Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (05.13.11)

УДК 004.032.26
DOI: 10.24160/1993-6982-2021-2-98-107

Применение нейронных сетей для автоматического определения словесных ударений

О.В. Бартеньев

Решена задача автоматического определения словесных ударений. Выделены классы слов с одним и двумя непереходящими ударениями, с переходящими ударениями и без ударений. Ударение определяется в тех словах, где оно не является переходящим. Слова группируются по числу слогов. Каждая группа делится на классы слов с одинаковыми номерами ударных слогов. Таким образом, задача определения ударений, выполняемая нейронными сетями, сводится к классификации слов.
Набор данных (обучающие и проверочные множества) формируется по грамматическому словарю русского языка А.А. Зализняка, содержащему словоформы с расставленными ударениями. Модель слова — список слогов. В наборе данных слоги заменяются числовыми кодами, для получения которых составляются словари слогов. Числовой код слога — его номер в словаре слогов.
Поиск ударений проходит в два этапа. Прежде всего выясняется, обладает ли слово непереходящими ударениями, и затем, в случае положительного результата, оно передается определяющей ударения нейронной сети. Все проектируемые в работе нейронные сети содержат слой Embedding, переводящий скалярные представления слогов слова в векторные. На входе сеть принимает вектор с числовыми кодами слогов слова, а на выходе выдает номер класса, который, в случае одного непереходящего ударения, совпадает с номером ударного слога, а в случае двух непереходящих ударений указывает на номера двух ударных слогов. Вероятность правильного определения одного непереходящего ударения оценивается как 0,9474, а двух — как 0,9759.

Ключевые слова: словесное ударение, слогоделение, классификатор, нейронная сеть, набор данных.

Для цитирования: Бартеньев О.В. Применение нейронных сетей для автоматического определения словесных ударений // Вестник МЭИ. 2021. № 2. С. 98—107. DOI: 10.24160/1993-6982-2021-2-98-107.

Application of Neural Networks for Automatically Detecting Verbal Accents

O.V. Bartenyev

The problem of automatically detecting verbal accents is solved. Word classes with one and two non-transitive accents, with transitive accents, and without accents are identified. An accent is determined in words in which it is not transitive. Words are grouped by the number of syllables. Each group is divided into word classes with the same numbers of accented syllables. Thus, the accents determination problem solved by means of neural networks boils down to word classification. The data array (training and test sets) is formed from A.A. Zaliznyak's Russian language grammatical dictionary, which contains word forms with placed accents. A word model comprises a list of syllables. In the data array, syllables are replaced by their numerical codes, for which syllable dictionaries are compiled. The numerical code of a syllable is its number in the syllable dictionary. The accents are searched in two stages. First, it is found out whether the word has non-transitive accents, and if yes, the word is transferred to the neural network that determines the accents. All neural networks designed in this study contain an Embedding layer which translates scalar representations of word syllables into vector ones. At its input, the neural network receives a vector with the numerical codes of word syllables, and at the output it yields the word class number, which in the case of one non-transitive accent coincides with the number of the accented syllable, and in the case of two non-transitive accents indicates the numbers of two accented syllables. The probabilities of correctly determining one and two non-transitive accents are estimated at 0.9474 and 0.9759, respectively.

Key words: word accent, syllable division, classifier, neural network, data array.

For citation: Bartenyev O.V. Application of Neural Networks for Automatically Detecting Verbal Accents. Bulletin of MPEI. 2021;2: 98—107. (in Russian). DOI: 10.24160/1993-6982-2021-2-98-107.

Введение

Автоматическое определение словесных ударений — одна из задач машинной обработки текстов. Результаты определения могут быть использованы автоматом, переводящим текст в речь, или программой, сочиняющей стихотворения с заданным стихотворным размером, например, ямбом или хореем.

Определить ударный слог в заданном слове можно, обратившись к грамматическому словарю [1], электронная версия которого представлена в [2]. Всего в нем около 85 000 слов (основ) и около 2 400 000 словоформ (для краткости вместо «словоформа» далее употребляется «слово»).

Для автоматизации поиска ударений по словарю [2] его следует перенести в базу данных (БД) и обращаться к ней для получения результата, который затем интерпретировать. В интернете существуют сервисы расстановки ударений в заданном слове или тексте, основанные на данном подходе [3]. Если вариантов расстановки ударений несколько, то будут показаны все: *На го́ре|горе́ стоя́л за́мок|замо́к; на врата́х его́ за́мок|замо́к*. Ударение не будет определено вовсе, если рассматриваемое слово отсутствует в БД. Например, в предложении *татаканье веялочных бараба́нов* (М.А. Шолохов) сервис [3] поставит ударение только в одном слове.

Иные известные подходы к решению рассматриваемой задачи заключаются в создании эвристических и статистических правил расстановки ударений [4, 5]. Так, в [5] при определении ударения слово прежде ищется в словаре [2] (полном или сокращенном). Если оно не найдено, то применяются эвристические правила. Если ни одно из правил не выполняется, то используется статистическое правило. При тестировании с сокращенным словарем 30,9% слов тестового корпуса были найдены в словаре, условиям применения эвристических правил удовлетворили 31,9% слов, в оставшихся 37,2% слов применены статистические правила. Ошибка определения ударения — 4,5%. Сведений о точности применения эвристических и статистических правил без использования словаря в [5] нет.

В настоящей работе определение ударений выполнено нейронными сетями (НС).

Русское словесное ударение

В [6, 7] дано следующее описание русского словесного ударения.

В русском языке ударение является разноместным, т. е. может встречаться на любом слоге, например, *ко́мната*, *кора́блик*, *сарафа́н*.

Ударение является подвижным. Оно может переходить с одного слога на другой: *окно́ — о́кна*, *и́глы — иглы́*, *о́блака — облака́*.

