САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Филологический факультет

Кафедра математической лингвистики

**ОТЧЕТ О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОГО ЗАДАНИЯ ПО КУРСУ МОРФО-СИНТАКСИЧЕСКИЕ АНАЛИЗАТОРЫ**

Студентки 1 курса магистратуры Шевелевой Анны

Направление: Филология

Программа: Прикладная и экспериментальная лингвистика

Профиль: Компьютерная лингвистика и интеллектуальные технологии

Санкт–Петербург

2018

**Введение**

Целью данной работы является подключение теггера Брилла для гибридного морфоанализатора NLTK4Russian, его обучение на выборках из корпусов НКРЯ[[1]](#footnote-1) и OpenCorpora[[2]](#footnote-2) и его тестирование на данных соревнований Dialogue Evaluation 2017[[3]](#footnote-3).

Для обучения морфоанализатора были взяты материалы проекта «Открытый корпус» и выборки из Национального корпуса русского языка. Для тестирования были использованы тексты, которые являлись тестовыми для участников соревнований Dialogue Evaluation 2017, а именно корпусы текстов новостного сайта Lenta, социальной сети ВКонтакте и тексты писателя Олега Зайончковского (JZ). Стоит отметить, что данный материал представляет собой тексты с различными жанровыми и стилистическими особенностями.

Таким образом, **задачи** настоящей работы заключаются в следующем:

1. Установка и исследование основных режимов работы морфоанализатора NLTK4Russian.
2. Интеграция теггера Брилла в морфоанализатор NLTK4Russian.
3. Обучение теггера и его тестирование на выбранных данных.
4. Обработка результатов и анализ допущенных ошибок.
5. Сравнение полученных данных для использованных корпусов.

**Методы** исследования: в ходе эксперимента были использованы корпусный метод и методы машинного обучения.

**Принцип работы морфоанализатора NLTK4RUSSIAN**

Анализатор NLTK4RUSSIAN – лингвистический комплекс для исследования русскоязычных корпусов текстов, разрабатывающийся на кафедре математической лингвистики СпбГУ. Главная задача – улучшение морфологического анализа с использованием уже существующих инструментов. NLTK4RUSSIAN называется гибридным морфоанализатором, так как объединяет в себе алгоритмы NLTK[[4]](#footnote-4) и PyMorphy2[[5]](#footnote-5).

NLTK – библиотека Python, которая предназначена для работы с текстом на всех уровнях языка. Средства библиотеки позволяют проводить как базовые операции (лемматизация, токенизация, стемминг), так и многоуровневый анализ с применением алгоритмов машинного обучения. NLTK также позволяет проводить морфологический анализ и разметку. В числе инструментов теггеры, использующие словарные данные, теггер на основе скрытых марковских моделей, анализаторы с применением контекстной информации и др. Важная особенность NLTK заключается в том, что в большинстве случаев используется размеченный корпус, что делает лингвистический комплекс независимым от языка текста для анализа. Одновременно с этим среда NLTK не адаптирована для конкретных языков, она использует универсальную метаинформацию. NLTK4RUSSIAN, таким образом, является адаптацией NLTK к русскоязычным текстам, создаваемой путем интеграции лингвонезависимых алгоритмов NLTK и морфологического анализатора для русского языка.

PyMorphy2 – анализатор русских текстов, который использует морфологический словарь OpenCorpora. Для этой системы не нужен обучающий корпус, все необходимые словоформы описаны в словаре. Для редких или несуществующих форм работает предсказатель, который выводит предполагаемый тег по префиксу и по концу слова. Для совпадающих словоформ или форм с неоднозначностью выводятся все возможные интерпретации, вероятность того или иного варианта рассчитывается не на основе контекста, но по данным словаря. Можно сказать, что NLTK4RUSSIAN предлагает альтернативное решение неоднозначности: если PyMorphy2 выводит несколько вариантов разбора, то теггер пользуется возможностями анализа соседних словоформ от NLTK. Таким образом, NLTK4RUSSIAN объединяет достоинства NLTK и PyMorphy2: текст анализируется с помощью анализатора PyMorphy2, для дизамбигуации вызывается теггер NLTK. При этом система может работать как в гибридном режиме, так и с использованием NLTK и PyMorphy2 по отдельности.

