

Зачем нужен правиловый морфологический анализ в XXI веке?

Язык, текст, культура через призму цифровых технологий

НИУ ВШЭ, Санкт-Петербург

04–05 декабря 2025

Г. А. Мороз НИУ ВШЭ, Москва

Международная лаборатория языковой конвергенции

5.12.2025



Зачем нужен морфологический анализ?



Что такое морфологический анализ

Морфологический анализ, как его обычно видят лингвисты, обычно включает в себя несколько вещей:

- определение морфологической формы (например, латинское *mensam* — acc.sg);
 - приведение к начальной форме или основе (*mensam* — *mensa*, ж. р., первое склонение);
 - перевод основы (*mensam* — ‘стол’).



Что такое морфологический анализ

Морфологический анализ, как его обычно видят лингвисты, обычно включает в себя несколько вещей:

- определение морфологической формы (например, латинское *mensam* — acc.sg);
 - приведение к начальной форме или основе (*mensam* — *mensa*, ж. р., первое склонение);
 - перевод основы (*mensam* — ‘стол’).

Не бывает правильного морфологического анализа — это каждый раз компромисс между языковой моделью, конвенциями исследователей и текущими задачами. Например, лингвисты, приводя примеры, редко перечисляют несловоизменительную информацию (например, род для существительных).



Зачем нужен морфологический анализ?

- Польза от морфологически размеченных текстов для теоретической лингвистики неоценима

Зачем нужен морфологический анализ?

- Польза от морфологически размеченных текстов для теоретической лингвистики неоценима
 - В области NLP — все не так однозначно:
 - Какие-то задачи можно решить при помощи *стэминга*
 - Выгода зависит от языка к языку
 - Легко представить, что в некоторых задачах выгоднее иметь лишь какую-то часть морфологического анализа, например, частеречную разметку

Подходы к морфологическому анализу

Много данных

Обычно, если много данных, люди используют нейросети. Для морфологического анализа русского языка их использовали в следующих работах [Arefyev et al., 2018, Sorokin and Kravtsova, 2018, Bolshakova and Sapin, 2019a,b, 2020, Garipov et al., 2023]. Используются разные архитектуры:

- свёрточные нейронные сети (convolutional neural network, CNN);
 - деревья решения с градиентным бустингом (decision trees with gradient boosting);
 - двунаправленная длинная цепь элементов краткосрочной памяти (Bidirectional long short-term memory network, Bi-LSTM);

Много данных

Обычно, если много данных, люди используют нейросети. Для морфологического анализа русского языка их использовали в следующих работах [[Arefyev et al., 2018](#), [Sorokin and Kravtsova, 2018](#), [Bolshakova and Sapin, 2019a,b, 2020](#), [Garipov et al., 2023](#)]. Используются разные архитектуры:

- свёрточные нейронные сеть (convolutional neural network, CNN);
- деревья решения с градиентным бустингом (decision trees with gradient boosting);
- двунаправленная длинная цепь элементов краткосрочной памяти (Bidirectional long short-term memory network, Bi-LSTM);
- [UDPipe](#) — проект, основанный на размеченных в формате [Universal Dependencies](#) трибанках большого количества языков [[Straka, 2018](#)]. Внутри: LSTM, которая работает на основе векторного представления слов.
- [Morfessor](#), в котором используют скрытые марковские цепи [[Grönroos et al., 2014](#)].



Мало данных

В случаях, когда ресурсов мало, используют правиловые методы

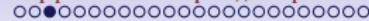
- проект uniparser-morph Тимофея Архангельского [[Архангельский, 2012](#)];
- нечто, что работает в SIL Fieldworks;
- множество узконаправленных парсеров, написанных для конкретных языков;
- морфологические анализаторы на основе трансдьюсеров
 - Helsinki Finite-State Tookit hfst [[Lindén et al., 2011](#)];
 - аналогичным инструментом от сообщества Apertium lttoolbox [[Ortiz Rojas et al., 2005](#)].

