Аннотация

Тут будет аннотация

Содержание

1. Введение

На стыке лингвистики и computer science в середине XX века возникла компьютерная лингвистика. Это научное направление развивается по мере развития электронно-вычислительных машин [3].

2. Анализ проблемы

Задача согласования единственного и множественного числа не решена; причём, не только в русском языке.

Английский язык «проще» тем, что в нём строгий порядок слов: SVO («субъект – глагол – объект») [4]. Однако, даже для английского языка сформулированная проблема не решена.

Одной из наиболее успешных работ в этой области стала публикация Дэмиана Конвея (Damian Conway) «An algorithmic approach to English pluralization» [5]. В ней автор разрабатывает алгоритмы, преобразующие существительные, прилагательные и глаголы в единственном числе в соответствующие формы множественного числа. Также Конвей приводит алгоритм, позволяющий идентифицировать слова, отличающиеся только числом. Полная реализация данных алгоритмов была сделана автором публикации на языке Perl.

Русский язык, с одной стороны, относится к языкам с фиксированным порядком слов «SVO» (как и английский). Однако, с другой стороны, гибкий: SV / VS [6]. За счёт этого задача становится на порядок сложнее.

Согласование единственного и множественного числа в русском предложении было исследовано в бакалаврской диссертации Дзюбенко Василия Александровича в 2020 году [1]. Автором была разработана модель со стеком, совершающую свёртку и сдвиг, аналогичную GLR-анализатору; данная модель позволяла распознавать ошибки согласования в некоторых предложениях.

Тем не менее, В.А. Дзюбенко создал базу данных, в которой содержится информация о подавляющем большинстве слов русского языка. Данная база стала одним из базовых инструментов для решения нами поставленной проблемы.

3. Описание подхода

3.1. Идея

Как было сказано в анализе проблемы, именно база данных слов русского языка, разработанная В.А. Дзюбенко, стала базовым инструментом для решения задачи при помощи предложенного нами подхода.

Мы дополнили существующую базу данных несколькими таблицами; её схема представлена на рис. 1, а структура таблиц расписана в приложенииA.

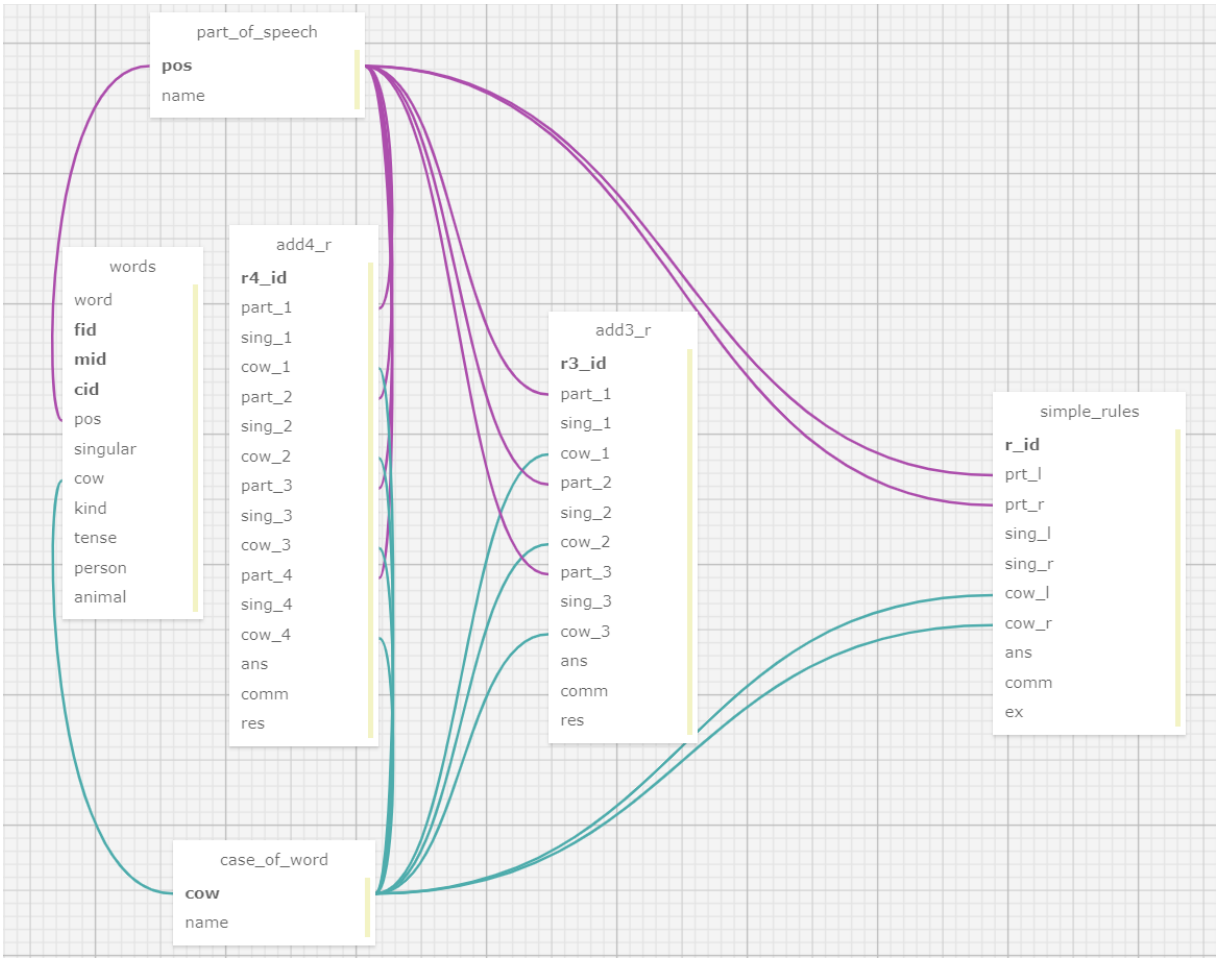

Рис. 1 – Концептуальная схема усовершенствованной базы данных

В рамках предложенного нами подхода мы описываем слово при помощи трёх параметров: часть речи, число и падеж. Нами было выявлено около сотни правил для словосочетаний длины 2, 3 или 4, по которым можно определить, нет ли ошибки в согласовании единственного и множественного числа? Полученные результаты представлены в приложении B

В основе предложенного нами подхода лежит гипотеза, согласно которой од-

но и то же предложение являться и не являться ошибочным одновременно с точки зрения согласования единственного и множественного числа не может.

Также считаем, что в предложении нет орфографических, пунктуационных и др. ошибок, поскольку данная задача была успешно решена, например, компанией LanguageTooler GmbH [7].

3.2. Подготовительный этап

На вход программе подаётся предложение, состоящее из существительных, местоимений, личных глаголов или (и) инфинитивов (с, возможно, перечислением инфинитивов или личных глаголов с зависимыми словами, принадлежащим указанным частям речи).

Полученное предложение передаётся функции space(), которая преобразует считанную строку в список. Механизм её работы описывает алгоритм 1.