Ударение позволяет различать слова: *за́мок* и *замо́к*, *жа́ркое* (лето) и *жарко́е* (блюдо), *е́док* и *едо́к*.

В некоторых словах, преимущественно образованных из двух и более основ, может быть несколько ударений — главное и побочные: *а́виабензи́н*, *ле́созаво́д*, *о́бме́ршая*.

Служебные части речи (предлоги, союзы, частицы) обычно не имеют ударений: *под окно́м*, но могут и иметь: *Что́ счастие? Короткий миг и тесный... И за́ сердце хватающий полет*. (А.А. Блок.)

Место ударения с течением времени меняется: *Ах, новость, да какая! Музы́ка будет полковая!* (А.С. Пушкин.)

Описание русского словесного ударения позволяет говорить о четырех классах слов:

C_0 — с переходящими ударениями;

C_1 — с одним непереходящим ударением;

C_2 — с несколькими непереходящими ударениями;

C_3 — служебные части речи, употребляющиеся без ударения (состав данного класса будет уточнен).

Классы C_0 — C_2 формируются по словарю [2], C_3 — по электронной версии морфологического словаря русского языка [8], содержащей около 4 200 000 словоформ (в [2] служебные части речи не представлены). Поскольку в [2] нет слов с числом ударений 3 и более, то в C_2 попадут слова только с двумя непереходящими ударениями. Очевидно, число слогов в словах C_0 и C_2 — два и более, а в C_1 и C_3 имеются слова с одним слогом.

Замечание. Число слогов в словах словаря [2] варьируется от 1 до 12.

Место ударения или его отсутствие в C_0 и C_3 определяется только при наличии контекста. В C_1, C_2 контекст для поиска места ударения не нужен. Заметим, что методы, описанные в [5], автоматически определяют ударение в словах C_1.

Формулирование решаемых задач

Цель работы — создание нейронных сетей (НС), устанавливающих непереходящие словесные ударения. Поиск непереходящих ударений не требует привлечения контекста. В качестве источника сведений об ударениях взят словарь [2], по которому составлены наборы данных для обучения и проверки НС.

Формально задача определения ударений заключается в поиске номеров всех ударных слогов в исследуемом слове.

Перед началом поиска следует определить класс C_m ($0 \le m \le 3$) анализируемого слова (выполнить его начальную классификацию). Класс C_3 содержит части речи, употребляемые без ударений. Принадлежность слова C_3 устанавливается в результате поиска слова в этом классе. Процедура нетрудозатратна ввиду малой мощности C_3. Принадлежность слова другим классам выявляет НС.

Поиск номеров ударных слогов продолжается, если слово входит в C_1 или C_2. Заметим, что результат очевиден, если в слове один слог, а также если в слове класса C_2 два слога.

Сведем определение ударений к классификации. Разделим слова класса C_m ($m = 1, 2$) на группы, включая

в группу $G_{ns}^{C_m}$ ($m \le ns \le 12$, ns — число слогов в слове) слова с ns слогами (в [2] нет слов с числом слогов более 12).

В группе $G_{ns}^{C_1}$ выделим ns классов, собрав в класс i (i = 1, ns) слова, в которых ударение падает на слог с номером i.

В C_2 определим два ударных слога. Составим список L возможных пар ударных слогов и отберем в класс k слова, в которых пара ударных слогов имеет в списке L номер k.

При таком подходе установить ударения — это найти номер класса слова. В C_1 номер класса совпадает с номером ударного слога, а в C_2 он указывает на пару ударных слогов.

Хорошими классификаторами являются НС. Поскольку в группах слов разное число классов, то для каждой группы потребуется своя НС. В C_1 нужно 11 НС (в словах с одним слогом позиция ударения очевидна), а в C_2 — 10 (в словах с двумя слогами позиции двух ударений также очевидны).

Решение указанных задач требует создания НС трех типов: начального классификатора слова (НКС) и определителей одного (ОУ1) и двух (ОУ2) непереходящих ударений.

Обучение НС проводится на данных, формируемых по словарю [2]. Оценка качества обучения выполняется по критерию точности классификации.

Задача определения номеров ударных слогов предполагает использование представления слова в виде слогов. Поскольку НС оперируют числовыми данными, то слог должен быть представлен числом. В качестве такового используется код слога — его номер в словаре слогов. НС, получив на входе вектор кодов слогов слова, выдает на выходе номер его класса.

ОУ1 и ОУ2 создаются для каждой группы слов, поэтому все векторы на входе указанных НС имеют одинаковый размер (в группе имеются слова с равным числом слогов). В НКС также следует обеспечить равенство размеров входных векторов. Это достигается за счет деления множества V (V — класс, объединяющий C_0, C_1, C_2) начально классифицируемых слов на группы G_{ns}^{V} ($1 \le ns \le 12$) с равным числом слогов в каждой. Поскольку позиция ударения в словах с одним слогом очевидна, то требуется создание 11 НКС.

Схематично введенные классы и группы изображены на рис. 1.

Создание классификаторов станет возможным, если решены задачи:

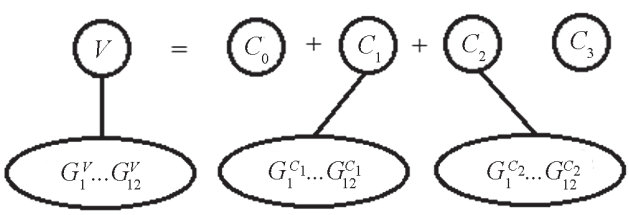


Рис. 1. Введенные классы и группы

● формирования классов C_0 — C_3 и групп слов класса t с ns слогами G_{ns}^{t} ($t \in [V, C_1, C_2]$);

● деления слова на слоги;

● подготовки обучающих и оценочных данных для каждого классификатора.

После этого надлежит создать и обучить классификаторы и оценить их точность.

Решение этих задач проходит с помощью приведенной в [9] программы, реализованной на языке программирования Python.