Морфологические трансдьюсеры

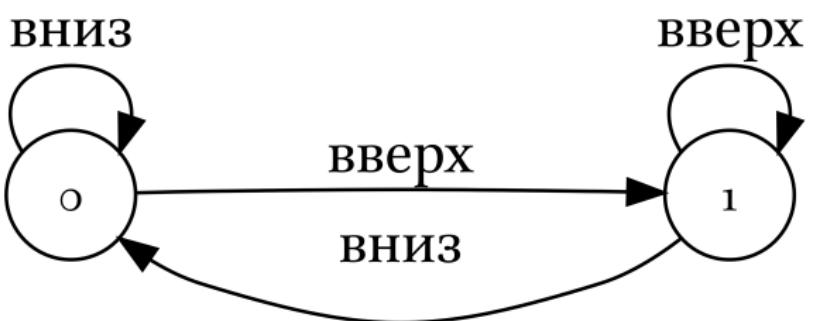
Конечные автоматы

Теория автоматов — это дисциплина на стыке математики и компьютерных наук, которая появилась в XX веке. Первые конечных автоматов были предложены в работах [Mealy, 1955, Moore, 1956]. Данный раздел основан на первой главе из [Beesley and Karttunen, 2003а, 1–42].

Под автоматами мы понимаем абстрактные машины, которые принимают разные состояния, а изменение состояний вызывается некоторым действием.



Конечные автоматы

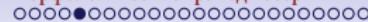


Конечные автоматы

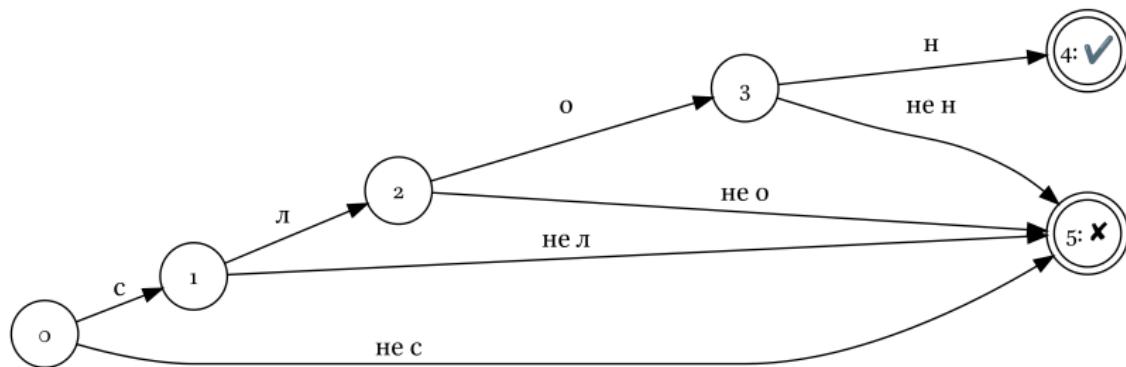
Конечных автоматы состоят:

- алфавит, который автомат понимает;
 - конечное количество состояний;
 - переходы между состояниями;
 - одно начальное состояние (часто обозначают нулем);
 - набор конечных состояний (часто обозначают двойным кружочком).

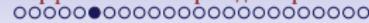
Конечные автоматы можно использовать для побуквенной верификации поданных на вход слов.



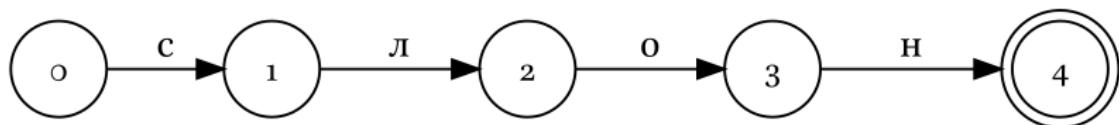
Конечные автоматы



Если программа смогла пройти путь до конечного состояния (обозначен двойным кружочком), значит операция завершилась успехом, в остальных случаях — неудачей.