Алгоритм 1 – Предварительная обработка входных данных

```
1:  function SPACE(str1)
2:      str1 ← str1.lower()        ▷ Приводим полученную строку к нижнему регистру
3:      str2 ← «»        ▷ В этой переменной будет храниться преобразованная строка
4:      l ← len(str1)
5:      for from i = 0 to l − 1 do
6:          if str1[i] ∈ {«.»; «,» } then
7:              str2 ← str2 + « »
8:          end if
9:          str2 ← str2 + str1[i]
10:     end for
11:     if str2[len(str2) −1] = « » then
12:         str2 ← str2[ : len(str2) −1]                    ▷ Отбрасываем этот пробел
13:     end if
14:     return str2.split()          ▷ Возвращаем список, полученный из строки str2
    разбиением её по пробелам
15: end function
```

Таким образом, функция space() возвращает список, состоящий из слов и знаков препинания исходной строки.

3.3. Обработка предложения

Итак, как было сказано выше, проверка согласования единственного и множественного числа в русском языке — процесс сложный: нужно учесть много критериев.

Нами было принято решение декомпозировать задачу.

Для начала (при наличии перечислений инфинитивов или личных личных глаголов) предложение упрощается: перечисление мы заменяем на инфинитив или личный глагол соответственно (параллельно проверяя, что внутри заменяемой части нет ошибок в согласовании единственного и множественного числа). Если же перечисления не обнаружено, сразу переходим к следующему этапу.

Затем проверяем предложение без перечислений при помощи разработанной нами системы правил.

Согласно теореме Гёделя о неполноте, формальная арифметика либо противоречива, либо неполна [2]. Чтобы избежать противоречивости разработанной системы, мы включили лишь те правила, которые встречаются на практике, а не перебрали все возможные комбинации используемых нами параметров.

3.3.1. Упрощение предложения

Под упрощением мы будем понимать замену перечисления инфинитивов или личных глаголов одиночным инфинитивом или личным глаголом.

За упрощение предложения отвечает функция comma(), которая принимает на вход список, полученный из исходного предложения при помощи функции space(), описанной выше; а возвращает список, в виде которого представлено упрощённое предложение. Функция comma() вызывается только в том случае, если в предложении есть «и» или «или». Механизм её работы описывает алгоритм 2.

Алгоритм 2 – Обработка перечислений

```
1:  function COMMA(l)                          ▷ l — подготовленная строка в виде списка
2:      part ← [ ]      ▷ Список значений параметра «часть речи» для данного слова (изначально пустой)
3:      end← (−1)                              ▷ Индикатор нахождения начала перечисления
4:      left ← (−1)      ▷ Левая и правая границы заменяемого участка изначально
5:      right ← (−1)                           ▷ инициализируем невозможными значениями: (−1)
6:      llen ←len(l)                                          ▷ Длина исходного списка
7:      if «и» ∈ l then
8:          w← «и»                             ▷ w хранит союз, использующийся в перечислении
9:      else if «или» ∈ l then
10:         w← «или»
11:     end if
12:     id1 ← l.index(w)                                  ▷ Записываем индекс w в списке
```

В рамках установленных нами ограничений «и» или «или» могут быть использованы только для перечислений личных глаголов или инфинитивов.

Алгоритм 3 – Продолжение алгоритма 2

```
13:     if llen > id1 +1 then
14:         resp1 ← l[id1 +1].[pos, singular, cow]
15:     end if
16:     if llen > id1 +2 then
17:         resp2 ← l[id1 +2].[pos, singular, cow]
18:     end if
19:     if llen > id1 +3 then
20:         resp3 ← l[id1 +3].[pos, singular, cow]
21:     end if
22:     if llen > id1 +4 then
23:         resp4 ← l[id1 +4].[pos, singular, cow]
24:     end if
25:     if id1 −1 ≥ 0 then
26:         resl1 ← l[id1 −1].[pos, singular, cow]
27:     end if
28:     if id1 −2 ≥ 0 then
29:         resl2 ← l[id1 −2].[pos, singular, cow]
30:     end if
31:     if id1 −3 ≥ 0 then
32:         resl3 ← l[id1 −3].[pos, singular, cow]
33:     end if
34:     if id1 −4 ≥ 0 then
35:         resl1 ← l[id1 −4].[pos, singular, cow]
36:     end if
37:     if id1 −5 ≥ 0 then
38:         resl5 ← l[id1 −5].[pos, singular, cow]
39:     end if
40:     if id1 −6 ≥ 0 then
41:         resl6 ← l[id1 −6].[pos, singular, cow]
42:     end if
43:     if id1 −7 ≥ 0 then
44:         resl7 ← l[id1 −7].[pos, singular, cow]
45:     end if
46:     if id1 +1 < llen then
47:         for from i = 0 to len(resp1)−1 do
48:             if resp1[i][0] =«6» then                ▷ Слово оказалось инфинитивом
49:                 part← «6» and right ← id1 +1
50:             end if
51:         end for
52:         if right= (−1) then                         ▷ Если же это не инфинитив
53:             for from i = 0 to len(resp1)−1 do
54:                 if resp1[i][0] =«5» then            ▷ Слово оказалось личным глаголом
55:                     right ←id1+1 and part ← «5»
56:                     sng ←l[i][1]                    ▷ Для личных глаголов важно число
57:                     break
58:                 end if
59:             end for
60:         end if
61:     end if
```

Таким образом, в результате исполнения блока 3 будет определено, слова (словосочетания) какой части речи перечисляются (если в предложении присутствует перечисление с союзом).

Заметим, что в случае перечисления с союзом за союзом идёт слово той же части речи, что и остальные перечисляемые слова. Например: *«Он хотел* ***читать*** *книги,* ***рисовать*** *картины и* ***познавать*** *тайны мироздания»*. Легко видеть, что в предложении перечисляются инфинитивы, и в то же время после союза «и» идёт инфинитив «познавать».

Если перечисляются инфинитивы, то упрощение предложения идёт согласно алгоритму 4.

Прежде всего, нужно определить начало левого операнда союза. В зависимости от длины буквосочетания, возможны различные варианты:

1. Словосочетание длины 6. Например, инф. + сущ. + сущ. + сущ. + сущ. + сущ.: *«Организовать проверку знаний требований охраны труда»*.

2. Словосочетание длины 5. Например, инф. + сущ. + сущ. + сущ. + сущ.: *«Организовать проверку знаний основ программирования»*.

3. Словосочетание длины 4. Например, инф. + инф. + сущ. + сущ.: *«Пойти спать сном младенца»*.

4. Словосочетание длины 3. Например, инф. + сущ. + сущ.: *«Оценить игру слов»*.

5. Словосочетание длины 2. Например, инф. + инф.: *«Пойти позавтракать»*.

6. Одиночный инфинитив. Например: *«Быть»*.

Итак, первым делом инициализируем левую границу заменяемого «куска» списка.

