

ЛАБОРАТОРНАЯ РАБОТА №4

Проверка проективности предложений

Цель. Освоить методы проверки проективности предложений, построение деревьев зависимостей и отрезочных представлений в виде системы составляющих. Научиться анализировать синтаксическую структуру предложения с использованием библиотеки Natasha.

4.1. Теоретические сведения

Проективность – свойство синтаксического дерева, при котором все дуги (зависимости) между словами не пересекаются при линейном расположении слов. Дерево называется проективным, если для любого узла все его потомки образуют непрерывный отрезок в предложении.

Пример проективного дерева:

(признание (международное) (программ (образовательных) (вузов (российских))))

Если дуги пересекаются, дерево является **непроективным**.

4.2. Выполнение работы

1. Убедитесь, что установлены библиотеки:

– Natasha

– rumorphy2

2. Пример кода для анализа предложения:

```
from natasha import Segmenter, NewsEmbedding, NewsMorphTagger,  
NewsSyntaxParser, Doc
```

```
def norm(txt):  
    _, x = map(int, txt.split('_'))  
    return x
```

```
segmenter = Segmenter()  
emb = NewsEmbedding()  
morph_tagger = NewsMorphTagger(emb)  
syntax_parser = NewsSyntaxParser(emb)
```

```
def analyze_sentence(text):  
    doc = Doc(text)  
    doc.segment(segmenter)  
    doc.tag_morph(morph_tagger)  
    doc.parse_syntax(syntax_parser)  
    sent = doc.sents[0]
```

```
words = {}  
pos = {}  
tree = {0: []}  
root = None
```

токены без пунктуации + компактная нумерация позиций

```
tokens = [token for token in sent.tokens if token.pos != 'PUNCT']
```

```

for i, token in enumerate(tokens, start=1):
    norm_id = norm(token.id)
    words[norm_id] = token.text
    pos[norm_id] = i          # новая позиция в предложении без
                               пунктуации
    if token.rel == 'root':
        root = norm_id
        tree[norm_id] = []

# строим дерево по norm_id
for token in tokens:
    norm_id = norm(token.id)
    norm_head_id = norm(token.head_id)
    # крепим только к тем головам, которые есть в tree (0 или
    непунктуационные слова)
    if norm_head_id in tree:
        tree[norm_head_id].append(norm_id)

# убираем искусственный корень 0
if 0 in tree:
    tree.pop(0)

# Левая скобочная запись
def lrep(a):
    s = '(' + words[a]
    for child in tree[a]:
        s += ' ' + lrep(child)
    s += ')'
    return s

# Правая скобочная запись
def rrep(a):
    s = '('
    for child in tree[a]:
        s += rrep(child) + ' '
    s += words[a] + ')'
    return s

# Проверка проективности: по поддеревьям и по компактным позициям
pos
def is_projective(tree, pos):
    from functools import lru_cache

    @lru_cache(None)
    def subtree(node):
        nodes = [node]
        for child in tree[node]:
            nodes.extend(subtree(child))
        return nodes

    for node in tree:
        nodes = subtree(node)
        inds = sorted(pos[n] for n in nodes)
        if inds[-1] - inds[0] + 1 != len(inds):
            return False
    return True

projective = is_projective(tree, pos)

print(f"Предложение: {text}")
print(f"Слова: {words}")
print(f"Корень: {root}")
print(f"Дерево: {tree}")
print(f"Левая скобочная запись: {lrep(root)}")

```

```

print(f"Правая скобочная запись: {rrep(root)}")
print(f"Проективно: {'Да' if projective else 'Нет'}")
sent.syntax.print()
print()

# Примеры предложений
sentences = [
    "Мама мыла раму.",
    "Я люблю программирование.",
    "Кошка сидит на столе.",
    "Он читает интересную книгу.",
    "Дети играют в парке."
]

for sent in sentences:
    analyze_sentence(sent)

```

Пример вывода:

Предложение: Мама мыла раму.
 Слова: {1: 'Мама', 2: 'мыла', 3: 'раму'}
 Корень: 1
 Дерево: {1: [2], 2: [3], 3: []}
 Левая скобочная запись: (Мама (мыла (раму)))
 Правая скобочная запись: (((раму) мыла) Мама)
 Проективно: Да

```

    Мама
  ┌ мыла
  │   └ раму obj
  │
  └ .

```

Предложение: Я люблю программирование.
 Слова: {1: 'Я', 2: 'люблю', 3: 'программирование'}
 Корень: 2
 Дерево: {1: [], 2: [1, 3], 3: []}
 Левая скобочная запись: (люблю (Я) (программирование))
 Правая скобочная запись: ((Я) (программирование) люблю)
 Проективно: Да

```

    Я nsubj
  ┌   └ люблю
  │   └─┬─ программирование obj
  │   └─┴─ . punct
  └─┬─
  └─┬─
  └─┬─

```

Предложение: Кошка сидит на столе.
 Слова: {1: 'Кошка', 2: 'сидит', 3: 'на', 4: 'столу'}
 Корень: 2
 Дерево: {1: [], 2: [1, 4], 3: [], 4: [3]}
 Левая скобочная запись: (сидит (Кошка) (столу (на)))
 Правая скобочная запись: ((Кошка) ((на) столу) сидит)
 Проективно: Да

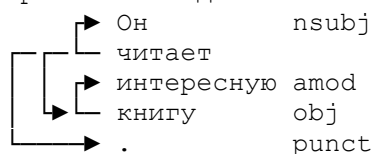
```

    Кошка nsubj
  ┌   └ сидит
  │   └─┬─ на case
  │   └─┬─ столу obl
  │   └─┴─ . punct
  └─┬─
  └─┬─
  └─┬─

```

Предложение: Он читает интересную книгу.
 Слова: {1: 'Он', 2: 'читает', 3: 'интересную', 4: 'книгу'}
 Корень: 2
 Дерево: {1: [], 2: [1, 4], 3: [], 4: [3]}
 Левая скобочная запись: (читает (Он) (книгу (интересную)))
 Правая скобочная запись: ((Он) ((интересную) книгу) читает)

Проективно: Да



Предложение: Дети играют в парке.

Слова: {1: 'Дети', 2: 'играют', 3: 'в', 4: 'парке'}

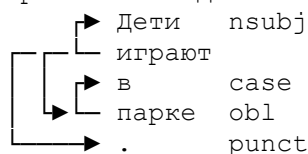
Корень: 2

Дерево: {1: [], 2: [1, 4], 3: [], 4: [3]}

Левая скобочная запись: (играют (Дети) (парке (в)))

Правая скобочная запись: ((Дети) ((в) парке) играют)

Проективно: Да



4.3. Задания к лабораторной работе №4

1. Выберите пять предложений из вашего датасета.
2. Для каждого предложения:
 - Постройте дерево зависимостей с помощью Natasha.
 - Получите левую и правую скобочные записи.
 - Проверьте проективность предложения.
3. Сравните результаты автоматической проверки с ручным анализом.

Требования к отчету к ЛР №4

1. Листинг к заданиям 1-2.
2. **Результаты анализа** для пяти предложений:
 - Исходное предложение.
 - Дерево зависимостей.
 - Левая и правая скобочные записи.
 - Результат проверки на проективность.
3. **Выводы** по работе:
 - Какие предложения оказались проективными, а какие – нет?
 - С чем связана непроективность (если есть)?
4. Готовность ответить на вопросы по выполненной работе.