Two years of transducers at HSE

I dedicate this talk to Nick Howell

George Moroz
Linguistic Convergence Laboratory, HSE University, Moscow

18 October 2022

Outline of the talk

Tranducers for morphological parsing at HSE

What is transducer?

How to use transducers?

Zilo: from 0 to 40866 wordforms

Tranducers for morphological parsing at HSE

- Nick Howell gave a talk at School of Linguistics' seminar
- In 2020 we started four projects on:
  - Abaza (Daria Arakelova)
  - Agul (Roman Klimov)
  - Bagvalal (Daniil Ignatiev)
  - Botlikh (Artyom Sinelshikov)

Tranducers for morphological parsing at HSE

- Nick Howell gave a talk at School of Linguistics' seminar
- In 2020 we started four projects on:
  - Abaza (Daria Arakelova)
  - Agul (Roman Klimov)
  - Bagvalal (Daniil Ignatiev)
  - Botlikh (Artyom Sinelshikov)
- In summer of 2020 Danya continued his work in Google Summer of Code

Tranducers for morphological parsing at HSE

- Nick Howell gave a talk at School of Linguistics' seminar
- In 2020 we started four projects on:
  - Abaza (Daria Arakelova)
  - Agul (Roman Klimov)
  - Bagvalal (Daniil Ignatiev)
  - Botlikh (Artyom Sinelshikov)
- In summer of 2020 Danya continued his work in Google Summer of Code
- In 2021 I started five projects on my own (here some lectures):
  - Chamalal (Zina Budilova)
  - Botlikh (Igor Philatov)
  - Andi (Lera Buntyakova)
  - Agul (Nastya Burakova)
  - Rutul (Hanna Cupery)
  - continuation of Bagvalal (Daniil Ignatiev)
  - Even (Tanya Kazakova under supervision of Oleg Serikov)

Tranducers for morphological parsing at HSE

- Nick Howell gave a talk at School of Linguistics' seminar
- In 2020 we started four projects on:
  - Abaza (Daria Arakelova)
  - Agul (Roman Klimov)
  - Bagvalal (Daniil Ignatiev)
  - Botlikh (Artyom Sinelshikov)
- In summer of 2020 Danya continued his work in Google Summer of Code
- In 2021 I started five projects on my own (here some lectures):
  - Chamalal (Zina Budilova)
  - Botlikh (Igor Philatov)
  - Andi (Lera Buntyakova)
  - Agul (Nastya Burakova)
  - Rutul (Hanna Cupery)
  - continuation of Bagvalal (Daniil Ignatiev)
  - Even (Tanya Kazakova under supervision of Oleg Serikov)
- Result: (Arakelova and Ignatiev 2021; Budilova 2022; Buntyakova 2022; Burakova 2022; Kazakova 2022; Cupery and Philatov 2022)

Outline of the talk

Tranducers for morphological parsing at HSE

What is transducer?

