме. Примерно это мы и делаем, вероятность выбрать i-ое слово из `Codebook` вычисляется по формуле:

$$p_i = \frac{\exp\left(-d_i/T\right)}{\sum_{j=0}^{V} \exp\left(-d_j/T\right)},$$

где d_i — расстояние[11] между выходным эмбеддингом и i-ым вектором из `Codebook`, T — обучаемый параметр модели, V — размер словаря.

В наших экспериментах такая модификация показала практически такие же результаты, что и оригинальная версия `Enquirer`. Далее мы решили проверить, возможно ли изменение набора слов без дообучения модели. Для этого мы обучили `Codebook Enquirer` на половине словаря и протестировали его на другой половине. В таком случае мы наблюдали лишь небольшое падение точности[12], которое, скорее всего, просто объясняется уменьшением размера используемого словаря.

[11] Для численной стабильности мы используем среднеквадратичную ошибку (MSE) вместо L_2-нормы.

[12] Например, точность обученной в режиме $K = 20$, $T = 2$ модели упала с 98.9% до 98.0%.

3.3. Добавление шума

Следующим экспериментом была проверка того, будет ли работать предложенный подход при наличии фонового шума. Для этого мы выбрали 6 аудиозаписей шума из датасета MUSAN [14] и добавили их случайные фрагменты[13] к аудиозаписям слов. Соотношение сигнал / шум было выбрано равным 3 дБ. При обучении и тестировании моделей тип шума выбирался случайно, но он не менялся в течение игры.

Выбор слов	Идентификация	Верификация
случайный	0.887	0.895
`Enquirer`	0.946	0.934
эвристика	0.957	0.938

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

Полученные результаты приведены в табл. 3. Видно, что добавление шума сделало задачу тяжелее, из-за чего точность SR-систем немного упала. Также любопытно, что простой эвристический агент снова не проиграл `Enquirer` и даже оказался немного (на уровне погрешности) лучше. Причина этого стала понятна после измерения средней точности `Guesser` на аудиозаписях зашумлённых слов: выяснилось, что хотя добавление того или иного типа шума влияет на градацию слов (которую использует эвристический агент), этот эффект невелик. Иными

[13] Аудиозаписи специально выбирались таким образом, чтобы их случайные короткие фрагменты отличались слабо.

словами, “хорошие” слова, с помощью которых в среднем достигается самая высокая точность распознавания диктора, остались такими же и при добавлении различных типов фонового шума.

3.4. Альтернативные эмбеддинги

Во всех описанных ранее экспериментах для получения эмбеддингов мы использовали `x-vector` [2], повторяя подход авторов оригинальной статьи. Проблема в том, что выбор таких старых (2017 год) эмбеддингов выглядел немного странным уже на момент написания оригинальной статьи (2020 год).

Поэтому для последнего эксперимента мы проверили, как разработанная SR-модель работает с другими эмбеддингами. Для этого использовалась нейросеть, обученная нашими коллегами из лаборатории Huawei CBG AI на 960 часах аудиозаписей из датасета LibriSpeech[15] и использующая метод контрастного прогнозирующего кодирования[16].

Выбор слов	`x-vector`	CPC
случайный	0.755	0.946
`Enquirer`	0.914	0.990

Таблица 4. Точность идентификации при использовании различных типов эмбеддингов. $K = 20$, $T = 2$.

Пример результатов показан в табл. 4. Общий вывод прост — замена эмбеддингов позволяет существенно повысить точность, как при случайном выборе слов, так и при использовании `Enquirer`.

Заключение

все работает, но хотелось бы большего

Список литературы

[1] M. Seurin, F. Strub, P. Preux, and O. Pietquin, "A machine of few words – interactive speaker recognition with reinforcement learning," 2020.

[2] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-Vectors: Robust DNN embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Apr. 2018.

[3] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," 2019.

[4] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," 2020.

[5] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," 2018.

[6] T. Bäckström, O. Räsänen, A. Zewoudie, P. P. Zarazaga, L. Koivusalo, S. Das, E. G. Mellado, M. B. Mansali, D. Ramos, S. Kadiri, and P. Alku, *Introduction to Speech Processing*, ch. Speaker Recognition and Verification. 2 ed., 2022.

[7] S. Ioffe, "Probabilistic linear discriminant analysis," in

Computer Vision – ECCV 2006, pp. 531–542, Springer Berlin Heidelberg, 2006.

[8] C. Zeng, X. Wang, E. Cooper, X. Miao, and J. Yamagishi, "Attention back-end for automatic speaker verification with multiple enrollment utterances," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, may 2022.

[9] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[10] Garofolo, John S., Lamel, Lori F., Fisher, William M., Pallett, David S., Dahlgren, Nancy L., Zue, Victor, and Fiscus, Jonathan G., "TIMIT acoustic-phonetic continuous speech corpus," 1993.

[11] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Signal Processing Society, Dec. 2011. IEEE Catalog No.: CFP11SRW-USB.

[12] "SRE16 Xvector Model," 2017.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[14] D. Snyder, G. Chen, and D. Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015. arXiv:1510.08484v1.

[15] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, 2015.

[16] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2019.