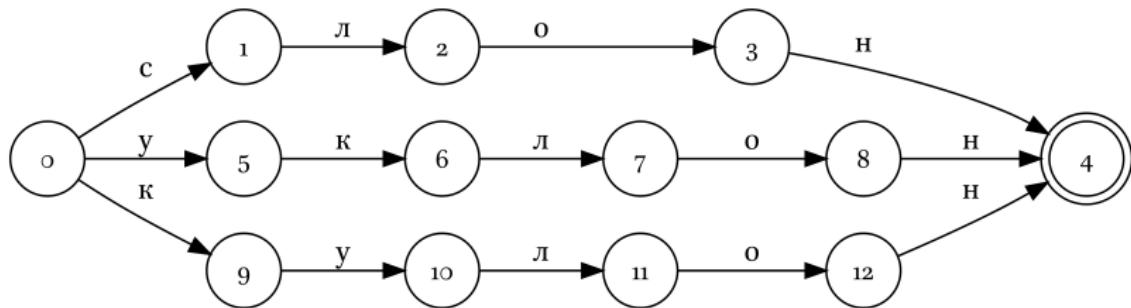


Конечные автоматы

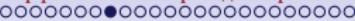


Обычно путь, ведущий к неудаче, не отображают.

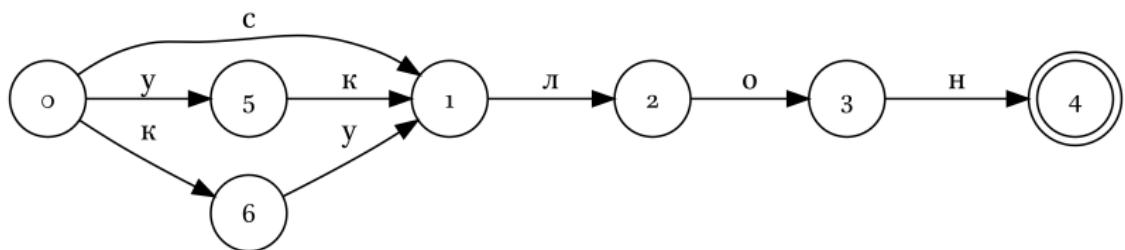
Конечные автоматы



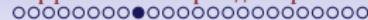
Можно сделать так, чтобы автомат верифицировал несколько слов.



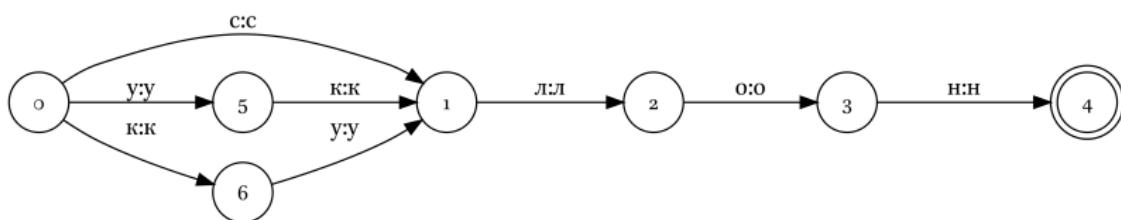
Конечные автоматы



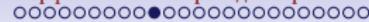
Полученный автомат можно оптимизировать, так, чтобы там было меньше узлов, а задачи он решал те же самые.



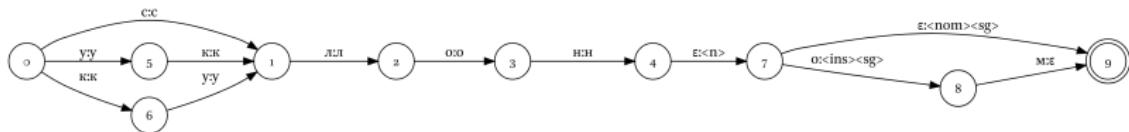
Трансдьюсеры



Если мы немножко усложним автомат, добавив в него еще выходной алфавит, то мы получим трансдьюсер (в [русской википедии](#) они названы конечными автоматами с выходом). Мы будем использовать обновленную нотацию: то, что представлено на вход, мы пишем слева от двоеточия, а то, что получается на выходе — справа.