NLTK4RUSSIAN работает в 3 режимах, для переключения каждого из них при вызове морфоанализатора необходимо указать специальный параметр-название "-n":

— гибридный (PyMorphy2 + NLTK);

— PyMorphy2 (PyMorphyTagger);

— униграммный, биграммный и триграммный теггеры NLTK.

Гибридный режим – это режим с использованием PyMorphy2 и биграммного теггера NLTK nltk.tag.sequentials. При вызове анализатора указывается файл с текстом, размеченный корпус для обучения и название файла, который будет создан в ходе работы. Файл с неразмеченным текстом читается построчно. Вызывается функция MorphAnalyzer() от PyMorphy2, при обнаружении неоднозначности запускается биграммный теггер NgramTagger c соответствующим параметром. Режим получил название "PyMorphyContext", при запуске анализатора нужно указать параметр "-n" "pmcontext".

В режиме PyMorphy2 вызывается функция MorphAnalyzer() от PyMorphy2, и, как в гибридном режиме, выполняется разметка на основе словарных данных. Но и в отличие от PyMorphyContext, неоднозначность снимается с помощью оценки вероятности. Параметр рассчитывается на основе данных корпуса OpenCorpora: для всех слов со снятой неоднозначностью вычисляется, сколько раз словоформа была отмечена данным тегом. На основе этих частот и выводится условная вероятность тега (параметр «score»). (http://pymorphy2.readthedocs.io/en/latest/user/guide.html)

Последний режим анализирует текст на основании контекста – предшествующих словоформе тегов слов. Униграммный теггер (UnigramTagger) ищет подходящий вариант разбора в обучающем корпусе, биграммный (BigramTagger) анализирует тег предыдущего слова в тексте и ищет соответствие в обучающем корпусе, триграммный (TrigramTagger) составляет контекст из разбора слова и двух предыдущих слов и ищет подходящие разборы в обучающем корпусе.

Для вызова морфоанализтора в одном из этих режимов при запуске нужно указать соответственно "-n 1gram", "2gram" или "3gram".

**Теггер Брилла: краткая характеристика и принципы работы**

Алгоритм Брилла с управляемым обучением был впервые описан Эриком Бриллом из Массачусетского технологического института в 1994 г., а еще через год был описан алгоритм Брилла с неуправляемым обучением.

Обучение начинается с присвоения каждому слову в обучающей выборке наиболее частотного для данной словоформы (только одного!) тега. Затем происходит сравнение с правильной разметкой, формулируются правила, которые дают наилучший результат. Действие правила – изменение приписанного тега. Производятся несколько итераций до тех пор, пока не будет достигнут запланированный эффект (полное отсутствие улучшений, запланированная степень близости к правильной разметке, запланированное максимальное число правил) Правила бывают двух видов: лексические и контекстные.

Для работы алгоритма с неуправляемым обучением используются корпус текстов, не содержащий предварительной разметки, и словарь. Происходит предварительная разметка текста в соответствии со словарем, с указанием всех вариантов. Анализируя корпус текстов при помощи словаря, мы можем обнаружить, что среди всех слов, которые встречаются после слова "the" (и для которых в словаре указан только один возможный тэг), чаще всего встречаются слова с тэгом NN. Исходя из этого, мы можем сформулировать следующее правило: «Заменять тег MD\_NN\_VB (т.е. сохраняющий три варианта разметки) на NN после слова «the»» Таким образом, первичная разметка дает неоднозначно размеченный текст. Затем выводятся правила вида: «Заменить тег Х на тег Y в контексте C, где Х является последовательностью из двух или более тегов, а Y – один тег, такой что Y ∈ Х».

Шаблоны выводимых правил:

«Заменить тэг Х на тэг Y, если ...

* предшествующая словоформа маркирована тэгом Z
* предшествующая словоформа есть W
* последующая словоформа маркирована тэгом Z
* последующая словоформа есть W…»

В пакете NLTK теггер, работающий на основе алгоритма Брилла, вызывается через nltk.tag.brill.

# Подготовительная работа

Обучение морфоанализатора NLTK4Russian выполнено на размеченных текстах со снятой омонимией. Обучающие материалы включают выборки из Национального корпуса русского языка, а также подкорпус OpenCorpora объёмом около 1,7 млн. словоупотреблений.

