Обработка запятых

Галицкий Борис Васильевич

28 мая 2021 г.

Описание алгоритма

Запятые могут участвововать как разделители частей одного простого предложения, так и как разделители простых предложений в составе сложного. Таким образом, они могут как влиять, так и не влиять на согласование числа в предложении.

Будем считать, что на вход подаётся правильное с точки зрения орфографии и пунктуации предожение (задачи проверки орфографии и пунктуации были успешно решены, например, корпорацией Microsoft).

Согласно нашей ***гипотезе***, можно разработать систему правил, позволяющую обработать все запятые, возможно, упростив предложение (проверив во время упрощения согласование единственного и множественного числа в упрощаемых частях). При этом, единственное и множественное число в исходном предложении будут согласованы тогда и только тогда, когда они будут согласованы в упрощённом предложении.

Ключевые этапы реализации:

1. Формализация правил, когда запятые влияют на согласование единственного и множественного числа.

2. Применение данной системы правил для идентификации запятых, влияющих на согласование единственного и множественного числа. Упрощение предложения на основе этой системы правил (с проверкой согласования единственного и множественного числа в частях предложения).

3. Замена оставшихся запятых на точки (разбиение предложения на части без запятых, к каждой из которых можно применить уже реализованный алгоритм проверки согласования единственного и множественного числа в предложении без запятых).

Данная задача в рамках нашей работы рассматривается только для существительных, местоимений, инфинитивов и личных глаголов.

Одним из важных случаев, когда запятые влияют на согласование единственного и множественного числа, являются перечисления. В них используются союзы «И» или «ИЛИ». Однако, уже в самом начале возникают проблемы:

- Слова какой части речи перечисляются?

- Какие слова в этой части предложения являются зависимыми?

- А если одним из перечисляемых объектов является словосочетание, нет ли там ошибки в согласовании числа?

Предложенный нами алгоритм даст ответы на эти вопросы.

Перечисление инфинитивов

Как было сказано выше, существует проблема идентификации: слова какой части речи перечисляются? Итак, для начала рассмотрим случаи, когла перечисляются инфинитивы.

Для начала находим союз «И» или «ИЛИ». Дальше смотрим на операнд слева. Если имеет место перечисление инфинитивов, то могут быть следующие варианты:

- инфинитив: «Он любил есть, пить и танцевать».
- инфинитив + существительное: «Он любил есть, пить сок и танцевать».
- инфинитив + существительное + существительное: «Они могли понимать его, оценить игру слов и сделать выводы».
- инфинитив + местоимение: «Он мог гордится ими, корить себя и винить друзей».
- инфинитив + инфинитив + существительное: «Он мог любить свободу, любить делать добро и любить себя».

Определив, что это точно одна из данных конструкций,

1. Проверим в ней согласование числа.
2. Проверим в правом операнде согласование числа.
3. Внутри текущего предложения в цикле, двигаясь влево, будем искать запятые и соответствующие конструкции, тем самым определив границы фрагмента с перечислением инфинитивов.

Если оказалось, что в какой-то из частей не согласовано время, то возвращаем значение «N» — не согласовано.

Если же на данном этапе всё корректно, то фрагмент, содержащий перечисление инфинитивов, заменяем на инфинитив (мы решили заменять перечисление инфинитивов на «знать»). Так, в результате работы описанного алгоритма предложение «Они могли понимать его, оценить игру слов и сделать выводы» будет преобразовано в предложение «Они могли знать».

Перечисление личных глаголов

Как и в ситуации с инфинитивами, сначала ищем союз «И» или «ИЛИ», затем смотрим на операнд слева. Если имеет место перечисление личных глаголов, то возможны следующие варианты:

- личный глагол: «Он пришёл, увидел и победил».
- личный глагол + существительное: «Он купил дом, посадил дерево и вырастил сына».
- личный глагол + инфинитив: «Он учил, любил учиться и хотел научить».
- личный глагол + инфинитив + существительное: «Он мечтал построить дом, собирался посадить дерево и хотел найти клад».
- личный глагол + местоимение: «Он ценил его, жалел себя и ненавидел их».

