таггер шугнанских глаголов

Юрий Макаров, Макс Меленченко
первокурсники ОП ФиКЛ Школы лингвистики

Что мы делали

Преодолевали трудности:

- отсутствие единой орфографии: нужен был орфографический конвертер
- словарь Карамшоева не очень качественно оцифрован, нужно было приводить его в порядок
- разнообразие глагольных основ
- нерегулярность образования форм
- и прочие морфологические нюансы

Masc	Fem	Plur
andoʒd	andūvʒ	andūvʒ
indoʒd	indöʒč	indöʒč

примеры основ перфекта

А зачем всё это?

- есть проект по изучению шугнанского языка;
  в его рамках разрабатывается **корпус**, которому нужен **парсер**
- таггер — первый шаг на пути к парсеру

Словарь

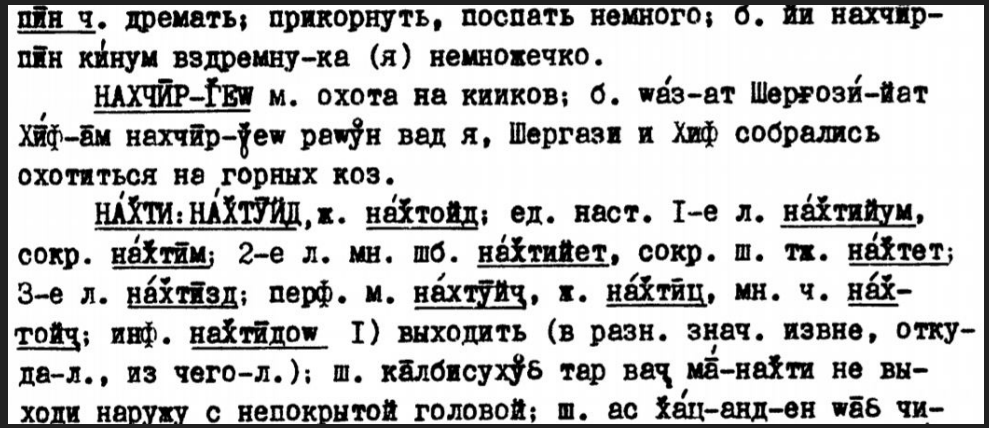
пӣн ч. дремать; прикорнуть, поспать немного; б. йи нахчӣр-пӣн ки́нум вздремну-ка (я) немножечко.

НАХЧӢР-ӀЕW м. охота на кииков; б. ва́з-ат Шерӏози́-йат Хи́ф-ӓм нахчӣр-ӯew pawӯн вад я, Шергази и Хиф собрались охотиться на горных коз.

НА́ӮТИ:НА́ӮТӮЙД, ж. на́ӯтойд; ед. наст. I-е л. на́ӯтийум, сокр. на́ӯтӣм; 2-е л. мн. шб. на́ӯтийет, сокр. ш. тж. на́ӯтет; 3-е л. на́ӯтӣзд; перф. м. на́хтӯйч, ж. на́ӯтӣц, мн. ч. на́ӯтойч; инф. наӯтӣдow I) выходить (в разн. знач. извне, откуда-л., из чего-л.); ш. кӓлбисухӯб тар вач ма́-наӯти не выходи наружу с непокрытой головой; ш. ас ӯа́ц-анд-ен wӓб чи-

Словарь: что мы с ним делали

- вручную вычитывали ошибки распознавания
- с помощью RegEx меняли систему глосс
- писали скрипты, извлекающие нужные для глагольной БД формы из распознанных словарных статей
- добавляли лексические значения к части глаголов



пӣн ч. дремать; прикорнуть, поспать немного; б. йи нахчӣр-пӣн кӣнум вздремну-ка (я) немножечко.

НАХЧӢР-ГЕw м. охота на кииков; б. wáз-ат Шерҕозӣ-йат Хӣф-ӑм нахчӣр-ӯew pawӯн вад я, Шергази и Хиф собрались охотиться на горных коз.