Формирование классов и групп слов

Классы C_0 — C_2 формируются по словарю [2], представленному в виде связанных html-страниц. В результате их чтения создаются текстовые файлы: слова, начинающиеся с одной буквы, записываются в отдельный файл в следующем виде (рис. 2):

з′амок
з′амки
з′амка
з′амков
з′амку

Рис. 2. Фрагмент файла, созданного по словарю [2]

Ударение обозначено апострофом. В сформированных файлах 2 381 715 слов.

В примере рис. 2 показана малая часть форм слова *замок*. Всего в [2] описано 24 формы этого слова: *за́мок*; *за́мки*; *за́мка*; *за́мков*; *за́мку*; *за́мкам*; *за́мок*; *за́мки*; *за́мком*; *за́мками*; ...; *замка́ми*; *замке́*; *замка́х*. Среди них имеются омографы — слова с одинаковым написанием. Некоторые из них обладают вдобавок одинаковым ударением. Например, *замо́к* приведен в [2] трижды. В задаче определения ударений подобные повторы избыточны, поэтому на следующем этапе подготовки данных взамен нескольких слов с одинаковыми написанием и ударением оставляется одно. После этой операции в файлах остается 1 401 759 слов. Обозначим это множество слов как V и поделим его на классы C_0, C_1 и C_2 (служебные части речи в [2] не представлены).

Класс C_0, содержащий слова с переходящими ударениями, формируется следующим образом: в написании слов V удаляется знак ударения — апостроф и затем формируется таблица с полями Слово и Количество — число присутствий слова в V (табл. 1).

Таблица 1

Фрагмент таблицы, формируемой при создании C_0

Слово	Количество	Варианты ударений
автопоезд	1	а́втопо́езд
автопоезда	2	автопоезда́, а́втопо́езда
автопоездов	1	автопоездо́в

Все слова, для которых в столбце «Количество» указано значение 2 или более, переносятся в C_0. Всего в C_0 окажется 15 036 слов. Преимущественно каждое слово, попавшее в C_0, повторяется в исходных файлах два раза, их число — 14 895. Однако есть слова с числом повторов 3, например: *вертела́*, *ве́ртела*, *верте́ла*. В C_0 с каждым словом записываются варианты простановки ударений и число повторов слова в V, например:

аминокисл'оты; *аминокислот'ы* — 2 аминокислоты;

в'ыходите; *вых'одите*; *выход'ите* — 3 выходите.

Оставшиеся в V слова распределяются по классам C_1 и C_2: в C_1 попадают слова с одним ударением; в C_2 — с двумя.

Дополнительно в C_1 с каждым словом сохраняется информация о номере ударного слога, а в C_2 — о номерах двух ударных слогов, например:

C_1: *азбука* — 1; *баклуши* — 2; *абонент* — 3;

C_2: *артшкола* — 1,2; *автоклуб* — 1,3; *диетврач* — 2,3.

Класс C_3 создается на основе служебных частей речи, представленных в [8]. Словарь содержит 411 предлогов, 322 союза и 488 частиц. В общем случае служебная часть речи может быть составной: *во славу* — предлог; *до тех пор пока* — союз; *а если бы* — частица. При формировании C_3 такие части речи делятся на составляющие их компоненты. Компонент добавляется в C_3, если он не найден в C_0 — C_2. По тем же принципам в C_3 добавляются и несоставные служебные части речи.

Мощности сформированных классов:

C_0 — 15 036 слов с переходящим ударением;

C_1 — 1 323 899 слов с одним непереходящим ударением;

C_2 — 45 651 слов с двумя непереходящими ударениями;

C_3 — 286 служебных частей речи или их компонентов.

Слова, входящие в C_0 и C_3, из процесса определения ударений исключаются.

Классы V, C_1 и C_2 делятся на группы G^t_{ns} ($t \in [V, C_1, C_2]$): в группу G^t_{ns} помещаются слова с *ns* слогами. Число слогов в слове равно числу гласных [6], поэтому для отнесения слова к группе достаточно вычислить число его гласных. Размеры групп даны в табл. 2.

По словам каждой группы создается набор данных для обучения и проверки НС, классифицирующей слова группы.

Замечание. Как следует из приведенных данных, в [2] в C_1 и C_2 входит 98,91% слов — $(|C_1| + |C_2|)/|V|$.

Модель слова

В задачах начальной классификации слов и определения ударений слово представляется в виде списка слогов. На вход НС поступает слово в виде одномерного массива (вектора) кодов слогов:

Слово: *биохимия*.

Число слогов: *ns* = 5.

Слоги: [«*би*», «*о*», «*хи*», «*ми*», «*я*»].

Коды слогов: (99, 3, 144, 6, 8).

Таблица 2

Размеры групп G^t_{ns} ($t \in [V, C_1, C_2]$)

ns	$\lvert G^V_{ns}\rvert$	$\lvert G^{C_1}_{ns}\rvert$	$\lvert G^{C_2}_{ns}\rvert$
1	3 589	3 515	—
2	64 703	61 611	88
3	262 631	256 333	1 306
4	404 185	394 888	4 519
5	336 803	324 692	10 571
6	187 117	173 701	12 899
7	84 492	74 749	9 638
8	30 964	26 094	4 844
9	7 968	6 535	1 433
10	1 661	1 385	276
11	376	307	69
12	97	89	8
Всего:	1 384 586	1 323 899	45 651

Коды слогов берутся из словаря слогов группы, которой принадлежит слово. Код слога — его номер в словаре слогов (нумерация начинается с 1). Так, словарь слогов группы $G^{C_1}_5$, в которую входит слово *биохи́мия*, содержит 5 127 слогов; слог «*би*» стоит на 99-й строке словаря (словари упорядочены по частоте применения слогов; в каждой строке — один слог).