Алгоритм 4 – Продолжение алгоритма 3

```
62:     if part =«6» then                ▷ Если перечисляемая часть речи — инфинитив
63:         if id1−6 ≥ 0 and left= (−1) and «,»∉ l[id1−6 : id1] then      ▷ Проверяем буквосочетания длины 6
64:             for from i = 0 to len(resl6)−1 do
65:                 if resl6[i][0] =part then
66:                     r← check(l[id1−6 : id1])
67:                     if «N»∈ r then
68:                         return [«он», «писали» ]   ▷ Заведомо неверное предложение
69:                     else if «Y»∈ r then
```

Алгоритм 5 – Продолжение алгоритма 4

```
                    left ← id1−6   ▷ Инициализировали границу левого операнда «и»
                    break
                end if
            end if
        end for
    end if
    if id1−5 ≥ 0 and left= −1 and «,»∉ l[id1−5 : id1] then
        for from i = 0 to len(resl5)−1 do
            if resl5[i][0] =part then
                r← check(l[id1−5 : id1])
                if «N»∈ r then
                    return [«он», «писали» ]
                else if «Y»∈ r then
                    left ← id1−5
                    break
                end if
            end if
        end for
    end if
    if id1−4 ≥ 0 and left= −1 and «,»∉ l[id1−4 : id1] then
        for from i = 0 to len(resl4)−1 do
            if resl4[i][0] =part then
                r← check(l[id1−4 : id1])
                if «N»∈ r then
                    return [«он», «писали» ]
                else if «Y»∈ r then
                    left ← id1−4
                    break
                end if
            end if
        end for
    end if
    if id1−3 ≥ 0 and left= −1 and «,»∉ l[id1−3 : id1] then
        for from i = 0 to len(resl3)−1 do
            if resl3[i][0] =part then
                r← check(l[id1−3 : id1])
                if «N»∈ r then
                    return [«он», «писали» ]
                else if «Y»∈ r then
                    left ← id1−3
                    break
                end if
            end if
        end for
    end if
```

Таким образом, в результате выполнения данного фрагмента кода будет определены границы левого и правого операндов союза.

Алгоритм 6 – Продолжение алгоритма 5

```
115:        if id1−2 ≥ 0 and left= −1 and «,»∉ l[id1−2 : id1] then
116:            for from i = 0 to len(resl2)−1 do
117:                if resl2[i][0] =part then
118:                    r← check(l[id1−2 : id1])
119:                    if «N»∈ r then
120:                        return [«он», «писали» ]
121:                    else if «Y»∈ r then
122:                        left ← id1−2
123:                        break
124:                    end if
125:                end if
126:            end for
127:        end if
128:        if id1−1 ≥ 0 and left= (−1) then
129:            for from i = 0 to len(resl1)−1 do
130:                if l[i][0] =part then
131:                    left ← id1−1
132:                    break
133:                end if
134:            end for
135:        end if
136:    end if
137:    Алгоритм 7
138:    Алгоритм 11
139:    Алгоритм 12
140:    return l
141: end function
```

В алгоритмах 5 и 6 неоднократно фигурирует функция check(). В данном случае она используется для проверки предложения, не содержащего знаки пунктуации. Её описание будет в следующем параграфе.

Далее возможен один из двух вариантов:

- Союз связывает только два слова или словосочетания, или слово и словосочетание.
- Союз используется для перечисления 3 и более словосочетаний и (или) слов.

В первом случае предложение готово к упрощению: «кусок» от left до right заменяем единичным инфинитивом.

Во втором же случае необходимо продолжить анализ предложения, сдвигая левую границу заменяемого участка.

Для начала будем искать участки между двумя запятыми (при их наличии). Особенность данного этапа заключается в том, что между запятыми может оказаться ошибочное словосочетание, — потому фрагменты между запятыми нужно также проверять на согласованность.

Также важен порядок рассмотрения случаев: в первую очередь следует искать самые «короткие» словосочетания между запятыми (иначе можем «захватить» подстроку с запятыми). Этот и последующие этапы описаны в алгоритме 7.

Алгоритм 7 – Фрагмент алгоритма 6

```
1:  while «,»∈ l[1 : left ] do                                  ▷ Пока есть запятые
2:      if l[left −1] = «,» and l[left −3] = «,» then      ▷ Между запятыми одно слово
3:          res1← l[left−2].[pos, singular, cow]
4:          for from i = 0 to len(res1) do
5:              if res1[i][0] =part then
6:                  left← left−2
7:                  break
8:              end if
9:          end for
10:     else if l[left −1] = «,» and l[left −4] = «,» then          ▷ Между запятыми
    словосочетание из двух слов
11:         res1← l[left−3].[pos, singular, cow]
12:         for from i = 0 to len(res1) do
13:             if res1[i][0] =part then
14:                 r ← check(l[left−3 : left−1])
15:                 if «N» ∈ r then
16:                     return [«он», «писали» ]
17:                 else if «Y» ∈ r then
18:                     left← left−3
19:                     break
20:                 end if
21:             end if
22:         end for
23:     else if l[left −1] = «,» and l[left −5] = «,» then
24:         res1← l[left−4].[pos, singular, cow]
25:         for from i = 0 to len(res1) do
26:             if res1[i][0] =part then
27:                 r ← check(l[left−4 : left−1])
28:                 if «N» ∈ r then
29:                     return [«он», «писали» ]
30:                 else if «Y» ∈ r then
31:                     left← left−4
32:                     break
```

Алгоритм 8 – Продолжение алгоритма 7

```
                end if
            end if
        end for
    else if l[left −1] = «,» and l[left −6] = «,» then
        res1← l[left−5].[pos, singular, cow]
        for from i = 0 to len(res1) do
            if res1[i][0] =part then
                r ← check(l[left−5 : left−1])
                if «N» ∈ r then
                    return [«он», «писали» ]
                else if «Y» ∈ r then
                    left← left−5
                    break
                end if
            end if
        end for
    else if l[left −1] = «,» and l[left −7] = «,» then
        res1← l[left−6].[pos, singular, cow]
        for from i = 0 to len(res1) do
            if res1[i][0] =part then
                r ← check(l[left−6 : left−1])
                if «N» ∈ r then
                    return [«он», «писали» ]
                else if «Y» ∈ r then
                    left← left−6
                    break
                end if
            end if
        end for
    else if l[left −1] = «,» and l[left −8] = «,» then
        res1← l[left−7].[pos, singular, cow]
        for from i = 0 to len(res1) do
            if res1[i][0] =part then
                r ← check(l[left−7 : left−1])
                if «N» ∈ r then
                    return [«он», «писали» ]
                else if «Y» ∈ r then
                    left← left−7
                    break
                end if
            end if
        end for
    else
        Алгоритм 9
    end if
end while
```

Итак, в результате работы фрагментов 7 и 8 будет сдвинута граница до «первой» запятой.

Следующий этап — поиск начала перечисления. Соответствующий фрагмент описан алгоритмом 9.

Индикатор end отвечает за нахождение начала перечисления (изначально был инициализирован (-1), а после нахождения начала перечисления будет равен 1). Как и раньше, проверяем первый найденную подстроку на выполнение правил в ней.