НАӐТИ:НАӐТӮЙД, ж. наӐтойд; ед. наст. I-е л. наӐтийум, сокр. наӐтӣм; 2-е л. мн. шб. наӐтийет, сокр. ш. тж. наӐтет; 3-е л. наӐтӣзд; перф. м. наӐтӯйч, ж. наӐтӣц, мн. ч. наӐтойч; инф. наӐтӣдow I) выходить (в разн. знач. извне, откуда-л., из чего-л.); ш. кӑлбисухӯб тар вач мӑ-наӐти не выходи наружу с непокрытой головой; ш. ас ӐаӐц-анд-ен wӑб чи-

Словарь: скрипт

```
def extract(gloss,place,source,sample):
  if gloss in source:
     n = source.index(gloss)
     if n == -1:
        pass
     else:
        sample[place] = source[n+1]
  return sample
```

1. Находит помету (*gloss*) в строке (*source*) вида:
   Praes ФИНУХС/ФИНУХЦ:
   Past ФИНУХСТ/ФИНУХЦТ;
   Perf финухсч/финухцч;
   Inf бинухстоw/фИНУХЦТОW
   медленно делать что-л.
2. Вычисляет её положение и добавляет её на место (*place*) в ячейке словаря словоформу после пометы.
3. sample = [*'0','1','2','3','4','5','6','7','8','9'*]

Таблица глагольных форм и основ

PRS.M	PRS.F	PRS.3SG	PST.M	PST.FPL	PRF.M	PRF.F	PRF.PL	INF.M	INF.F	lemma
0	1	2	3	4	5	6	7	8	9	10
θāw	—	θöd	—	—	θuvʒ θūðʒ	θic	θavʒ θoðʒ	θid	—	—
naɤti	—	—	naχtūjd	—	naχtūjʒ	naɤtīc	—	naɤtīd	—	*make cry*
pirmir parmir	—	pirmirt parmirt	pirmūd parmūd	pirmod parmod	pirmūɣʒ parmūɣʒ	pirmīɣʒ parmīɣʒ	pirmoɣʒ parmoɣʒ	pirmird pirmīd parmirt parmīd	—	—