- личный глагол + инфинитив + местоимение: «Он хотел увидеть её, собирался жалеть себя и видел их».
- личный глагол + существительное + существительное: «Он услышал тебя, оценил игру слов и сделал выводы».
- личный глагол + местоимение + существительное + существительное: «Он знал его, оценил его игру слов и сделал выводы».

Определив, что это точно одна из данных конструкций,

1. Проверим в ней согласование числа.
2. Проверим в правом операнде согласование числа.
3. Внутри текущего предложения в цикле, двигаясь влево, будем искать запятые и соответствующие конструкции, тем самым определив границы фрагмента с перечислением инфинитивов.

Если оказалось, что в какой-то из частей не согласовано время, то возвращаем значение «N» — не согласовано. Здесь, в отличие от инфинитивов, нужно проверять, что перечисляются глаголы только единственного или только множественного числа.

Если же на данном этапе всё корректно, то фрагмент, содержащий перечисление инфинитивов, заменяем на личный глагол (мы решили заменять перечисление инфинитивов на «знал» или «знали» в зависимости от того, личные глаголы единственного или множественного числа перечисляются). Так, в результате работы описанного алгоритма предложение «Он хотел увидеть её, собирался жалеть себя и видел их» будет преобразовано в предложение «Он знал».

Замечание 1. *Пока наша задача заключается лишь в согласовании числа, потому вполне может получиться упрощённое предложение вида «Она знал». Однако, в данном случае оба слова в единственном числе, потому ошибки в согласовании числа нет.*

Algorithm 1 Добавление пробелов между знаками пунктуации

```
1:  function SPACE(str1)
2:      str2 ← ««                ▷ В этой переменной будет храниться преобразованная строка
3:      l = len(str)
4:      for i ∈ {1, 2, . . . , l} do
5:          if str1[i] ∈ {«.»; «,» } then
6:              str2 ← str2 + « »
7:          end if
8:          str2 ← str2 + str1[i]
9:      end for
10:     return str2
11: end function
```

Algorithm 2 Обработка запятых

```
function COMMA(l)                                              ▷ l — подготовленная строка
    l ← l.split()
    llen ← len(l)                                                  ▷ llen — длина списка l
    h ← −1                                                ▷ h — индекс начала заменяемой части
    t ← −1                                                 ▷ t — индекс конца заменяемой части
    while 'и' ∈ l do
        id1 ← l.index('и')                                ▷ в id1 хранится индекс вхождения 'и'
        if l[id1 − 2] = ',' then
            res1 ← l[id1 − 1].(pos, singular, cow)      ▷ res1 содержит часть речи, число и падеж слова
            for i ∈ res1 do
                part = i.pos                                          ▷ Записываем в part часть речи
                h ← id1 − 1                          ▷ Левая граница заменяемого перечисления
                t ← id1                             ▷ Правая граница заменяемого перечисления
                if part = '6' then                          ▷ Обработка перечисления инфинитивов
                    if id1 + 1 ⩽ llen then
                        res2 ← l[id1 + 1].(pos, singular, cow)          ▷ Анализируем слово, расположенное следом за 'и'
                        if res2.pos = part then          ▷ Если оно оказалось инфинитивом
                            t ← id1 + 1
                        end if
                        idpr ← id1 − 2                       ▷ Индекс потенциальной запятой
                        idd ← False                                ▷ Индикатор перечисления
                        while idpr ⩾ 0 do                ▷ Пока не дошли до начала слова
                            if l[idpr] = ',' then
                                res2 ← l[idpr + 1].(pos, singular, cow)
                                if res2.pos = part then
                                    l[idpr] ← «‘»      ▷ Заменяем «,» на «’», чтобы дальше не находить эту запятую
                                    h ← idpr          ▷ Сдвигаем «голову» заменяемой части
                                    idpr ← idpr − 2
                                    idd ← True
                                end if
                            end if
                        end while
                    end if
                end if
            end for
        end if
    end while
end function
```

Algorithm 3 Пример алгоритма

```
l
```