Алгоритм 9 – Фрагмент алгоритма 8

```
1: if left−2 ≥ 0 and l[left−1] =«,» and end= (−1) then
2:     res1← l[left−2].[pos, singular, cow]
3:     for from i = 0 to len(res1)−1 do
4:         if res1[i][0] =«6» then
5:             left← left−2
6:             break
7:         end if
8:     end for
9: end if
10: if left−3 ≥ 0 and l[left−1] =«,» and end= (−1) then
11:     res1← l[left−3].[pos, singular, cow]
12:     for from i = 0 to len(res1)−1 do
13:         if res1[i][0] =«6» then
14:             r ← check(l[left−3 : left−1])
15:             if «N»∈ r then
16:                 return [«он», «писали» ]
17:             else if «Y» ∈ r then
18:                 left← left−3
19:                 end ← 1
20:                 break
21:             end if
22:         end if
23:     end for
24: end if
25: if left−4 ≥ 0 and l[left−1] =«,» and end= (−1) then
26:     res1← l[left−4].[pos, singular, cow]
27:     for from i = 0 to len(res1)−1 do
28:         if res1[i][0] =«6» then
29:             r ← check(l[left−4 : left−1])
30:             if «N»∈ r then
31:                 return [«он», «писали» ]
32:             else if «Y» ∈ r then
33:                 left← left−4
34:                 end ← 1
35:                 break
36:             end if
```

Алгоритм 10 – Продолжение алгоритма 9

```
        end if
    end for
end if
if left−5 ≥ 0 and l[left−1] =«,» and end= (−1) then
    res1← l[left−5].[pos, singular, cow]
    for from i = 0 to len(res1)−1 do
        if res1[i][0] =«6» then
            r ← check(l[left−5 : left−1])
            if «N»∈ r then
                return [«он», «писали» ]
            else if «Y» ∈ r then
                left← left−5
                end ← 1
                break
            end if
        end if
    end for
end if
if left−6 ≥ 0 and l[left−1] =«,» and end= (−1) then
    res1← l[left−6].[pos, singular, cow]
    for from i = 0 to len(res1)−1 do
        if res1[i][0] =«6» then
            r ← check(l[left−6 : left−1])
            if «N»∈ r then
                return [«он», «писали» ]
            else if «Y» ∈ r then
                left← left−6
                end ← 1
                break
            end if
        end if
    end for
end if
if left−7 ≥ 0 and l[left−1] =«,» and end= (−1) then
    res1← l[left−7].[pos, singular, cow]
    for from i = 0 to len(res1)−1 do
        if res1[i][0] =«6» then
            r ← check(l[left−7 : left−1])
            if «N»∈ r then
                return [«он», «писали» ]
            else if «Y» ∈ r then
                left← left−7
                end ← 1
                break
            end if
        end if
    end for
end if
```

Итак, после выполнения алгоритма 9 будут определены границы заменяемой подстроки, после чего необходимо вставить вместо перечисления инфинитивов одиночный инфинитив. Нами было выбрано слово *«учить»* (для данной цели можно было выбрать любой инфинитив, так как мы решаем проблему согласования единственного и множественного числа).

Алгоритм 11 – Фрагмент алгоритма 6

```
1: l← l[ : left]+ [«учить»]+ l[ right+1 :id1]
```

Если же при помощи союза «и» перечисляются личные глаголы, то упрощение идёт согласно алгоритму 12. В зависимости от длины буквосочетания, возможны различные варианты словосочетаний:

1. Словосочетание длины 7. Например, личн. глаг. + инф. + сущ. + сущ. + сущ. + + сущ.: *«Хотел организовать проверку знаний требований охраны труда»*.

2. Словосочетание длины 6. Например, личн. глаг. + сущ. + сущ. + сущ. + сущ. + + сущ.: *«Организовывал проверку знаний требований охраны труда»*.

3. Словосочетание длины 5. Например, личн. глаг. + инф. + сущ. + сущ. + сущ.: *«Хотел изучить основы теории кодирования»*.

4. Словосочетание длины 4. Например, личн. глаг. + сущ. + сущ. + сущ.: *«Изучил основы теории кодирования»*.

5. Словосочетание длины 3. Например, личн. глаг. + инф. + сущ.: *«Желает знать правду»*.

6. Словосочетание длины 2. Например, личн. глаг. + инф.: *«Желает знать»*.

7. Одиночный личный глагол. Например: *«Желать»*.

Во многом алгоритм обработки перечислений личных глаголов похож на алгоритм обработки перечислений инфинитивов.

Однако, в отличие от последних, для личных глаголов определено понятие числа. И в данной ситуации возникает ***проблема омографии***. Так, слово *«спАли»* — личный глагол во множественном числе, а *«спалИ»* — личный глагол в единственном числе. В самом деле, данные слова совпадают по написанию, но различны по звучанию и значению. Заметим, что в единственном числе слово интерпретируется тогда

и только тогда, когда оно в повелительном наклонении. Легко видеть, что на множестве рассматриваемых в данной работе частей речи перечисляются личные глаголы в повелительном наклонении тогда и только тогда, когда предложение начинается с глагола в повелительном наклонении. Потому сразу определим, является ли первое слово глаголом. Если да, однозначно ли определяется его число.

Алгоритм 12 – Продолжение алгоритма 6

```
1:  if part=«5» then
2:      sng0 ← [ ]                ▷ Для определения числа первого слова в предложении
3:      pov ← (−1)                          ▷ Индикатор повелительного наклонения
4:      res0← l[0].[cow, singular, cow]
5:      end← (−1)
6:      for from i = 0 to len(res0)−1 do
7:          if res0[i][0] =«5» then
8:              sng0 ← sng0 + list(res0[i][1])
9:          end if
10:     end for
11:     sng0 ← list(set(sng0))
12:     if len(sng0)> 1 then
13:         pov ← 1
14:         sng ←«Y»          ▷ Считаем единственным число перечисляемых личных
    глаголов
15:     end if
16:     if id1−7 ≥ 0 and left= (−1) and «,»∉ l[id1−7 : id1] then
17:         sng1 ← [ ]                          ▷ Для записи возможных значений числа
18:         r ← [ ]                ▷ Для записи результата проверки по системе правил
19:         for from i = 0 to len(resl7)−1 do
20:             if resl7[i][0] =part then
21:                 sng1← sng1 + list(resl7[i][1])
22:                 r← r + list(check(l[id1−7 : id1 ]))
23:             end if
24:         end for
25:         if len(sng1)> 0 then    ▷ Если словосочетание действительно начинается с
    личного глагола
26:             if (len(sng1)= 1 and sng ∈ sng1 and «Y»∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov= 1) and «Y» ∈ r) then
27:                 left← id1−7
28:             else
29:                 return [«он», «писали» ]
30:             end if
31:         end if
32:     end if
```

Здесь следует рассмотреть решение проблемы омографии. Оно представлено в строках 12.6 – 12.32. Для начала проверяем, является ли первое слово рассматри-

ваемого предложения личным глаголом, число которого определено неоднозначно. Это имеет значение, поскольку в рамках поставленных ограничений если в предложении есть личные глаголы в повелительном наклонении, то с одного из них оно начинается. Например: *«Учите математику, высыпайтесь и будьте людьми»*.