Таблица глагольных форм и основ

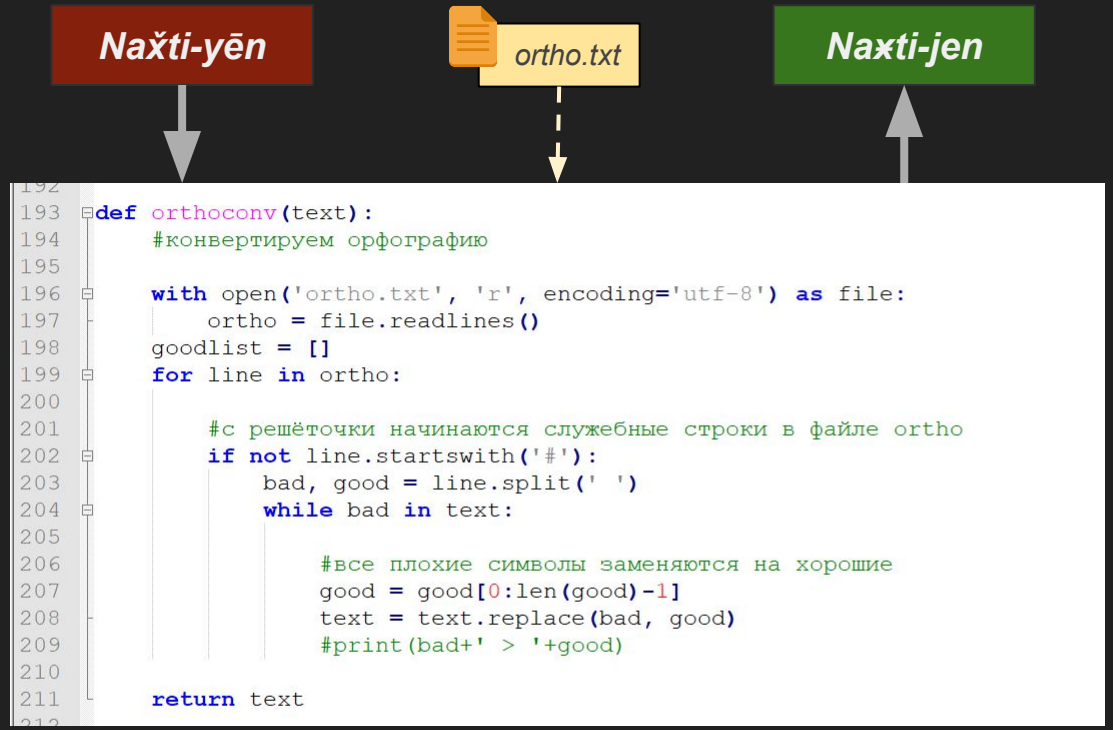
wuqsen	0	0	wuqsent/waqsen/aqsen	0	wuqsenč/waqsenč/aqsenč	0	0	wuqsent/waqsent/aqsent	0	vomit
wöj/woj	0	0	wöjt/wojt	0	wöjč/wojč	0	0	wöjt/wojt	0	cry_loudly
wug	0	0	wugd	0	wögu	0	0	wögd	0	—
gārð	0	gārðd	gāɤt	0	gāɤč	0	0	gāɤt	0	—
gilnɛɣd	0	0	galnɛɣd	0	gilnɛɣʒ	0	0	gilnɛɣd/ʁalnɛɣd	0	—
girz/čirz	0	0	girzd/čirzd	0	girzč/čirzʒ	0	0	1	0	grieve/mourn
gīr	0	0	gīrt	0	gīrč	0	0	gīrt	0	agree
gumbun	0	0	gumbunt	0	gumbönč	0	0	gumbönt	0	—
gumor/gimor	0	0	gumort/gimort	0	gumorč/gimorč	0	0	gumort/gimort	0	authorize/delegate
dawun	0	0	dawunt	0	dawönč	0	0	dawönt	0	—
dak	0	0	dakt	0	dakč	0	0	dakt	0	dry_(about_throat)
darjov	0	darjovd/darjɛvd	darjovd/darjɛvd	0	darjovʒ/darjɛvʒ	0	0	darjovd/darjɛvd	0	—
dām	0	0	dāmt	0	dāmč	0	0	dāmt	0	rotate
deð	0	deðd	ded	0	deðʒ	0	0	ded	0	fan_the_fire
di/dɛ	0	0	dīt/dɛt	0	dīč/š/tž/dɛč	0	0	1	0	enter
diven	0	0	divent	0	divenč	0	0	divent	0	beat
dives	0	divest	diviɤt	0	diviɤč	0	0	diviɤt	0	blow
diwen/dawen	0	0	diwent/dawent	0	diwenč/dawenč	0	0	1	0	show_up/appear

Код: глобальная структура

Код: конвертер орфографии

1.транскрипция из словаря (Д. Карамшоев)	Ӯӯ	Ӱӱ	Фф	Хх	Ҳҳ	Цц	Ҙҙ
2.иранистическая транскрипция (Д. Карамшо	Ūū	Ůů	Ff	Xx	Ҳҳ	Cc	Ҙҙ
3.транскрипция Плунгяна (см. 1, кроме і, ū, ё	Ūū	Ёё	Фф	Хх	Ҳҳ	Цц	Ҙҙ
4.Зарубин (см.2)	Ūū	Ůů	Ff	Xx	Ҳҳ	Cc	Ҙҙ
5.Бахтибеков (см.1)	Ӯӯ	Ӱӱ	Фф	Хх	Ҳҳ	Цц	Ҙҙ
6.Umass (см.2, кроме много чего)	Uu	Oo	Ff	Xx	Ҳҳ	Tsts	Dzdz
7.Расторгуева (Эдельман) (см.2)	Ūū	Ůů	Ff	Xx	Ҳҳ	Cc	Ҙҙ
8.Morgenstierne (см.2, кроме Éé и Óó)	Ūū	Ůů	Ff	Xx	Ҳҳ	Cc	Ҙҙ
9.Сердюченко Пахалина (см.2, кроме Ɛɛ)	Ūū	Ůů	Ff	Xx	Ҳҳ	Cc	Ҙҙ
10.Windfuhr (см.2)	Ūū	Ůů	Ff	Xx	Ҳҳ	Cc	
	29	30	31	32	33	34	3
IPA	uː	ů	f	χ	x?	ts	dz
наша	ū	ö	f	χ	x	c	ʒ
как в слове…	mūn 'ябл	göl 'немо	fawt 'сме	χūb 'хорс	xīn 'синий	cem 'глаз	awʒ 'ба