Все классификаторы содержат обучаемый слой Embedding [10], преобразующий код слога в вектор заданного размера. Например, на входе Embedding в НС группы $G^{C_1}_5$ вектор кодов слогов размера 5, а на выходе — массив формы (5, 128), в котором каждый слог слова представлен вещественным вектором размера 128. Отображение кода слога в вектор позволяет существенно повысить точность классификации. Размер 128 подобран экспериментально: при его снижении наблюдается падение точности классификации, при увеличении — растет число обучаемых параметров (весов) НС и, следовательно, время обучения, а существенного повышения точности классификации не наблюдается. Векторное представление слога изначально устанавливается случайным образом, а затем уточняется в процессе обучения НС.

Замечания.

1. В качестве альтернативной взята модель слога в виде вектора размера *n_max* из номеров букв, составляющих слог, а слова — в виде массива таких векторов. Например, *при n_max* =6:

Слово: *кон-церт*.

Модель слова: ((12, 16, 15, 0, 0, 0), (24, 6, 18, 20, 0, 0)).

Нули появляются в конце вектора, если число букв в слоге меньше *n_max*. Точность определения ударений с подобными моделями слога и слова незначительна — чуть более 70%.

2. При создании единой НС для всех рассмотренных задач классификации слово на входе НС (слоя Embedding) следует представить в виде вектора размера 12 (максимальное число слогов в словах словаря [2]). Это означает, что для слов с числом слогов менее 12 размер вектора с кодами слогов слова следует увеличить до 12, используя в качестве заполнителя 0, например:

Слово: *бездна*.

Модель слова: (51, 3874, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0).

Число нейронов на выходе такой НС должно быть равно числу возможных классов: $3 + 12 + C_2^{12}$, где 3 — число классов в НКС; два последующих слагаемых — соответственно максимально возможное число классов в ОУ1 и ОУ2.

Данная модель избыточна и на входе и на выходе НС. Так, при определении ударения в слове *бездна* группы $G_2^{C_1}$ входной вектор содержит 10 лишних элементов (нулей), а на выходе НС достаточно иметь 2 нейрона (число классов равно числу слогов). Избыточность ведет к увеличению времени обучения и низкому качеству классификации (менее 80%).

Использование модели слова в виде вектора кода слогов требует решения двух следующих задач:

- деления слова на слоги;
- формирования словарей слогов всех групп слов.

Алгоритм слогоделения

Приводимый алгоритм выполняет слогоделение в соответствии с изложенными в [6, 7] правилами:

- сочетание шумных согласных между гласными отходит к последующему слогу: *про-стой*, *зве-зда*, *ло-дка*;
- сочетание шумного согласного с сонорным между гласными отходит к последующему слогу: *до-бро*, *ве-сна*, *ве-сло*, *до-гма*;
- сочетание сонорных согласных между гласными отходит к последующему слогу: *ко-рма*, *то-мный*;
- сочетание сонорного согласного с шумным между гласными имеет слогораздел внутри этого сочетания: *пар-та*, *кол-ба*, *лом-кий*, *брон-за*;
- сочетание «*й*» с шумным или сонорным согласными имеет слогораздел внутри этого сочетания: *лей-ка*, *вой-дем*, *тай-на*, *кай-ма*, *сай-ра*;
- буквы «ъ» и «ь» от предшествующих букв не отделяются.

Перечень сонорных и шумных согласных взят из [7]:

- согласные сонорные *й*, *л*, *м*, *н*, *р* образуются с помощью голоса и незначительного шума;
- согласные шумные звонкие *б*, *в*, *г*, *д*, *ж*, *з* создаются с помощью шума в сопровождении голоса;
- согласные шумные глухие *п*, *ф*, *к*, т, *ш*, *с*, *х*, *ц*, *ч*, *щ* получаются с помощью шума без участия голоса.

Алгоритм слогоделения последовательно формирует слоги, начиная с последней буквы поступившего на вход алгоритма слова, опирающегося на известный факт [6]: число слогов в слове равно числу его гласных. Интерес, следовательно, представляют слова с двумя и более гласными. Если начать анализ слова с последней буквы, то в текущий слог следует включить первую встретившуюся гласную букву и все последующие согласные. Судьба согласных перед гласной определяется по приведенным правилам: они либо отходят к формируемому слогу, либо к следующему. После завершения создания слога слогоделение повторяется с оставшейся частью слова, если в ней менее двух гласных, или завершается — в противном случае.

Пример. При слогоделении слова *концерт* в формируемый слог последовательно включаются согласные *т*, *р* и гласная *е* (слогоделение начинается с последней буквы). Буква *ц*, обнаруженная перед *е*, также окажется в формируемом слоге, а вот *н* перейдет в другой слог, поскольку алгоритм обнаруживает сочетание сонорного согласного *н* с шумным *ц*, имеющее внутренний слогораздел. Таким образом, алгоритм фиксирует слог *церт*. В оставшейся части слова одна гласная, что позволяет завершить слогоделение, выдав в качестве результата два слога *кон* и *церт*.

Алгоритм слогоделения.

Входные данные:

word — слово с числом гласных 2 или более;
vowels = "а,е,ё,и,о,у,ы,э,ю,я" — гласные;
sonor = "й,л,м,н,р" — сонорные согласные;
noise = "б,в,г,д,ж,з,п,ф,к,т,ш,с,х,ц,ч,щ" — шумные согласные (звонкие и глухие);
no_sound = "ъ,ь" — буквы, не образующие звуки.