Для файлов с тестовой выборкой потребовалась дополнительная обработка, связанная с извлечением слов для разметки. Эта задача была реализована при помощи специального скрипта[[6]](#footnote-6). На вход поступает файл с тестовой выборкой (Lenta, VK или JZ), в котором из каждой строки извлекаются имеющиеся словоформы. В результате мы получаем отдельный файл с извлечёнными формами без разметки, с которым далее работает морфоанализатор.

**Тестирование и результаты**

Для обучения и тестирования теггера Брилла на выбранных данных был разработан специальный скрипт (см. Приложение 2). Тестирование проводилось в трех вариантах:

* без подключения теггера Брилла; обучение n-граммного теггера (PMContext tagger) на подкорпусах НКРЯ и Open Corpora и оценка полученного результата на текстах Lenta, VK и JZ (Таблица 1)
* тестирование теггера Брилла при обучении на подкорпусе Open Corpora и оценка полученного результата на текстах Lenta, VK и JZ (Таблица 2)
* тестирование теггера Брилла при обучении на подкорпусе НКРЯ и оценка полученного результата на текстах Lenta, VK и JZ (Таблица 3)

По окончании работы программы выводится статистика по примененным шаблонам (templates) для вывода правил, а также в папке со скриптом сохраняются следующие файлы:

* *tagged\_text.txt* – результат разметки тестового файла
* *tagger\_result.txt* – сводка статистических показателей по проведенному тестированию – результаты оценивания на текстах Lenta, VK и JZ и полный список выведенных и примененных правил.

*Таблица 1. Результаты тестирования без подключения теггера Брилла*

|  |  |  |  |
| --- | --- | --- | --- |
| Материал обучения | Lenta | VK | JZ |
| OpenCorpora | *93 %* | *90, 8 %* | *92, 7 %* |
| НКРЯ | *98, 4 %* | *97, 9 %* | *98 %* |

*Таблица 2.* *Результаты тестирования при обучении теггера Брилла на подкорпусе* *Open Corpora*

|  |  |  |  |
| --- | --- | --- | --- |
| Мах кол-во правил | Lenta | VK | JZ |
| 2 | ***90, 7 %*** | *89, 4 %* | ***91, 8 %*** |
| 5 | *88, 4 %* | *88, 4 %* | *90, 5 %* |
| 10 | ***84, 9 %*** | ***85, 6 %*** | ***88, 3 %*** |
| 20 | *90, 5 %* | ***90, 5 %*** | *91 %* |
| 50 | *90, 1 %* | *90 %* | *90, 6 %* |
| 100 | *89, 6 %* | *89, 5 %* | *90 %* |
| 200 | *89, 2 %* | *89, 4 %* | *89, 5 %* |
| 300 | *89, 1 %* | *89 %* | *89 %* |

*Таблица 3. Результаты тестирования при обучении теггера Брилла на подкорпусе НКРЯ*

|  |  |  |  |
| --- | --- | --- | --- |
| Мах кол-во правил | Lenta | VK | JZ |
| 2 | ***95, 8 %*** | ***96, 2 %*** | ***96, 8 %*** |
| 5 | *93, 5 %* | *95 %* | *95, 5 %* |
| 10 | *89, 7 %* | *92 %* | *92, 8 %* |
| 20 | *85, 4 %* | *88, 9 %* | *90 %* |
| 50 | *76, 7 %* | *81, 8 %* | *85 %* |
| 100 | *69, 4 %* | *76 %* | *77, 8 %* |
| 200 | *62 %* | *68, 4 %* | *70 %* |
| 300 | ***58, 3 %*** | ***63, 5 %*** | ***66, 2 %*** |

**Заключение**

По итогам тестирования можно прийти к следующим выводам:

1. Наибольшая точность разметки достигается без использования теггера Брилла в качестве вспомогательного (backoff) теггера.
2. Лучше всего n-граммный теггер размечает новостные тексты.
3. При использовании теггера Брилла и его обучении на материалах НКРЯ наилучший результат достигается наименьшим количеством правил, с увеличением количества правил качество разметки постепенно снижается.
4. При обучении теггера на материалах Open Corpora, для текстов из социальной сети Вконтакте определилось иное оптимальное количество правил – 20: меньшее или большее количество правил ведет к снижению точности разметки. Тем не менее, для новостных и авторских текстов по-прежнему лучше всего подходит наименьшее количество правил.
5. Тексты писателя JZ являются более однородными по стилю, поэтому показывают лучшие результаты по сравнению с другими текстами.
6. Использование подкорпуса НКРЯ в качестве материала для обучения приводит к более точным результатам разметки.