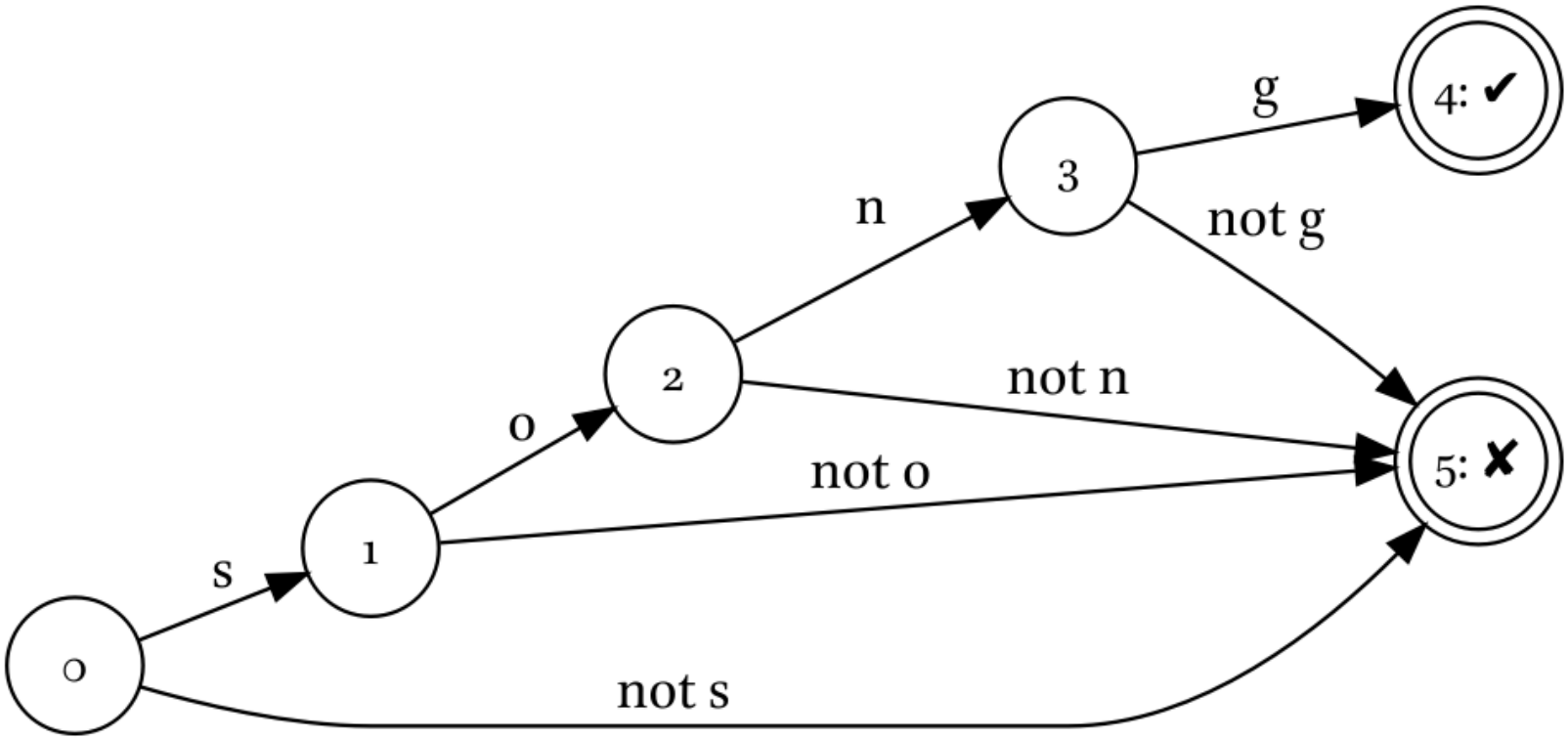
How to use transducers?

Zilo: from 0 to 40866 wordforms

What is transducer?

Trasducers are finite-state networks with two memory tapes of memory that can perform morphological analysis and related tasks. However, it easier to understand via the examples.

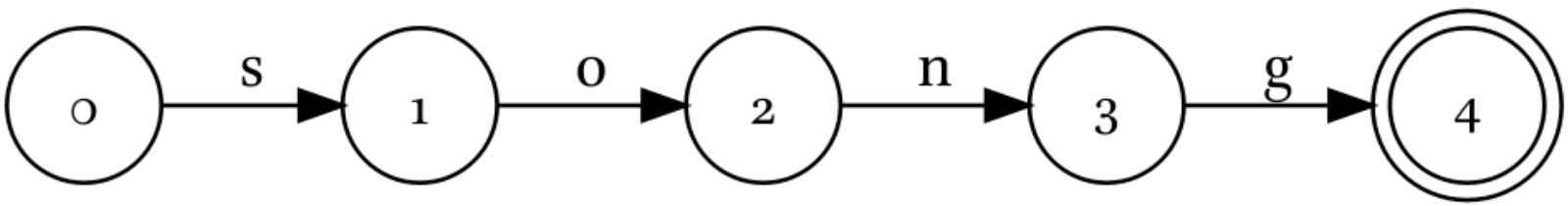
- Here is an example of finite-state automaton with one memory tape. It checks whether input string is a word *song*:



What is transducer?

