In [1]:

```
import os
import re
import pandas as pd
import random
```

Compute accuracy for the guesser.

Make sure that andi.guesser.hfst exists and is in the current repository.

To create andi.guesser.hfst: andi/guesser make andi.guesser.hfst

In [2]:

```
def guess_word(word):
    output = os.popen(f"echo {word} | hfst-guess andi.guesser.hfst -n 60").read()
    parses = []
    for el in output.split('\n'):
        parses.append(re.sub('\[GUESS_CATEGORY=\w+\]', '', ':'.join(el.split('\t'
    if parses == ['']:
        parses = []
    return parses
```

In [19]:

```
def count_guessed(standard, print_=False):
    guessed = 0
    not_analysed = 0
    analysed = 0
    g_pos_tags = 0
    g_tags = 0
    for line in standard:
        guesses = guess_word(line[0])
        if guesses:
            analysed +=1
            if f'{line[0]}:{line[1]}' in guesses:
                guessed += 1
            else:
                if print_:
                    print('FAIL')
                    print(f'standard: {line[0]}:{line[1]}')
                    print(guess_word(line[0]))
                    print()
                    print()

            guessed_pos_tags = [re.findall('<.*>', l)[0] for l in guesses]
            true_pos_tag = re.findall('<.*>', line[1])[0]
            if true_pos_tag in guessed_pos_tags:
                g_pos_tags += 1

            guessed_tags = [re.findall('<\w+>(<.*>)', l)[0] for l in guesses]
            true_tag = re.findall('<\w+>(<.*>)', line[1])
            if true_tag and true_tag[0] in guessed_tags:
                g_tags += 1
        else:
            not_analysed += 1


    print('analysed: ', analysed)
    print('not analysed: ', not_analysed)
    print('coverage: ', analysed/(analysed+not_analysed))

    print('guessed: ', guessed)
    print('accuracy: ', guessed/(analysed+not_analysed))

    print('guessed_pos_tags: ', g_pos_tags)
    print('pos_tags accuracy: ', g_pos_tags/(analysed+not_analysed))

    print('guessed_tags: ', g_tags)
    print('tags accuracy: ', g_tags/(analysed+not_analysed))
```

Fox

In [4]:

```
with open('unrecog-fox.txt', 'r') as file:
    file = file.read()
    fox_unrecog = re.findall('\d+ \^(?P<word>[а-яёI]+)\/\*[а-яёI]+\$', file)
```

In [5]:

```
df = pd.read_csv('fox-parses.csv')
```

In [6]:

```
st_unr = [(word, parse) for word, parse in zip(list(df.word), list(df.standard_pa
mod_unr = [(word, parse) for word, parse in zip(list(df.word), list(df.modified_p
st_noun = [(word, parse) for word, parse in zip(list(df.word), list(df.standard_p
mod_noun = [(word, parse) for word, parse in zip(list(df.word), list(df.modified_
st_verb = [(word, parse) for word, parse in zip(list(df.word), list(df.standard_p
mod_verb = [(word, parse) for word, parse in zip(list(df.word), list(df.modified_
```

In [20]:

```
print('standard_unrecog')
count_guessed(st_unr)
print()
print('standard_unrecog_verbs')
count_guessed(st_verb)
print()
print('standard_unrecog_nouns')
count_guessed(st_noun)
print()
```

```
standard_unrecog
analysed:  76
not analysed:  9
coverage:  0.8941176470588236
guessed:  37
accuracy:  0.43529411764705883
guessed_pos_tags:  46
pos_tags accuracy:  0.5411764705882353
guessed_tags:  48
tags accuracy:  0.5647058823529412

standard_unrecog_verbs
analysed:  27
not analysed:  0
coverage:  1.0
guessed:  16
accuracy:  0.5925925925925926
guessed_pos_tags:  18
pos_tags accuracy:  0.6666666666666666
guessed_tags:  19
tags accuracy:  0.7037037037037037

standard_unrecog_nouns
analysed:  35
not analysed:  8
coverage:  0.813953488372093
guessed:  21
accuracy:  0.4883720930232558
guessed_pos_tags:  28
pos_tags accuracy:  0.6511627906976745
guessed_tags:  28
tags accuracy:  0.6511627906976745
```