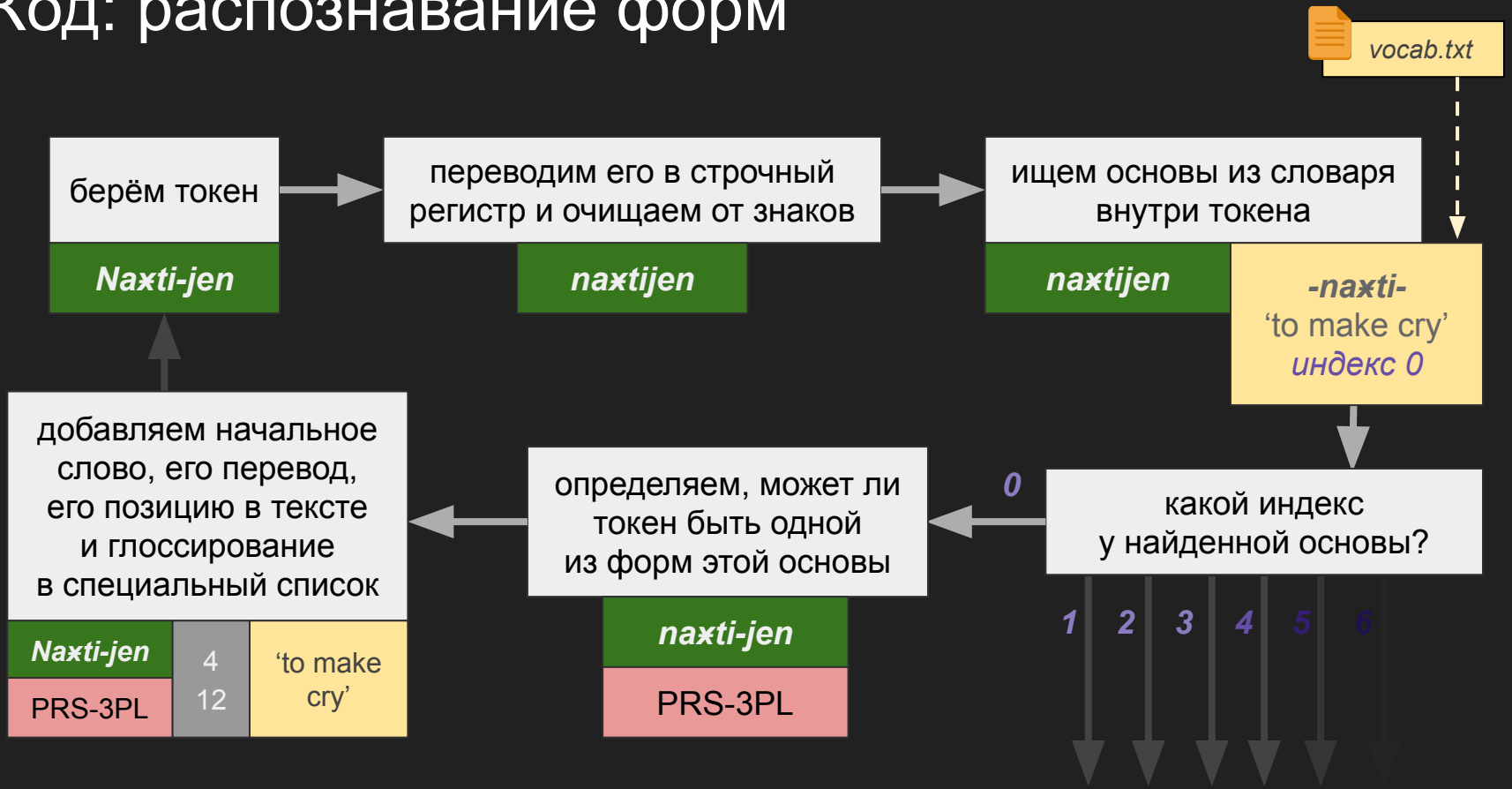
Код: конвертер орфографии




```python
def orthoconv(text):
    #конвертируем орфографию

    with open('ortho.txt', 'r', encoding='utf-8') as file:
        ortho = file.readlines()
    goodlist = []
    for line in ortho:

        #с решёточки начинаются служебные строки в файле ortho
        if not line.startswith('#'):
            bad, good = line.split(' ')
            while bad in text:

                #все плохие символы заменяются на хорошие
                good = good[0:len(good)-1]
                text = text.replace(bad, good)
                #print(bad+' > '+good)

    return text
```

```
0 Tɕ C
1 tɕ č
2 Dʑ Ǯ
3 dʑ ǯ
4 Sh Š
5 SH Š
6 sh š
7 #noncompounds
8 Æ Ā
9 æ ā
0 Б B
1 б b
2 B V
3 в v
4 Г G
5 г g
6 Ғ ʁ
7 ғ ʁ
8 Ɣ ʁ
9 ɣ ʁ
0 Ґ ¥
1 ґ ¥
2 Γ ¥
3 Y ¥
4 Д D
```