**Приложения**

*Приложение 1. Скрипт для получения тестовых выборок.*

with open('VK\_RNC.txt', 'r', encoding='utf-8') as f:

o = []

for t in f:

if t == '\n':

o.append('\n')

t = t.split('\t')

try:

o.append(t[1])

except IndexError:

continue

with open('VK\_TEST.txt', 'w', encoding='utf-8') as j:

print(' '.join(o), file=j)

*Приложение 2. Подключение теггера Брилла в NLTK4RUS*

import nltk

from nltk4russian.tagger import PMContextTagger

from nltk4russian.util import read\_corpus\_to\_nltk

from nltk.tag.brill import \*

import nltk.tag.brill\_trainer as bt

import opencorpora

#brill templates

Template.\_cleartemplates()

templates = nltk.tag.brill.fntbl37()

#чтение подкорпуса Open Corpora + фильтрация пустых предложений

corpus = opencorpora.CorpusReader('annot.opcorpora.no\_ambig.xml')

sents\_OC = list(filter(lambda x: len(x), corpus.iter\_tagged\_sents()))

#чтение подкорпуса НКРЯ (media1.tab, можно подключить другие из папки nltk4russian/data):

with open('media1.tab', encoding='utf-8') as media:

sents\_RNC = list(read\_corpus\_to\_nltk(media))

#чтение подкорпуса LENTA

with open('LENTA\_OC.txt', encoding='utf-8') as LENTA:

sents1 = list(read\_corpus\_to\_nltk(LENTA))

#чтение подкорпуса VK

with open('VK\_OC.txt', encoding='utf-8') as VK:

sents2 = list(read\_corpus\_to\_nltk(VK))

#чтение подкорпуса JZ

with open('JZ\_OC.txt', encoding='utf-8') as JZ:

sents3 = list(read\_corpus\_to\_nltk(JZ))

tagger = PMContextTagger(sents\_OC) #выбираем обучающий корпус sents\_RNC или sents\_OC

tagger = bt.BrillTaggerTrainer(tagger, templates, trace=3)

tagger = tagger.train(sents\_RNC, max\_rules=10) #задаем max кол-во правил

tagger.print\_template\_statistics(printunused=False) #таблица статистических параметров

#читаем и разбиваем на токены файл, который размечаем обученным теггером (взят VK\_TEST без разметки)

inFile = nltk.word\_tokenize(open('VK\_TEST.txt', mode='r', encoding='utf-8').read())

#вывод в файл tagged\_text.txt размеченного текста

with open('tagged\_text.txt', mode='w', encoding='utf-8') as tagged:

print(tagger.tag(inFile), file=tagged)

#вывод в файл tagger\_result.txt результатов оценки на разных подкорпусах, списка правил)

with open('tagger\_result.txt', mode='w', encoding='utf-8') as result:

print('Оценка результатов по выборке LENTA: ', tagger.evaluate(sents1) \* 100, file=result)

print('Оценка результатов по выборке VK: ', tagger.evaluate(sents2) \* 100, file=result)

print('Оценка результатов по выборке JZ: ', tagger.evaluate(sents3) \* 100, file=result)

print('\n', 'Перечень выведенных правил: ', file=result)

for i in range(len(tagger.rules())):

print(str(i+1)+'.', tagger.rules()[i], file=result)

print('Итого (правил): ', len(tagger.rules()), file=result)

1. Национальный корпус русского языка: ruscorpora.ru [↑](#footnote-ref-1)
2. Проект «Открытый Корпус»: opencorpora.org [↑](#footnote-ref-2)
3. Конференция «Диалог» является крупнейшей российской научной конференцией по компьютерной лингвистике, в рамках которой ежегодно проводятся соревнования Dialogue Evaluation. Одной из задач соревнования Dialogue Evaluation 2017 являлся морфологический анализ. [↑](#footnote-ref-3)
4. Natural Language Toolkit: http:// www.nltk.org/ [↑](#footnote-ref-4)
5. http://pymorphy2.readthedocs.org/en/latest/ [↑](#footnote-ref-5)
6. Автор - Алексей Мельник; см. приложение 1. [↑](#footnote-ref-6)