Выходные данные:

word_syls — список слогов слова *word*.

```
1. Начало.
2. Вычислить n_vowels. // Число гласных в word
3. ns = 0 // Число найденных слогов
4. n_let = len(word) // Число непросмотренных букв слова
5. word_syls = [] // Список слогов
6. n_let = n_let – 1
// Берем в формируемый слог последнюю букву слова
7. syl = word[n_let]
// Флаг наличия гласной в формируемом слоге
8. has_vowel = syl В vowels
9. Пока ns < n_vowels - 1:
    n_let = n_let - 1
    let_cur = word[n_let] // Текущая буква слова
    Если let_cur В vowels:
      Если has_vowel:
        // В слоге уже имеется гласная, нужно начать новый слог
        Добавить syl В начало word_syls.
        ns = ns + 1
        Если ns == n_vowels - 1: Выход Из Цикла
        // Начинаем формировать новый слог
        syl = let_cur
      Иначе:
        // Включаем гласную букву в слог
        syl = let_cur + syl
        has_vowel = True
      Иначе:
        // Предшествующая буква слова
```

```
        let_0 = word[n_let + 1]
        Если has_vowel:
          Если let_cur В no_sound:
            // Буква 'ъ' или 'ь'; нужно начать новый слог
            Добавить syl В Начало word_syls.
            ns = ns + 1
            has_vowel = False
          ИначеЕсли let_cur В sonor И let_0 В noise:
            // Сочетание сонорного и шумного согласных: кол-ба
            Добавить syl В Начало word_syls.
            ns = ns + 1
            Если ns == n_vowels – 1: Выход Из Цикла
            has_vowel = False
          ИначеЕсли let_cur == 'й':
            // Сочетание 'й' с шумным или сонорным согласным: тай-на
            Добавить syl В Начало word_syls.
            ns = ns + 1
            Если ns == n_vowels - 1: Выход Из Цикла
            has_vowel = False
          ИначеЕсли let_0 В vowels:
            // Согласная перед гласной: зве-зда
            syl = let_cur + syl
          ИначеЕсли let_cur В noise И let_0 В noise:
            // Сочетание шумных согласных: зве-зда
            syl = let_cur + syl
          ИначеЕсли let_cur В noise И let_0 В sonor:
            // Сочетание шумного и сонорного согласных: до-бро
            syl = let_cur + syl
          ИначеЕсли let_cur В sonor И let_0 В sonor:
            // Сочетание сонорных согласных: то-мный
            syl = let_cur + syl
        Иначе:
          // Два согласных в конце слога: кон-церт
          syl = let_cur + syl
10. syl = word[:n_let + 1] // Первый слог слова
11. Добавить syl В Начало word_syls.
12. Останов.
```

Замечание. Дефис, если он присутствует в слове, алгоритм оставляет в следующим за ним слоге, например:

word = "темно-оранжев";
word_syls = ["те", "мно", "-о", "ран", "жев"].

С использованием алгоритма слогоделения составляются словари слогов S^t_{ns} для групп G^t_{ns} ($t \in [V, C_1, C_2]$). Размеры словарей приведены в табл. 3.

Таблица 3

Размеры словарей слогов S^t_{ns} ($t \in [V, C_1, C_2]$)

ns	$\lvert S^V_{ns}\rvert$	$\lvert S^{C_1}_{ns}\rvert$	$\lvert S^{C_2}_{ns}\rvert$
1	3 672	3 515	—
2	10 274	10 110	145
3	9 603	9 433	989
4	7 393	7 117	1 666
5	5 127	4 826	1 834
6	3 399	3 108	1 550
7	2 211	1 948	1 174
8	1 345	1 138	775
9	688	546	419
10	306	232	201
11	152	120	83
12	54	46	19

Алгоритм определения ударений

Алгоритм устанавливает порядок использования классификаторов при определении ударения в исследуемом слове. Всего имеется 32 классификатора: 11 НКС, 11 ОУ1 и 10 ОУ2. Алгоритм получает на вход слово и, если число гласных *ns* находится в допустимом диапазоне ($0 < ns \le 12$), и слово не является служебной частью речи (класс C_3), а $ns > 1$, выбирает, зная число слогов в слове, НКС и определяет класс C_m ($m = 0, 2$) слова. Если слово принадлежит C_1 или C_2, алгоритм, опять же по числу слогов, выбирает подходящий определитель ударения, находящий номер ударного слога в случае C_1 или номера двух ударных слогов при C_2. Ударение считается неопределенным, если слово принадлежит C_0 или C_3.

Алгоритм определения ударений.

Входные данные:
word — исследуемое слово.
Выходные данные:
acc — номер ударного слога, если *word* $\in C_1$, список номеров двух ударных слогов, если word $\in C_2$, или 0 — в противном случае.
word_class — класс слова.
Классификаторы:
НКС, ОУ1, ОУ2.

```
1. Начало.
// Число слогов в слове равно числу его гласных
2. ns = число_гласных_в(word)
3. Если ns = 0 или ns > 12:
        word_class = None
        acc = 0
        перейти к п. 8
// Принадлежность word классу C3 устанавливается
// по результатам поиска word в C3
3. Если word ∈ C3:
        word_class = C3
        acc = 0
        перейти к п. 8.
4. Если ns = 1:
        word_class = C1
        acc = 1
        перейти к п. 8.
5. классфикатор = НКС[ns] // Выбор классификатора
6. word_class = классфикатор (word) // Прогноз классификатора
7. Если word_class = C0:
        acc = 0
        перейти к п. 8.
    ИначеЕсли ns = 2 и word_class = C2:
        acc = [1, 2] // Два ударных слога в слове с ns = 2
        перейти к п. 8.
```

ИначеЕсли word_class = C_1:
классификатор = ОУ1[ns]
// Номер ударного слога
acc = *классификатор*(word)
Иначе: // word_class = C_2
классификатор = ОУ2[ns]
// Список с номерами двух ударных слогов
acc = *классификатор*(word)
8. Вернуть word_class и acc в качестве результата.
9. Останов.

Замечание. С целью снижения числа НС рассматривался вариант, в котором определение класса слова и непереходящего ударения поручались одному классификатору, выход которого интерпретировался следующим образом:

0 — слово входит в C_0;

i, $1 \le i \le ns$ — слово с ударным слогом i принадлежит C_1;

$ns + 1$ — слово входит в C_2.