Код: распознавание форм



```
589
590 def formdefinition(word, stem, y):
591     #это коммутатор, который выявляет, какая из основ найдена в слове, и перенаправляет программу к нужной функции (для verbfind)
592     #на вход принимается слово (токен текста) word, предполагаемая основа stem и индекс этой основы в матрице словаря y
593
594     #attributes — список глоссирований, который является результатом работы formdefinition
595     #здесь нужен именно список, потому что найденных глоссирований может быть больше одного
596     attributes = []
597
598     #если переданный индекс y равен индексу одного из стемов, идём к этому стему
599     #сначала проверяем на обычные формы
600     if word.startswith(stem):
601         if y == praesmasc or y == praesfemn:
602             attributes.append(isitpraestem(word, stem, y))
603         elif y == praes3sg:
604             attributes.append(isitpraes3sg(word, stem))
605         elif y == pastmasc or y == pastfepl:
606             attributes.append(isitpasttnse(word, stem, y))
607         elif y == perfmasc or y == perffemn or y == perfplur:
608             attributes.append(isitperftnse(word, stem, y))
609         elif y == infimasc or y == infifemn:
610             attributes.append(isitinfinite(word, stem, y))
611
612     #проверяем на стяжённые формы
613     if y == contracted:
614         attributes.append(isitcontract(word, stem))
615
616     #проверяем на отрицательные и условные формы
617     #если слово начинается на -ma- или -na-, а при удалении этих приставок найденная основа остаётся внутри слова
618     #то вызываем formdefinition ещё раз, но уже для слова без приставки (ДА!!! РЕКУРСИЯ!!! КРУТО ПРАВДА???)
619     #например, naxtijum не пройдёт, потому что стем naxti не сохраняется, если убрать -na-
620     #а вот navuðj пройдёт, потому что без -na- перфектный стем vuðj сохраняется, и formdefinition вызовется уже для vuðj, чтобы
        определить его форму
621     if word.startswith('ma') or word.startswith('na'):
622
623         #если без приставки стем не сохраняется, мы предполагаем, что это стяжённая форма, и присваиваем индекс contracted
624         if not word[2:].startswith(stem):
625             if word.endswith('m') or word.endswith('et') or word.endswith('en'):
626                 y = contracted
```

