Аннотация

Тут будет аннотация

Содержание

1. Введение

Тут будет введение

2. Описание подхода

2.1. 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      ▷ Если последним элементом полученного
    списка оказался пробел
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.

2.2. Обработка предложения

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

В основе предложенного нами подхода лежит гипотеза, согласно которой одно 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 [2].

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 [1]. Чтобы избежать противоречивости разработанной системы, мы включили лишь те правила, которые встречаются на практике, а не перебрали все возможные комбинации используемых нами параметров.

2.2.1. 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. Механизм работы функции comma() описывает алгоритм 2.

Алгоритм 2 – Обработка перечислений

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

Для начала инициализируем переменные, затем находим индекс вхождения «и» в список (при условии, что в списке есть «и»). После этого анализируем слова, находящиеся в окрестности слова «и»:

Алгоритм 3 – Продолжение алгоритма 2

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

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

Заметим, что в случае перечисления с союзом «и» 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 «и». 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

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

Алгоритм 5 – Продолжение алгоритма 4

```
65:                     else if «Y»∈ r then
66:                         left ← id1−6          ▷ Инициализировали границу левого операнда «и»
67:                         break
68:                     end if
69:                 end if
70:             end for
71:         end if
72:         if id1−5 ≥ 0 and left= −1 and «,»∉ l[id1−5 : id1] then
73:             for from i = 0 to len(resl5)−1 do
74:                 if resl5[i][0] =part then
75:                     r← check(l[id1−5 : id1])
76:                     if «N»∈ r then
77:                         return [«он», «писали» ]
78:                     else if «Y»∈ r then
79:                         left ← id1−5
80:                         break
81:                     end if
82:                 end if
83:             end for
84:         end if
85:         if id1−4 ≥ 0 and left= −1 and «,»∉ l[id1−4 : id1] then
86:             for from i = 0 to len(resl4)−1 do
87:                 if resl4[i][0] =part then
88:                     r← check(l[id1−4 : id1])
89:                     if «N»∈ r then
90:                         return [«он», «писали» ]
91:                     else if «Y»∈ r then
92:                         left ← id1−4
93:                         break
94:                     end if
95:                 end if
96:             end for
97:         end if
98:         if id1−3 ≥ 0 and left= −1 and «,»∉ l[id1−3 : id1] then
99:             for from i = 0 to len(resl3)−1 do
100:                 if resl3[i][0] =part then
101:                     r← check(l[id1−3 : id1])
102:                     if «N»∈ r then
103:                         return [«он», «писали» ]
104:                     else if «Y»∈ r then
105:                         left ← id1−3
106:                         break
107:                     end if
108:                 end if
109:             end for
110:         end if
```

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

Алгоритм 6 – Продолжение алгоритма 5

```
111:            if id1−2 ≥ 0 and left= −1 and «,»∉ l[id1−2 : id1] then
112:                for from i = 0 to len(resl2)−1 do
113:                    if resl2[i][0] =part then
114:                        r← check(l[id1−2 : id1])
115:                        if «N»∈ r then
116:                            return [«он», «писали» ]
117:                        else if «Y»∈ r then
118:                            left ← id1−2
119:                            break
120:                        end if
121:                    end if
122:                end for
123:            end if
124:            if id1−1 ≥ 0 and left= (−1) then
125:                for from i = 0 to len(resl1)−1 do
126:                    if l[i][0] =part then
127:                        left ← id1−1
128:                        break
129:                    end if
130:                end for
131:            end if
132:        end if
133:        Алгоритм 7
134:        Алгоритм 11
135:    end if
136:    Алгоритм 12
137: 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 «и» связывает только два сочетания.
- Союз «и» используется для перечисления 3 и более словосочетаний.

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
33:                 end if
34:             end if
```

Алгоритм 8 – Продолжение алгоритма 7

```
        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.

Алгоритм 12 – Продолжение алгоритма 6

```
1: if part=«5» then
2:     if id1−7 ≥ 0 and left= (−1) and «,»∉ l[id1−7 : id1] then
3:     end if
4: end if
```

Список литературы

[1] **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.: ил.

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

ПРИЛОЖЕНИЕ А

Система правил для анализа словосочетаний из двух слов

Таблица 1 – Система правил для словосочетаний из двух слов

№	prt_l	sing_l	cow_r	prt_r	sing_r	cow_l	ans	example
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