In [22]:

```
print('modified_unrecog')
count_guessed(mod_unr)
print()
print('mod_verbs')
count_guessed(mod_verb)
print()
```

```
modified_unrecog
analysed:  76
not analysed:  9
coverage:  0.8941176470588236
guessed:  40
accuracy:  0.47058823529411764
guessed_pos_tags:  50
pos_tags accuracy:  0.5882352941176471
guessed_tags:  51
tags accuracy:  0.6

mod_verbs
analysed:  27
not analysed:  0
coverage:  1.0
guessed:  19
accuracy:  0.7037037037037037
guessed_pos_tags:  22
pos_tags accuracy:  0.8148148148148148
guessed_tags:  22
tags accuracy:  0.8148148148148148
```

Perfect tests

In [11]:

```
with open('test_nouns.txt', 'r') as file:
    file = file.read().split()
    nouns = [(l.split(':')[1], l.split(':')[0]) for l in file]
```

In [23]:

```
count_guessed(nouns)
```

```
analysed:  225
not analysed:  0
coverage:  1.0
guessed:  225
accuracy:  1.0
guessed_pos_tags:  225
pos_tags accuracy:  1.0
guessed_tags:  225
tags accuracy:  1.0
```

In [13]:

```
with open('test_verbs.txt', 'r') as file:
    file = file.read().split()
    verbs = [(l.split(':')[1], l.split(':')[0]) for l in file]
```

In [14]:

```
count_guessed(verbs)
```

```
analysed:  833
not analysed:  0
coverage:  1.0
guessed:  581
accuracy:  0.6974789915966386
guessed_pos_tags:  716
pos_tags accuracy:  0.8595438175270108
guessed_tags:  716
tags accuracy:  0.8595438175270108
```

In [17]:

```
count_guessed(random.sample(verbs, 50), True)
```

```
FAIL
standard: авжарадогужа:авжиду<verb><progr><cvb.prs>
['авжарадогужа:авжду<verb><progr><cvb.prs>', 'авжарадогужа:авжаду<
verb><progr><cvb.prs>', 'авжарадогужа:авжану<verb><progr><cvb.prs
>', 'авжарадогужа:авжарадогIа<n><obl.pl><ess><prt>', 'авжарадогуж
а:авжарадогл<n><obl.pl><ess><prt>', 'авжарадогужа:авжарадога<n><ob
l.sg><ess><prt>', 'авжарадогужа:авжарадогIцIе<n><obl.sg><ess><prt
>', 'авжарадогужа:авжарадогл<n><obl.sg><ess><prt>', 'авжарадогужа:
авжарадогу<n><obl.sg><ess><prt>', 'авжарадогужа:авжарадогIа<n><ob
l.pl><in><prt>', 'авжарадогужа:авжарадогл<n><obl.pl><in><prt>', 'а
вжарадогужа:авжарадога<n><obl.sg><in><prt>', 'авжарадогужа:авжарад
огIцIе<n><obl.sg><in><prt>', 'авжарадогужа:авжарадогл<n><obl.sg><i
n><prt>', 'авжарадогужа:авжарадогу<n><obl.sg><in><prt>', 'авжарадо
гужа:авжарадогу<verb><inf><prt>', 'авжарадогужа:авжарадогу<n><abs.
sg><prt>', 'авжарадогужа:авжарадогуду<verb><aor><prt>', 'авжарадог
ужа:авжарадогуду<verb><imp><prt>', 'авжарадогужа:авжарадогуну<verb
><imp><prt>', 'авжарадогужа:авжарадогуну<verb><aor><prt>']
```