Далее проверяем, что рассматриваемое словосочетание нужной длины и не содержит запятых. Если вдруг первое слово данного словосочетания оказалось личным глаголом, то мы запоминаем какого оно числа может быть; а также проверяем данное словосочетание на наличие или отсутствие ошибок в согласовании единственного и множественного числа.

Словосочетание не содержит ошибок в следующих случаях:

1. Число личного глагола опрелеляется однозначно и совпадает с числом правого операнда союза, проверка словосочетания на наличие ошибок в согласовании единственного и множественного числа прошла успешно (ошибок и незнакомых сочетаний не обнаружено).

2. Число личного глагола определяется неоднозначно, и имеет место повелительное наклонение.

3. Число личного глагола определяется неоднозначно, первое слово в предложении не является личным глаголом в повелительном наклонении и перечисляются личные глаголы во множественном числе.

Это мы и проверяем. В остальных случаях возвращаем заведомо неверное предложение. Совершенно аналогично рассматриваются словосочетания меньшей длины:

Алгоритм 13 – Продолжение алгоритма 12

```
33:     if id1−6 ≥ 0 and left= (−1) and «,»∉ l[id1−6 : id1] then
34:         sng1 ← [ ]
35:         r ← [ ]
36:         for from i = 0 to len(resl6)−1 do
37:             if resl6[i][0] =part then
38:                 sng1← sng1 + list(resl6[i][1])
39:                 r← r + list(check(l[id1−6 : id1 ]))
40:             end if
41:         end for
42:         if len(sng1)> 0 then
```

Алгоритм 14 – Продолжение алгоритма 13

```
            if (len(sng1)= 1 and sng ∈ sng1 and «Y»∈ r) or (len(sng1)> 1 and (sng=«N» or pov= 1) and «Y» ∈ r) thenя
                left← id1−6
            else
                return [«он», «писали» ]
            end if
        end if
    end if
    if id1−5 ≥ 0 and left= (−1) and «,»∉ l[id1−5 : id1] then
        sng1 ← []
        r ← []
        for from i = 0 to len(resl5)−1 do
            if resl5[i][0] =part then
                sng1← sng1 + list(resl5[i][1])
                r← r + list(check(l[id1−5 : id1 ]))
            end if
        end for
        if len(sng1)> 0 then
            if (len(sng1)= 1 and sng ∈ sng1 and «Y»∈ r) or (len(sng1)> 1 and (sng=«N» or pov= 1) and «Y» ∈ r) then
                left← id1−5
            else
                return [«он», «писали» ]
            end if
        end if
    end if
    if id1−4 ≥ 0 and left= (−1) and «,»∉ l[id1−4 : id1] then
        sng1 ← []
        r ← []
        for from i = 0 to len(resl4)−1 do
            if resl4[i][0] =part then
                sng1← sng1 + list(resl4[i][1])
                r← r + list(check(l[id1−4 : id1 ]))
            end if
        end for
        if len(sng1)> 0 then
            if (len(sng1)= 1 and sng ∈ sng1 and «Y»∈ r) or (len(sng1)> 1 and (sng=«N» or pov= 1) and «Y» ∈ r) then
                left← id1−4
            else
                return [«он», «писали» ]
            end if
        end if
    end if
    if id1−3 ≥ 0 and left= (−1) and «,»∉ l[id1−3 : id1] then
        sng1 ← []
        r ← []
        for from i = 0 to len(resl3)−1 do
```

Алгоритм 15 – Продолжение алгоритма 14

```
88:             if resl3[i][0] =part then
89:                 sng1← sng1 + list(resl3[i][1])
90:                 r← r + list(check(l[id1−3 : id1 ]))
91:             end if
92:         end for
93:         if len(sng1)> 0 then
94:             if (len(sng1)= 1 and sng ∈ sng1 and «Y»∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov= 1) and «Y» ∈ r) then
95:                 left← id1−3
96:             else
97:                 return [«он», «писали» ]
98:             end if
99:         end if
100:    end if
101:    if id1−2 ≥ 0 and left= (−1) and «,»∉ l[id1−2 : id1] then
102:        sng1 ← [ ]
103:        r ← [ ]
104:        for from i = 0 to len(resl2)−1 do
105:            if resl2[i][0] =part then
106:                sng1← sng1 + list(resl2[i][1])
107:                r← r + list(check(l[id1−2 : id1 ]))
108:            end if
109:        end for
110:        if len(sng1)> 0 then
111:            if (len(sng1)= 1 and sng ∈ sng1 and «Y»∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov= 1) and «Y» ∈ r) then
112:                left← id1−2
113:            else
114:                return [«он», «писали» ]
115:            end if
116:        end if
117:    end if
118:    if id1−1 ≥ 0 and left= (−1) then
119:        sng1 ← [ ]
120:        for from i = 0 to len(resl1) do
121:            if resl1[i][0] =part then
122:                sng1← sng1 + list(resl1[i][1])
123:            end if
124:        end for
125:        sng1 ← list(set(sng1))
126:        if (len(sng1)= 1 and sng ∈ sng1) or (len(sng1)> 1) then
127:            left← id1−1
128:        else
129:            return [«он», «писали» ]
130:        end if
131:    end if
```

Итак, по завершении работы алгоритма 15 будут определены границы левого и правого операндов союза.

Затем, как и в случае с инфинитивами, осуществляется поиск участков между двумя запятыми (при их наличии), а затем — поиск начала перечисления.

Алгоритм 16 – Продолжение алгоритма 15

```
132:    while «,»∈ l[1 : left ] do
133:        if left−3 ≥ 0 and l[left−3] = «,» and l[left−1] = «,» then
134:            sng1← [ ]
135:            res1← l[left−2].[pos, singular, cow]
136:            for from i = 0 to len(res1)−1 do
137:                if res1[i][0] =part then
138:                    sng1← sng1 + list(res[i][1])
139:                end if
140:            end for
141:            sng1← list(set(sng1))
142:            if (len(sng1 = 1) and sng∈ sng1) or (len(sng1)> 1 and (sng=«N» or
    pov = 1)) then
143:                left ← left−2
144:            else
145:                return [«он», «писали» ]
146:            end if
147:        else if left−4 ≥ 0 and l[left−4] = «,» and l[left−1] = «,» then
148:            sng1← [ ]
149:            r← [ ]
150:            res1← l[left−3].[pos, singular, cow]
151:            for from i = 0 to len(res1)−1 do
152:                if res1[i][0] =part then
153:                    sng1← sng1 + list(res[i][1])
154:                    r← r + check(l[left −3 : left−1])
155:                end if
156:            end for
157:            sng1← list(set(sng1))
158:            if (len(sng1 = 1) and sng∈ sng1 and «Y»∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y»∈ r) then
159:                left ← left−3
160:            else
161:                return [«он», «писали» ]
162:            end if
163:        else if left−5 ≥ 0 and l[left−5] = «,» and l[left−1] = «,» then
164:            sng1← [ ]
165:            r← [ ]
```