В таком варианте общее число классификаторов снижается до 21, но одновременно в среднем на 1,5% падает точность классификации.

Подготовка наборов данных

Наборы данных формируются по группам слов G_{ns}^{t} ($t \in [V, C_1, C_2]$) . Число наборов данных равно числу обучаемых НС. Набор данных содержит собственно данные и их метки. Экземпляр данных — слово в виде вектора с числовыми кодами слогов.

В случае НКС метка слова *word* — это m, если $word \in C_m$ ($0 \le m \le 2$).

Пример. Слово с переходящим ударением *вертела* получит метку 0; *ско́рости* — 1; *дочу́рка* — 1; *абажу́р* — метку 1; *дие́твра́ч* — метку 2.

В случае ОУ1 метка слова *word* — i, если ударение падает на слог i.

Пример. Слово *ско́рости* получит метку 1; *дочу́рка* — 2; *абажу́р* — 3.

В случае ОУ2 для установки меток создается список возможных пар ударных слогов. Метка экземпляра данных — номер пары в этом списке.

Пример. В группе $G_3^{C_2}$ имеем следующий список пар номеров ударных слогов: [«1,2», «1,3», «2,3»], поэтому метками слова *а́втоклу́б* станет число 2, а *дие́твра́ч* — число 3.

Наборы данных разбиваются на обучающее и проверочное множества, используемые для обучения НС и оценки его качества.

Структуры классификаторов

НС реализованы на языке программирования Python с использованием библиотеки Keras [11], позволяющей в компактном виде (набором слоев) задавать модель НС и запускать процесс ее обучения в результате обращения к методу *fit*, одним из параметров которого является обучающее множество данных. В Keras слой НС создается в результате вызова конструктора (метода) соответствующего класса. В процессе обучения с использованием выбранного алгоритма оптимизации выполняется поиск значений весов НС, обеспечивающих минимум заданной в модели НС функции потерь.

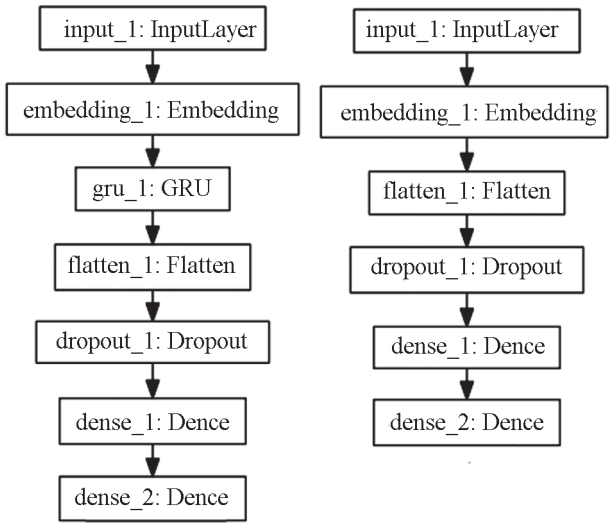
Все проектируемые в работе НС содержат слой Embedding, отображающий скалярные числовые входные данные в векторные. В рассматриваемой задаче он формирует векторные представления слогов слова, поступивших на вход слоя. Размер вектора, представляющего слог на выходе слоя, выбирается из следующего списка: [32, 64, 128], где пробуется каждое значение. Итоговая модель НС получается со значением, обеспечившим наибольшую точность классификации.

Состав последующих слоев зависит от размера словаря слогов S_{ns}^{t} группы слов, с которой работает НС. Если $|S_{ns}^{t}| \ge 2000$, то используется НС с рекуррентным слоем GRU [12] и последующими полносвязными слоями Dense, в противном случае рекуррентный слой НС отсутствует. Структуры НС приведены на рис. 3, *а*, *б*.

Структура с рекуррентным слоем (см. рис. 3, *а*) широко используется для классификации текстовых данных [13]. Исключение этого слоя (см. рис. 3, *б*) при работе с группами, в которых $|S_{ns}^{t}| < 2000$, упрощает НС и не приводит к потере качества классификации.

Слой Flatten [12] преобразует двумерные выходы слоев Embedding и GRU в одномерные, что необходимо для слоя Dense. Слой Dropout [12] препятствует переобучению НС. Слои Flatten и Dropout не имеют обучаемых параметров (весов).

Последний, классифицирующий, слой Dense имеет функцию активации *softmax*, а предшествующий ему слой — ReLU. При обучении НС функция потерь *mse*



а *б*

Рис. 3. Структуры НС для групп слов с разными размерами словаря слогов:

а — $|S_{ns}^{t}| \ge 2000$; *б* — $|S_{ns}^{t}| < 2000$

Таблица 4

Результаты обучения классификаторов

ns	acc_v^F, %	acc_1^F, %	acc_2^F, %	val_acc_v, %	val_acc_1, %	val_acc_2, %
2	99,04	98,29	-	94,41	86,40	—
3	99,44	97,52	100	98,72	94,30	92,37
4	99,50	97,86	100	99,14	96,44	95,35
5	99,55	98,70	100	99,26	97,79	97,35
6	99,61	99,51	100	99,22	98,95	97,75
7	99,99	99,68	100	99,04	99,06	97,92
8	100	100	100	99,00	99,31	97,31
9	100	100	100	98,49	98,78	93,05
10	100	100	100	100	100	89,28
11	100	100	100	100	100	100
12	100	100	100	100	100	100

П р и м е ч а н и е: acc_v^F, val_acc_v — точности НКС на полном наборе данных и проверочном множестве; acc_m^F, val_acc_m — точности ОУm (m = 1, 2) на полном наборе данных и на проверочном множестве.

(среднеквадратическая ошибка) минимизируется с использованием оптимизатора *Adam*.