parser.py | vocab.txt | ortho.txt | graphic.py | text.txt | text_sample.txt | output.txt

```
374
375 def isitpraestem(word, stem, y):
376     #функция проверяет, не является ли слово глагольной формой, образованной от praestem (для formdefinition)
377
378     #определяем род стема
379     if y == praesmasc:
380         gender = 'M'
381     if y == praesfemn:
382         gender = 'F'
383
384     #по умолчанию значение глоссирования — False, что сигнализирует о том, что формам этого стема токен не соответствует
385     attribute = False
386
387     #добавим к концу стема -j-, если он оканчивается на гласный
388     stem = jification(stem)
389
390     #проверяем, не является ли слово формой 3 лица ед. ч. презенса
391     if word.endswith('d') or word.endswith('t'):
392         flexias = ('d', 't')
393         for flexia in flexias:
394             if word == stem+flexia:
395                 attribute = 'PRS.'+gender+'-'+'3SG'
396
397     #проверяем, не является ли слово одной из других форм презенса
398     if attribute == False:
399         flexias = {'um': '1SG', 'i': '2SG', 'ām': '1PL', 'et': '2PL', 'en': '3PL'}
400         for flexia in flexias:
401             if word == stem+flexia:
402                 attribute = 'PRS.'+gender+'-'+flexias[flexia]
403
404     #проверяем, не является ли слово отглагольным существительным
405     if attribute == False:
406         if word == stem+'iʒ':
407             attribute = 'AGENT_NOUN'
408
409     #возвращаем глоссирование (False, если ничего не найдено)
410     return attribute
411
412
```

Python file | length : 48 138 | lines : 888 | Ln : 418 | Col : 5 | Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

parser.py | vocab.txt | ortho.txt | graphic.py

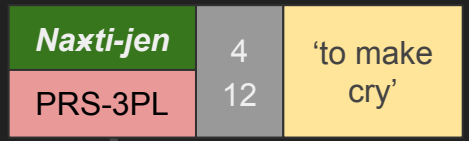
```
697         #очистим слово от мусора — пробелов и висящих в начале и конце знаках препинаний — и будем использовать его для определения формы
698         wordnew = wordclean(word)
699 
700         #если то, что осталось после очистки — нормальное слово, а не пустая строка, то продолжаем
701         if realword(wordnew):
702 
703             #glossfoundsinword — переменная, которая для каждой итерации цикла собирает найденные в токене глоссирования, а потом пакует
                их и передаёт в glossboxes
704             glossfoundsinword = []
705 
706             #для каждой лексемы в словаре:
707             for x in range(len(vocab)):
708 
709                 #для каждого индекса стема в лексеме:
710                 for y in range(len(vocab[x])):
711 
712                     #если этот стем недостаточный, не трогаем его
713                     if not nedostatochny_stem(vocab[x][y]):
714 
715                         #для каждой вариации стема:
716                         for z in range(len(vocab[x][y])):
717 
718                             #если мы нашли наш стем в токене
719                             if vocab[x][y][z] in wordnew:
720 
721                                 #отправляем токен, стем и его индекс на поиск формы и получаем attributes — глоссирование
722                                 attributes = formdefinition(wordnew, vocab[x][y][z], y)
723 
724                                 #если какие-то глоссирования нашлись, т.е. список не пустой:
725                                 if not attributes == []:
726                                     for attribute in attributes:
727 
728                                         #если можно указать лемму, то указываем её
729                                         if vocab[x][lemma][0] != '—':
730                                             attribute = vocab[x][lemma][0]+' > '+attribute
731 
732                                         #складываем все глоссирования в glossfoundsinword
733                                         glossfoundsinword.append(attribute)
734 
```

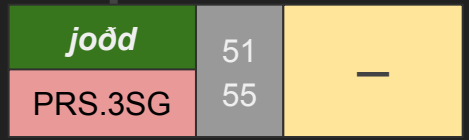
Python file | length : 48 138 lines : 888 | Ln : 16 Col : 137 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

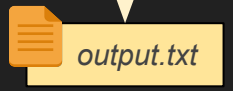
Вывод

Naxti-yēn tar māraka-yēn.
Xābatā yid yůrxak yoðd xu di boǰak fukaθ biɣent.

Naxti-jen (make_cry > PRS-3PL)
tar (crash > PRS.3SG) māraka-jen.
Xābatā jid jörxak **joðd** (PRS.3SG) χu **di** (enter > PRS.3SG) boǯak fukaθ **biɣent** (PRS-3SG, INF).







Что получилось

github.com/**iurmak/shughni**