Алгоритм 17 – Продолжение алгоритма 16

```
166:            res1← l[left−4].[pos, singular, cow]
167:            for from i = 0 to len(res1)−1 do
168:                if res1[i][0] =part then
169:                    sng1← sng1 + list(res[i][1])
170:                    r← r + check(l[left −4 : left−1])
171:                end if
172:            end for
173:            sng1← list(set(sng1))
174:            if (len(sng1 = 1) and sng∈ sng1 and «Y»∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y»∈ r) then
175:                left ← left−4
176:            else
177:                return [«он», «писали» ]
178:            end if
179:        else if left−6 ≥ 0 and l[left−6] = «,» and l[left−1] = «,» then
180:            sng1← [ ]
181:            r← [ ]
182:            res1← l[left−5].[pos, singular, cow]
183:            for from i = 0 to len(res1)−1 do
184:                if res1[i][0] =part then
185:                    sng1← sng1 + list(res[i][1])
186:                    r← r + check(l[left −5 : left−1])
187:                end if
188:            end for
189:            sng1← list(set(sng1))
190:            if (len(sng1 = 1) and sng∈ sng1 and «Y»∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y»∈ r) then
191:                left ← left−5
192:            else
193:                return [«он», «писали» ]
194:            end if
195:        else if left−7 ≥ 0 and l[left−7] = «,» and l[left−1] = «,» then
196:            sng1← [ ]
197:            r← [ ]
198:            res1← l[left−6].[pos, singular, cow]
199:            for from i = 0 to len(res1)−1 do
200:                if res1[i][0] =part then
201:                    sng1← sng1 + list(res[i][1])
202:                    r← r + check(l[left −6 : left−1])
203:                end if
204:            end for
205:            sng1← list(set(sng1))
206:            if (len(sng1 = 1) and sng∈ sng1 and «Y»∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y»∈ r) then
207:                left ← left−6
208:            else
209:                return [«он», «писали» ]
210:            end if
```

Алгоритм 18 – Продолжение алгоритма 17

```
211:        else if left−8 ≥ 0 and l[left−8] = «,» and l[left−1] = «,» then
212:            sng1← [ ]
213:            r← [ ]
214:            res1← l[left−7].[pos, singular, cow]
215:            for from i = 0 to len(res1)−1 do
216:                if res1[i][0] =part then
217:                    sng1← sng1 + list(res[i][1])
218:                    r← r + check(l[left −7 : left−1])
219:                end if
220:            end for
221:            sng1← list(set(sng1))
222:            if (len(sng1 = 1) and sng∈ sng1 and «Y»∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y»∈ r) then
223:                left ← left−7
224:            else
225:                return [«он», «писали» ]
226:            end if
227:        else if left−9 ≥ 0 and l[left−9] = «,» and l[left−1] = «,» then
228:            sng1← [ ]
229:            r← [ ]
230:            res1← l[left−8].[pos, singular, cow]
231:            for from i = 0 to len(res1)−1 do
232:                if res1[i][0] = part then
233:                    sng1← sng1 + list(res[i][1])
234:                    r← r + check(l[left −8 : left−1])
235:                end if
236:            end for
237:            sng1← list(set(sng1))
238:            if (len(sng1 = 1) and sng∈ sng1 and «Y»∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y»∈ r) then
239:                left ← left−8
240:            else
241:                return [«он», «писали» ]
242:            end if
243:        else
244:            if left−9 ≥ 0 and l[left−1] = «,» and end = (−1) then ▷ Поиск начала
    перечисления
245:                sng1← [ ]
246:                r ← [ ]
247:                res1← l[left−9].[pos, singular, cow]
248:                res1← list(set(res1))
249:                for from i = 0 to len(res1)−1 do
250:                    if res1[i][0] = part then
251:                        sng1← sng1 + list(res[i][1])
252:                        r ← r + list(l[left −9 : left−1])
253:                    end if
254:                end for
255:                sng1← list(set(sng1))
```

Алгоритм 19 – Продолжение алгоритма 18

```
256:                if (len(sng1) = 1 and sng∈ sng1 and «Y» ∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y» ∈ r) then
257:                    left ← left−9
258:                    end ← 1
259:                else if len(sng1)> 0 and «Y» ∈ sng1 then
260:                    return [«он», «писали» ]
261:                end if
262:            end if
263:            if left−8 ≥ 0 and l[left−1] = «,» and end = (−1) then
264:                sng1← [ ]
265:                r ← [ ]
266:                res1← l[left−8].[pos, singular, cow]
267:                res1← list(set(res1))
268:                for from i = 0 to len(res1)−1 do
269:                    if res1[i][0] = part then
270:                        sng1← sng1 + list(res[i][1])
271:                        r ← r + list(l[left −8 : left−1])
272:                    end if
273:                end for
274:                sng1← list(set(sng1))
275:                if (len(sng1) = 1 and sng∈ sng1 and «Y» ∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y» ∈ r) then
276:                    left ← left−8
277:                    end ← 1
278:                else if len(sng1)> 0 and «Y» ∈ sng1 then
279:                    return [«он», «писали» ]
280:                end if
281:            end if
282:            if left−7 ≥ 0 and l[left−1] = «,» and end = (−1) then
283:                sng1← [ ]
284:                r ← [ ]
285:                res1← l[left−7].[pos, singular, cow]
286:                res1← list(set(res1))
287:                for from i = 0 to len(res1)−1 do
288:                    if res1[i][0] = part then
289:                        sng1← sng1 + list(res[i][1])
290:                        r ← r + list(l[left −7 : left−1])
291:                    end if
292:                end for
293:                sng1← list(set(sng1))
294:                if (len(sng1) = 1 and sng∈ sng1 and «Y» ∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y» ∈ r) then
295:                    left ← left−7
296:                    end ← 1
297:                else if len(sng1)> 0 and «Y» ∈ sng1 then
298:                    return [«он», «писали» ]
299:                end if
300:            end if
```

Алгоритм 20 – Продолжение алгоритма 19

```
301:            if left−6 ≥ 0 and l[left−1] = «,» and end = (−1) then
302:                sng1← [ ]
303:                r ← [ ]
304:                res1← l[left−6].[pos, singular, cow]
305:                res1← list(set(res1))
306:                for from i = 0 to len(res1)−1 do
307:                    if res1[i][0] = part then
308:                        sng1← sng1 + list(res[i][1])
309:                        r ← r + list(l[left −6 : left−1])
310:                    end if
311:                end for
312:                sng1← list(set(sng1))
313:                if (len(sng1) = 1 and sng∈ sng1 and «Y» ∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y» ∈ r) then
314:                    left ← left−6
315:                    end ← 1
316:                else if len(sng1)> 0 and «Y» ∈ sng1 then
317:                    return [«он», «писали» ]
318:                end if
319:            end if
320:            if left−5 ≥ 0 and l[left−1] = «,» and end = (−1) then
321:                sng1← [ ]
322:                r ← [ ]
323:                res1← l[left−5].[pos, singular, cow]
324:                res1← list(set(res1))
325:                for from i = 0 to len(res1)−1 do
326:                    if res1[i][0] = part then
327:                        sng1← sng1 + list(res[i][1])
328:                        r ← r + list(l[left −5 : left−1])
329:                    end if
330:                end for
331:                sng1← list(set(sng1))
332:                if (len(sng1) = 1 and sng∈ sng1 and «Y» ∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y» ∈ r) then
333:                    left ← left−5
334:                    end ← 1
335:                else if len(sng1)> 0 and «Y» ∈ sng1 then
336:                    return [«он», «писали» ]
337:                end if
338:            end if
339:            if left−4 ≥ 0 and l[left−1] = «,» and end = (−1) then
340:                sng1← [ ]
341:                r ← [ ]
342:                res1← l[left−4].[pos, singular, cow]
343:                res1← list(set(res1))
```