Программы подготовки наборов данных, создания и обучения моделей НС и предсказания номеров ударных слогов доступны в [9].

Замечание. По такому же принципу обучались НС, в которых рекуррентный слой GRU заменен сверточным слоем Conv1D. Повышение качества классификации при такой замене не выявлено.

Результаты обучения классификаторов

Каждая НС обучается дважды:

- на полном наборе данных;
- наборе данных, разделенном на обучающее и проверочное множества.

Результаты обучения даны в табл. 4;

Показатели acc^F, val_acc характеризуют точности автоматического определения ударения в словах, имеющихся и отсутствующих в словаре [2].

Наибольшая значимость у результатов для часто употребляемых слов. Выделим их, обратившись к частотному словарю русского языка [14], содержащему частоты употребления лемм (исходных форм слов) в текстах Национального корпуса русского языка [15]. Сформируем в словаре [14] группы лемм, объединив в группу L_{ns} леммы с числом слогов ns (ns = 2, 12), и продемонстрируем диаграмму вероятностей принадлежности случайно выбранной леммы группе L_{ns} (рис. 4).

Вероятность принадлежности случайной леммы группе L_{ns} рассчитана следующим образом.

$$F_{ns} = \Sigma f_{ns} / \Sigma f;$$

$$P_{ns} = F_s \Big/ \sum_{ns=2}^{12} F_{ns}, \qquad (1)$$

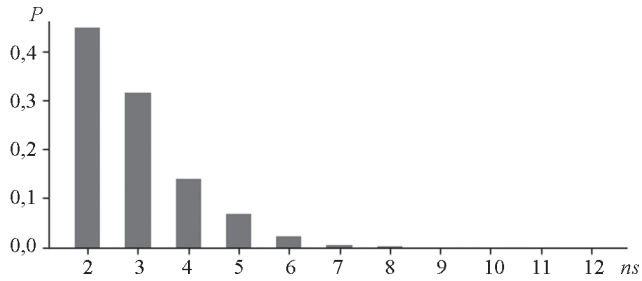


Рис. 4. Диаграмма вероятностей P принадлежности случайной леммы группам лемм

где Σ_{fns} — сумма частот всех лемм с числом слогов ns; Σf — сумма частот всех лемм словаря [14] с числом слогов 2, 3, ..., 12.

Замечание. В [14] имеются 3 леммы с числом слогов 13 и одна «*информационно-телекоммуникационный*» — с 15-ю слогами.

Используя (1), словарь [14] и данные табл. 4, оценим вероятности P_m^F и P_m правильного определения ударения случайно выбранного слова (табл. 5) класса C_m (m = 1, 2).

Приведенные в табл. 5 значения получены для полного и разделенного наборов данных:

$$P_m^F = \sum_{ns=2}^{12} P_{ns} acc_{v_{ns}}^F acc_{m_{ns}}^F;$$

$$P_m = \sum_{ns=2}^{12} P_{ns} val\_acc_{v_{ns}} val\_acc_{m_{ns}}, \ (m = 1,\ 2).$$

Оценим вероятность того, что случайно выбранное слово будет в словаре [2].

$$P_{[2]} = |W_{[2]} \cup W_{[14]}|/|W_{[14]}| = 0{,}7012,$$

где $W_{[2]}$, $W_{[14]}$ — множества основ словаря [2] и лемм [14] с числом слогов более 1.

Таблица 5

Оценки вероятности правильного определения ударения

Набор данных	C_1	C_2
Полный	$P_1^F = 0{,}9734$	$P_2^F = 0{,}9928$
Разделенный	$P_1 = 0{,}8863$	$P_2 = 0{,}9361$

Зная $P_{[2]}$, найдем для C_1 и C_2, с какой вероятностью классификаторы (см. табл. 4) правильно определят ударение в случайно выбранном слове:

$$P_{C_1} = P_{[2]}P_1^F + (1-P_{[2]})P_1 = 0{,}9474;$$
$$P_{C_2} = P_{[2]}P_2^F + (1-P_{[2]})P_2 = 0{,}9759.$$

Замечание. В предложении *татаканье веялочных бараба́нов* программа расставит ударения следующим образом: *тата́канье ве́ялочных бараба́нов*. В [2] имеется только последнее слово этого предложения.

Заключение

Новизна рассмотренного подхода состоит, во-первых, в двухуровневой декомпозиции задачи автоматического определения словесных ударений: выделяются классы слов C_m (m = 0, 3). Затем слова классов C_1 и C_2, соответственно с одним и двумя непереходящими ударениями, группируются по числу слогов, и поиск ударных слогов проходит независимо в каждой группе. Вторым аспектом новизны является сведение проблемы определения ударений к задаче классификации: в группах C_1 номер класса слова совпадает с номером ударного слога, а в группах C_2 — указывает на пару ударных слогов.

Таким образом, на языке Python разработана программа [9], устанавливающая класс C_m (m = 0, 3) исследуемого слова, а затем, если слово принадлежит C_1 или C_2, выбирающая подходящую НС (определитель ударения), передающая ей слово и возвращающая в случае C_1 номер ударного слога или номера двух ударных слогов — в случае C_2.

Созданные НС правильно определяют ударения в случайно выбранном слове с вероятностью 0,9474 в C_1 и 0,9759 — в C_2.

Полученный результат является экспериментальным и может быть улучшен за счет усовершенствования НС, выполняющих начальную классификацию слов и определяющих в них ударения.

Литература

1. **Зализняк А.А.** Грамматический словарь русского языка: словоизменение. М.: Русский язык, 1980.

2. **Грамматический** словарь Зализняка А.А. [Электрон. ресурс] www.gufo.me/dict/zaliznyak (дата обращения 01.06.2020).

3. **Программа** расстановки ударений [Электрон. ресурс] www.morpher.ru/accentizer/ (дата обращения 01.06.2020).