Trasducers are finite-state networks with two memory tapes of memory that can perform morphological analysis and related tasks. However, it easier to understand via the examples.

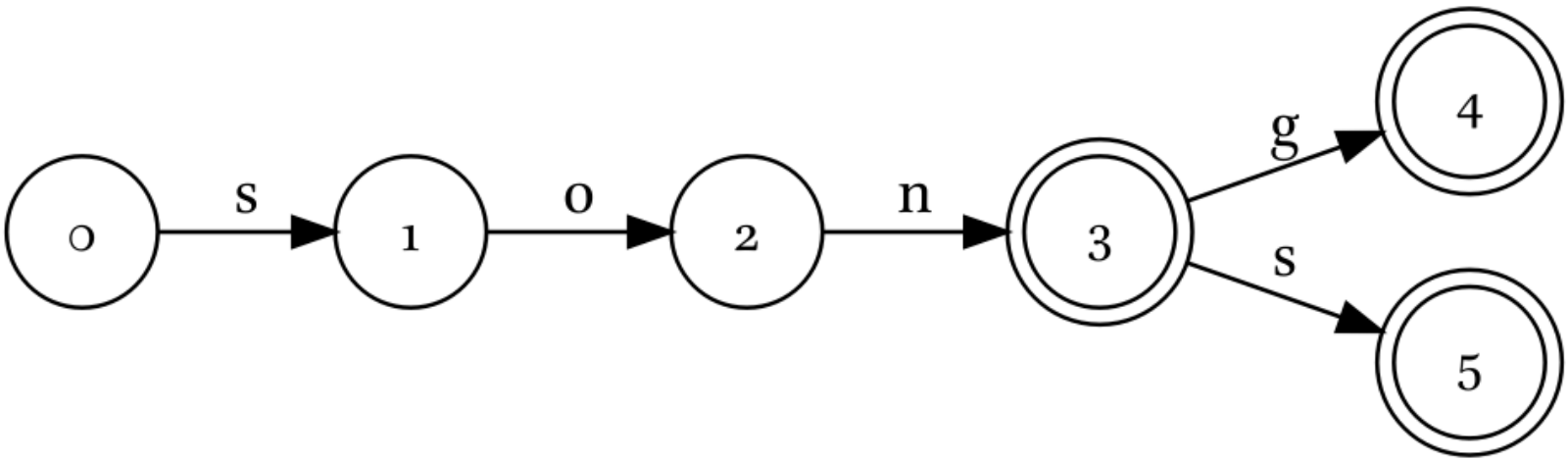
- Here is an example of finite-state automaton with one memory tape. It checks whether input string is a word *song*.
- Usually these “not X” arrow are not written:



What is transducer?

Trasducers are finite-state networks with two memory tapes of memory that can perform morphological analysis and related tasks. However, it easier to understand via the examples.