Алгоритм 21 – Продолжение алгоритма 20

```
344:                for from i = 0 to len(res1)−1 do
345:                    if res1[i][0] = part then
346:                        sng1← sng1 + list(res[i][1])
347:                        r ← r + list(l[left −4 : left−1])
348:                    end if
349:                end for
350:                sng1← list(set(sng1))
351:                if (len(sng1) = 1 and sng∈ sng1 and «Y» ∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y» ∈ r) then
352:                    left ← left−4
353:                    end ← 1
354:                else if len(sng1)> 0 and «Y» ∈ sng1 then
355:                    return [«он», «писали» ]
356:                end if
357:            end if
358:            if left−3 ≥ 0 and l[left−1] = «,» and end = (−1) then
359:                sng1← [ ]
360:                r ← [ ]
361:                res1← l[left−3].[pos, singular, cow]
362:                res1← list(set(res1))
363:                for from i = 0 to len(res1)−1 do
364:                    if res1[i][0] = part then
365:                        sng1← sng1 + list(res[i][1])
366:                        r ← r + list(l[left −3 : left−1])
367:                    end if
368:                end for
369:                sng1← list(set(sng1))
370:                if (len(sng1) = 1 and sng∈ sng1 and «Y» ∈ r) or (len(sng1)> 1 and
    (sng=«N» or pov = 1) and «Y» ∈ r) then
371:                    left ← left−3
372:                    end ← 1
373:                else if len(sng1)> 0 and «Y» ∈ sng1 then
374:                    return [«он», «писали» ]
375:                end if
376:            end if
377:            if left−2 ≥ 0 and l[left−1] = «,» and end = (−1) then
378:                sng1← [ ]
379:                res1← l[left−2].[pos, singular, cow]
380:                res1← list(set(res1))
381:                for from i = 0 to len(res1)−1 do
382:                    if res1[i][0] = part then
383:                        sng1← sng1 + list(res[i][1])
384:                    end if
385:                end for
386:                sng1← list(set(sng1))
387:                if (len(sng1) = 1 and sng∈ sng1) or (len(sng1)> 1 and (sng=«N» or
    pov = 1)) then
```

Алгоритм 22 – Продолжение алгоритма 21

```
388:                left ← left−2
389:                end ← 1
390:            else if len(sng1)> 0 and «Y» ∈ sng1 then
391:                return [«он», «писали» ]
392:            end if
393:        end if
394:    end if
395: end while
```

Итак, после выполнения окончания выполнения алгоритма 22 в переменной left будет записан индекс списка, соответствующий левой границе перечисления; в переменной right — индекс, соответствующий правой границе перечисления; в переменной sng — число перечисляемых личных глаголов (данный фрагмент будет выполнен только в том случае, если имеет место перечисление личных глаголов).

Далее, в зависимости от числа заменяем перечисление одним личным глаголом в таком же числе. Мы решили заменять перечисление личных глаголов на слово *«учил»* или *«учили»* в зависимости от числа.

Так, *«Он **писал диплом, ел и спал»*** будет преобразовано в *«Он **учил»***.

Данный этап описан алгоритмом 23.

Алгоритм 23 – Продолжение алгоритма 22

```
396:    if sng = «Y» then
397:        l← l[ : left]+ [«учил»] + l[ right +1 : ]
398:    else if sng = «N» then
399:        l← l[ : left]+ [«учили»] + l[ right +1 : ]
400:    end if
401: end if
```

Итак, на этом завершается обработка перечисления. Далее будет описан алгоритм, как же проверяются предложения без перечислений.

3.3.2. Непосредственная проверка предложения

Проанализировав предложения, состоящие из существительных, местоимений, личных глаголов и (или) инфинитивов, мы обнаружили, что важно проанализировать словосочетания длины 2, 3 или 4 во всём списке. Словосочетания бОльшей длины декомпозируются на составляющие указанной длины.

Так, предложение *«Он тебя видит и слышит»* разбивается на части: *«Он*

тебя видит» (длины 3) и *«слышит»* (длины 1). Союз «и» в данном случае выполняет роль связки.

Нами были написаны функции two() (алгоритм 24), three() (алгоритм 26) и four(), которые анализируют словосочетания соответствующей длины.

Алгоритм 24 – Анализ словосочетаний длины 2

```
1: function TWO(lst)
2:     result ← []                                   ▷ Результат выполнения проверки
3:     pov ← (−1)          ▷ Идентификатор наличия повелительного наклонения
4:     sng ← []
5:     ans1← []
6:     for from q = 0 to len(lst)−2 do
7:         res_l ← lst[q].[pos, singular, cow]
8:         k← 0
9:         uns← (−1)
10:        for from i = 0 to len(res_l)−1 do
11:            if res_l[i][0] = «1» or res_l[i][0] = «b» then
12:                if res_l[i][1] = «N» then
13:                    uns← 1
14:                end if
15:            end if
16:        end for
17:        res_r ← lst[q + 1].[pos, singular, cow]
18:        for from i = 0 to len(res_r)−1 do
19:            if res_r[i][0] = «5» then
20:                k← k+1
21:            else if res_r[i][0] = «1» or res_r[i][0] = «b» then
22:                if res_r[i][1] = «N» then
23:                    uns ← 1
24:                end if
25:            end if
26:        end for
27:        if k > 1 then
28:            pov← 1
29:        end if
30:        for from i = 0 to len(res_l)−1 do
31:            for from j = 0 to len(res_r)−1 do
32:                res ← (ans).simple_rules.(res_l[i])(res_r[j])   ▷ Записываем
   ответ для данного словосочетания из таблицы simple_rules, в которой хранятся
   правила для словосочетаний длины 2
33:                if len(res) > 0 then
34:                    for from r = 0 to len(res)−1 do
35:                        ans← ans + res[r][0]
```

Здесь, как в случае и с личными глаголами, дополнительный анализ текста производится для решения проблемы омографии.

Алгоритм 25 – Продолжение алгоритма 24

```
                    end for
                end if
            end for
        end for
        ans1← list(set(ans1))
        if pov= 1 and uns = (−1) then        ▷ Если правый операнд может быть в повелительном наклонении, а слева стоит подлежащее в единственном числе
            if «N» ∈ ans1 then
                result← result + [«N»]
            else if «Y» ∈ ans1 then
                result ← result + [«Y»]
            else
                result ← result + [«E»]                ▷ «E» — незнакомое сочетание
            end if
        else
            if «Y» ∈ ans1 then
                result ← result + [«Y»]
            else if «N» ∈ ans1 then
                result← result + [«N»]
            else
                result ← result + [«E»]                ▷ «E» — незнакомое сочетание
            end if
        end if
    end for
    return result
end function
```

Подобным образом устроена проверка словосочетаний длины 3.