4. **Хомицевич О.Г. и др.** Автоматическое определение места ударения в незнакомых словах в системе синтеза речи // Материалы XXXVI Междунар. филолог. конф. СПб., 2008. С. 175—183.

5. **Цирульник Л.И., Покладок Д.А.** Грамматический словарь и правила определения словесного ударения для синтеза речи по тексту на мобильных устройствах // Информатика. 2012. № 2. С. 61—68.

6. **Бабайцева В.В.** Русский язык. Теория. 5 — 11 классы. М.: Дрофа, 1998.

7. **Розенталь Д.Э., Голуб И.Б., Теленкова М.А.** Современный русский язык. М.: Рольф, Айрим-пресс, 1997.

8. **Морфологический** словарь русского языка в виде SQL скрипта [Электрон. ресурс] www.shra.ru/2017/03/morfologicheskijj-slovar-russkogo-yazyka-v-vide-sql-skripta (дата обращения 01.06.2020).

9. **Бартеньев О.В.** Подготовка данных для автоматического определения русских словесных ударений [Электрон. ресурс] www.100byte.ru/stdntswrks/accents/accents.html (дата обращения 01.06.2020).

References

1. **Zaliznyak A.A.** Grammaticheskiy slovar' Russkogo Yazyka: Slovoizmenenie. M.: Russkiy Yazyk, 1980. (in Russian).

2. **Grammaticheskiy** Slovar' Zaliznyaka A.A. [Elektron. Resurs] www.gufo.me/dict/zaliznyak (Data Obrashcheniya 01.06.2020). (in Russian).

3. **Programma** Rasstanovki Udareniy [Elektron. Resurs] www.morpher.ru/accentizer/ (Data Obrashcheniya 01.06.2020). (in Russian).

4. **Khomitsevich O.G. i dr.** Avtomaticheskoe Opredelenie Mesta Udareniya v Neznakomykh Slovakh v Sisteme Sinteza Rechi. Materialy XXXVI Mezhdunar. Filolog. Konf. SPb., 2008:175—183. (in Russian).

5. **Tsirul'nik L.I., Pokladok D.A.** Grammaticheskiy Slovar' i Pravila Opredeleniya Slovesnogo Udareniya dlya Sinteza Rechi po Tekstu na Mobil'nykh Ustroystvakh. Informatika. 2012;2:61—68. (in Russian).

6. **Babaytseva V.V.** Russkiy Yazyk. Teoriya. 5 — 11 Klassy. M.: Drofa, 1998. (in Russian).

7. **Rozental' D.E., Golub I.B., Telenkova M.A.** Sovremennyy Russkiy Yazyk. M.: Rol'f, Ayrim-press, 1997. (in Russian).

8. **Morfologicheskiy** Slovar' Russkogo Yazyka v Vide SQL Skripta [Elektron. Resurs] www.shra.ru/2017/03/morfologicheskijj-slovar-russkogo-yazyka-v-vide-sql-skripta (Data Obrashcheniya 01.06.2020). (in Russian).

9. **Barten'ev O.V.** Podgotovka Dannykh dlya Avtomaticheskogo Opredeleniya Russkikh Slovesnykh Udareniy [Elektron. Resurs] www.100byte.ru/stdntswrks/accents/accents.html (Data Obrashcheniya 01.06.2020). (in Russian).

10. **Word Embeddings** [Электрон. ресурс] www.tensorflow.org/tutorials/text/word_embeddings?hl.ru (дата обращения 01.06.2020).

11. **Keras:** the Python Deep Learning Library [Электрон. ресурс] www.keras.io/ (дата обращения 01.06.2020).

12. **Zhu S., Chollet F.** Working with RNNs [Электрон. ресурс] www.keras.io/guides/working_with_rnns/ (дата обращения 01.06.2020).

13. **Бенгфорт Б., Билбро Р., Охеда Т.** Прикладной анализ текстовых данных на Python. СПб.: Питер, 2019.

14. **Ляшевская О.Н., Шаров С.А.** Частотный словарь современного русского языка (на материалах Национального корпуса русского языка). М.: Азбуковник, 2009.

15. **Национальный** корпус русского языка [Электрон. ресурс] www.ruscorpora.ru/new/ (дата обращения 01.06.2020).

10. **Word Embeddings** [Elektron. Resurs] www.tensorflow.org/tutorials/text/word_embeddings?hl.ru (Data Obrashcheniya 01.06.2020). (in Russian).

11. **Keras: t**he Python Deep Learning Library [Elektron. Resurs] www.keras.io/ (Data Obrashcheniya 01.06.2020).

12. **Zhu S., Chollet F.** Working with RNNs [Elektron. Resurs] www.keras.io/guides/working_with_rnns/ (Data Obrashcheniya 01.06.2020).

13. **Bengfort B., Bilbro R., Okheda T.** Prikladnoy Analiz Tekstovykh Dannykh na Python. SPb.: Piter, 2019. (in Russian).

14. **Lyashevskaya O.N., Sharov S.A.** Chastotnyy Slovar' Sovremennogo Russkogo Yazyka (na materialakh Natsional'nogo Korpusa Russkogo Yazyka). M.: Azbukovnik, 2009. (in Russian).

15. **Natsional'nyy** Korpus Russkogo Yazyka [Elektron. Resurs] www.ruscorpora.ru/new/ (Data Obrashcheniya 01.06.2020). (in Russian).

Сведения об авторе:

Бартеньев Олег Васильевич — кандидат технических наук, доцент кафедры прикладной математики и искусственного интеллекта НИУ «МЭИ», e-mail: mdf4@mail.ru

Information about author:

Barten′ev Oleg V. — Ph.D. (Techn.), Assistant Professor of Applied Mathematics and Artificial Intelligence Dept., NRU MPEI, e-mail: mdf4@mail.ru

Статья поступила в редакцию: *22.06.2020*
The article received to the editor: *22.06.2020*