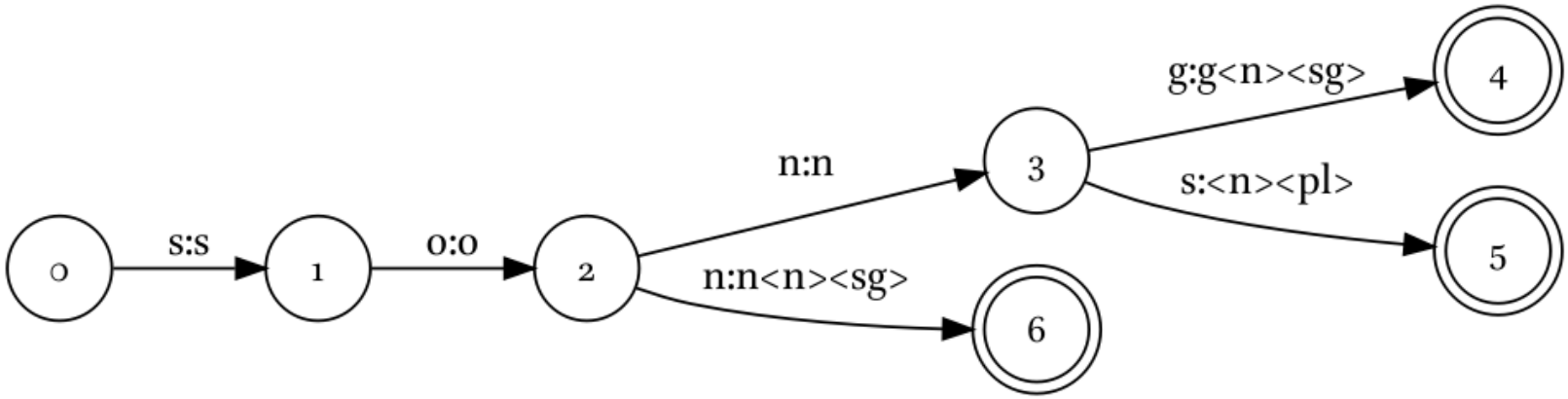
- Here is an example of finite-state automaton with one memory tape. It checks whether input string is a word *song*.
- Usually this “not X” arrow is not written
- It is possible to code multiple words (*son*, *sons*, *song*):



What is transducer?

Transducers has two memory tapes that can be treated as overwriting:

- `son` becomes `son<n><sg>`
- `song` becomes `song<n><sg>`
- `sons` becomes `son<n><pl>`
- everything else returns an error



Why use transducers?

- they are reversible, so analysis (`sons` → `son<n><pl>`) and generation (`son<n><pl>` → `sons`) can be done with the same transducer

Why use transducers?

- they are reversible, so analysis (`sons` → `son<n><pl>`) and generation (`son<n><pl>` → `sons`) can be done with the same transducer
- they can be optimized for the fast search

Why use transducers?

- they are reversible, so analysis (`sons` → `son<n><pl>`) and generation (`son<n><pl>` → `sons`) can be done with the same transducer
- they can be optimized for the fast search
- they can be easily combined with other transducers (e. g. transliteration or even translation)

Outline of the talk

Tranducers for morphological parsing at HSE

What is transducer?

How to use transducers?

Zilo: from 0 to 40866 wordforms

How to use transducers?

- read (Beesley and Karttunen 2003; Karttunen and Beesley 1992)
- `lexd` — a finite-state lexicon compiler (Swanson and Howell 2021)
- `twol` — a tool for (mor)phonology

`lexd` example (Zilo Andi numerals)

```
PATTERNS
Numerals NumearalMarker

LEXICON Numerals
иʰшду                  # пять; five
ойлIи                  # шесть; six
гьокьу                 # семь; seven
бейкьи                 # восемь; eight
гьочIо                 # девять; nine

LEXICON NumearalMarker
<num>:гу
```

lexd example (Zilo Andi numerals)

```
PATTERNS
Numerals NumearalMarker

LEXICON Numerals
иʰшду                  # пять; five
ойлIи                   # шесть; six
гьокьу                  # семь; seven
бейкьи                  # восемь; eight
гьочIо                  # девять; nine

LEXICON NumearalMarker
<num>:гу

иʰшдугу:иʰшду<num>
ойлIигу:ойлIи<num>
гьокьугу:гьокьу<num>
бейкьигу:бейкьи<num>
гьочIогу:гьочIо<num>
```

Pipeline for developing morphological transducer

- describe morphology and (mor)phonology using available sources

Pipeline for developing morphological transducer

- describe morphology and (mor)phonology using available sources
- compile lexicon with inflectional type annotation

Pipeline for developing morphological transducer

- describe morphology and (mor)phonology using available sources
- compile lexicon with inflectional type annotation
- compile test forms and their analysis based on available data (optional)