Алгоритм 26 – Анализ словосочетаний длины 3

```
function THREE(lst)
    result ← [ ]
    for from i = 0 to len(lst)−3 do
        res1 ← lst[i].[pos, singular, cow]
        res2 ← lst[i + 1].[pos, singular, cow]
        res3 ← lst[i + 2].[pos, singular, cow]
        for from i1 = 0 to len(res1)−1 do
            for from i2 = 0 to len(res2)−1 do
                for from i3 = 0 to len(res3)−1 do
                    res ← (ans).add3_r.(res1[i1])(res1[i2])(res3[i3])        ▷ Записываем ответ для данного словосочетания из таблицы add3_r, в которой хранятся правила для словосочетаний длины 3
                    if len(res)> 0 then
                        for from r = 0 to len(res)−1 do
                            result← result + res[r][0]
                        end for
```

Алгоритм 27 – Продолжение алгоритма 26

```
                end if
            end for
        end for
    end for
    result ← list(set(result))
    if len(result)> 0 then
        if «N» ∈ result then
            return [«N»] ▷ Если на каком-то этапе сработало правило ошибки, не проверяем дальше
        end if
    end if
end for
return result
end function
```

Аналогично устроена проверка словосочетаний длины 4.

Алгоритм 28 – Анализ словосочетаний длины 3

```
function FOUR(lst)
    result ← []
    for from i = 0 to len(lst)−4 do
        res1 ← lst[i].[pos, singular, cow]
        res2 ← lst[i + 1].[pos, singular, cow]
        res3 ← lst[i + 2].[pos, singular, cow]
        res4 ← lst[i + 3].[pos, singular, cow]
        for from i1 = 0 to len(res1)−1 do
            for from i2 = 0 to len(res2)−1 do
                for from i3 = 0 to len(res3)−1 do
                    for from i4 = 0 to len(res4)−1 do
                        res ← (ans).add4_r.(res1[i1])(res1[i2])(res3[i3])(res4[i4]) ▷ Записываем ответ для данного словосочетания из таблицы add4_r, в которой хранятся правила для словосочетаний длины 4
                        if len(res)> 0 then
                            for from r = 0 to len(res)−1 do
                                result← result + res[r][0]
                            end for
                        end if
                    end for
                end for
            end for
        end for
        result ← list(set(result))
        if len(result) then
            if «N» ∈ result then
                return [«N»] ▷ Если на каком-то этапе сработало правило ошибки, не проверяем дальше
```

Алгоритм 29 – Продолжение алгоритма 28

```
            end if
        end if
    end for
    return result
end function
```

3.4. Примеры работы алгоритма

4. Заключение

Тут будет заключение

Список литературы

[1] **Дзюбенко, В.А. Согласование единственного и множественного числа в русском предложении**: бакалаврская диссертация: 03.03.01 / Дзюбенко Василий Александрович. – Долгопрудный, 2020. – 20 с.

[2] **Журавлёв, Ю.И. Дискретный анализ. Формальные системы и алгоритмы**: Учебное пособие / Ю.И. Журавлёв, Ю.А. Флёров, Н.М. Вялый – М.: ООО Контакт Плюс, 2010. – 336 с.

[3] **Чесебиев, И. А. Компьютерное распознавание и порождение речи**: монография. – Москва: Спорт и Культура-2000, 2008. – 125 с.

[4] **Comrie, B. Language universals and linguistic typology: Syntax and morphology**. – University of Chicago press, 1989.

[5] **Conway, D. An algorithmic approach to English pluralization** // Proceedings of the Second Annual Perl Conference. – 1998.

[6] **The world atlas of language structures** / M. Haspelmath [and others]. – Oxford Univ. Press, 2005.

[7] LanguageTool — Проверка грамматики и стилистики [Электронный ресурс] — https://languagetool.org/ru

ПРИЛОЖЕНИЕ А

Структура таблиц базы данных

Таблица 1 – words

№	Имя столбца	Тип данных	Комментарий
1	word		слово в нижнем регистре

ПРИЛОЖЕНИЕ В

Система правил для анализа словосочетаний из двух слов

Обозначения:

- prt — часть речи: 1 — существительное, 5 — личный глагол, 6 — инфинитив, *b* — местоимение;
- sing — число: «N» — множественное, «Y» — единственное, «–» — не определено (инфинитивы);
- cow — падеж: 1 — именительный, 2 — родительный, 3 — дательный, 4 — винительный, 5 — творительный, 6 — предложный, «–» — не определено (личные глаголы и инфинитивы);
- ans — ответ: «Y» — верно, «N» — неверно.

_r и _l — указатель правого и левого операнда соответственно.

Таблица 2 – Система правил для словосочетаний из двух слов

№	prt_l	sing_l	cow_r	prt_r	sing_r	cow_l	ans	Пример
1	b	N	1	5	N	–	Y	мы делали
2	1	Y	1	5	N	–	N	собака лаяли
3	1	Y	1	5	Y	–	Y	самолёт летит
4	b	Y	1	5	Y	–	Y	я делаю
5	6	–	–	1	Y	4	Y	делать дело
6	5	Y	–	6	–	–	Y	хочет есть
7	6	–	–	b	Y	2	Y	знать его
8	6	–	–	1	N	5	Y	гордиться детьми
9	b	Y	1	6	–	–	Y	я есть
10	b	N	1	6	–	–	Y	вы есть
11	5	N	–	6	–	–	Y	пришли договориться
12	b	N	1	5	Y	–	N	мы писал
13	5	Y	–	b	Y	2	Y	победил меня

14	5	Y	–	b	N	1	N	вздохнул мы
15	5	Y	–	1	N	1	N	вздохнул люди
16	5	Y	–	1	Y	1	Y	бежал человек
17	5	N	–	1	Y	1	N	бегут собака
18	5	N	–	1	N	1	Y	бежали собаки
19	5	N	–	b	Y	1	N	бежали я
20	5	N	–	b	N	1	Y	бежали мы
21	6	–	–	b	N	2	Y	укусить нас
22	6	–	–	5	N	–	N	видеть хотели
23	6	–	–	5	Y	–	N	быть хотел
24	1	Y	3	6	–	–	Y	чуду быть
25	1	N	1	5	N	–	Y	люди делали
26	1	N	1	5	Y	–	N	люди учил
27	1	N	3	6	–	–	Y	праздникам быть
28	b	Y	1	5	N	–	N	я делали
29	b	Y	2	5	N	–	Y	меня ранили
30	b	Y	3	5	N	–	Y	мне позвонили
31	5	N	–	b	Y	2	Y	переиграли меня
32	5	N	–	b	N	2	Y	позвали нас
33	5	Y	–	1	N	4	Y	вижу кошек
34	5	N	–	1	N	2	Y	позвали друзей
35	1	Y	1	1	Y	3	Y	человек собаке