Pipeline for developing morphological transducer

- describe morphology and (mor)phonology using available sources
- compile lexicon with inflectional type annotation
- compile test forms and their analysis based on available data (optional)
  - `чIe<NUM><num><obl.m><epent.m><an.sg><aff>` `чIегушубо` (Zilo Andi)
- test your transducer against some annotated (or not annotated) corpus (optional)

Main problems during morphological transducer development

- not usual development environment (at least for our students/not computer linguists)
  - but Tanya Kazakova make it into Google Colab

Main problems during morphological transducer development

- not usual development environment (at least for our students/not computer linguists)
  - but Tanya Kazakova make it into Google Colab
- time

Main problems during morphological transducer development

- not usual development environment (at least for our students/not computer linguists)
  - but Tanya Kazakova make it into Google Colab
- time
- lack of resources

Main problems during morphological transducer development

- not usual development environment (at least for our students/not computer linguists)
  - but Tanya Kazakova make it into Google Colab
- time
- lack of resources
- traditions of Apertium people
  - there is a straightforward, but not the shortest way
    - from `чIе<NUM><num><obl.m><epent.m><an.sg><aff> чIерушубо` (Zilo Andi)
    - to two-num-obl.m-<an.sg>aff

Main problems during morphological transducer development

- not usual development environment (at least for our students/not computer linguists)
  - but Tanya Kazakova make it into Google Colab
- time
- lack of resources
- traditions of Apertium people
  - there is a straightforward, but not the shortest way
    - from `чIе<NUM><num><obl.m><epent.m><an.sg><aff> чIерушубо` (Zilo Andi)
    - to two-num-obl.m-<an.sg>aff
- difference in linguistic descriptions

Main problems during morphological transducer development

- not usual development environment (at least for our students/not computer linguists)
  - but Tanya Kazakova make it into Google Colab
- time
- lack of resources
- traditions of Apertium people
  - there is a straightforward, but not the shortest way
    - from `чIе<NUM><num><obl.m><epent.m><an.sg><aff> чIерушубо` (Zilo Andi)
    - to two-num-obl.m-<an.sg>aff
- difference in linguistic descriptions
- difference in languages

Outline of the talk

Tranducers for morphological parsing at HSE

What is transducer?

How to use transducers?

Zilo: from 0 to 40866 wordforms

Zilo: from 0 to 40866 wordforms

This September I have been in Zilo (Andi) for about 5 days:

- collected a dictionary with inflectional type annotation (more than 700 words)
- created a transducer that analyze/generate 40866 wordforms
  - about 400 nouns, 70 adjectives, 50 adverbs, numerals
  - just nominal and adjective inflection
  - future work: pronouns, verb inflection

References

D. Arakelova and D. Ignatiev. Developing morphological analyzers for low-resource languages of the Caucasus. NRU HSE, 2021.

K. R. Beesley and L. Karttunen. *Finite-state morphology: Xerox tools and techniques*. CSLI, Stanford, 2003.

Z. A. Budilova. Morphological parser of Chamalal in lexd and twol. NRU HSE, 2022.

V. A. Buntyakova. Morphological parser of Andi in lexd and twol. NRU HSE, 2022.

A. V. Burakova. Morphological parser of Agul in lexd and twol. NRU HSE, 2022.

H. Cupery and I. Philatov. Developing morphological analyzers for low-resource Nakh-Dagestani languages. NRU HSE, 2022.

References

L. Karttunen and K. R. Beesley. *Two-level rule compiler*. Xerox Corporation, Palo Alto Research Center, 1992.

T. B. Kazakova. Building a corpus of the Bystraja Even language using automatic morphological annotation. NRU HSE, 2022.

D. Swanson and N. Howell. Lexd: a Finite-State lexicon compiler for non-suffixational morphologies. In M. Hämäläinen, N. Partanen, and K. Alnajjar, editors, *Multilingual facilitation*. 2021.