Трансдьюсеры



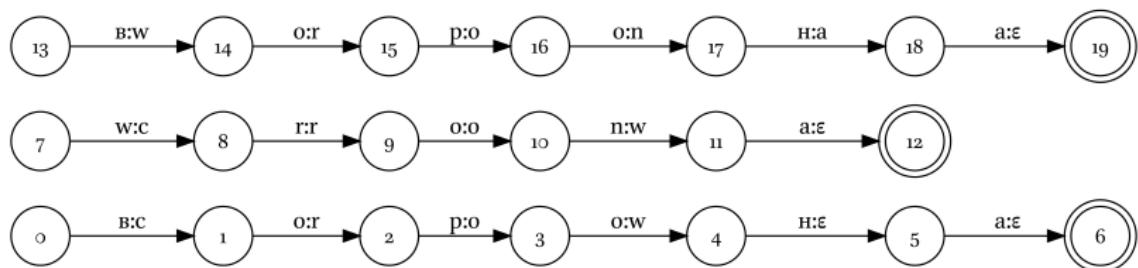
Введя обозначение пустой строки, которую принято обозначать греческой буквой эпсилон ϵ , мы можем представить трансдьюсер, который, наконец-то, делает некоторый морфологический анализ.

Операции с трансдьюсерами

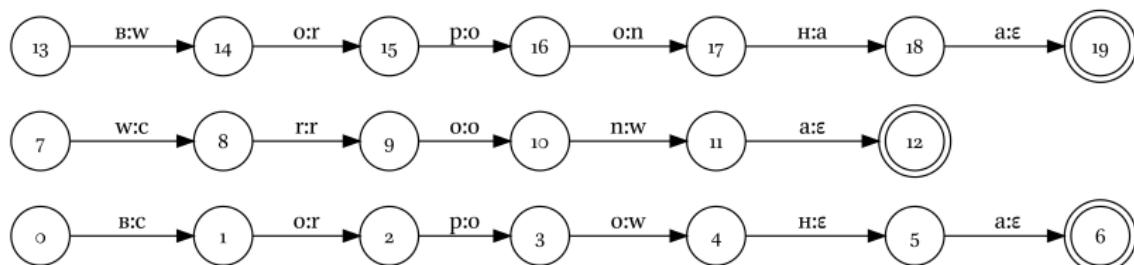
С трансдьюсерами можно делать много разных операций:

- инвертирование;
 - объединение (конечные и начальные состояния совпадают, все промежуточные сохраняются);
 - конкатенация (конечное состояние одного трансдьюсера становится начальным состоянием другого);
 - пересечение (в особых случаях);
 - вычитание (в особых случаях);
 - композиция.

Композиция трансдьюсеров



Композиция трансдьюсеров



Вообще перевод близкородственных языков при помощи трансдьюсеров делают, но алгоритм там конечно умнее:

- запускаем анализатор языка A;
- составляем правила соответствий переходов от лемм и морфологических форм языка A в язык B;
- дальше запускаем генератор языка B;
- запускаем правила постобработки.

Так как трансдьюсеры обратимы — данная последовательность действий будет работать в обе стороны.

Морфология lexd

PATTERNS

Noun NounInflection

Noun Suffix AdjInflection

LEXICON Noun

ночь

печь

LEXICON Suffix

<adj>:н

LEXICON NounInflection

<nom><sg>:

<ins><pl>:ами

LEXICON AdjInflection

<m><nom><sg>:ой

<f><nom><sg>:ая

Морфология lexd

ночь<nom><sg>:ночь

печь<nom><sg>:печь

ночь<ins><pl>:ночьями

печь<ins><pl>:печьами

ночь<adj><m><nom><sg>:ночной

печь<adj><m><nom><sg>:печкой

ночь<adj><f><nom><sg>:ночья

печь<adj><f><nom><sg>:печная

Двухуровневая фонология/морфология

Двухуровневая фонология/морфология (two level morphology) была разработана в диссертации [[Koskenniemi, 1983](#)]. Еще в 1972 вышла диссертация [[Johnson, 1972](#)], в которой автор указывал на некоторые недостатки последовательности фонологических правил, которые были приняты в генеративной фонологии, а также доказывал, что любую последовательность правил можно моделировать при помощи трансдьюсера.

Двухуровневая фонология/морфология

В рамках двухуровневой фонологии/морфологии:

- правила — посимвольные ограничения на поверхностное представление, которые применяются параллельно.
- правила могут оперировать единицами глубинного представления, поверхностного представления или одновременно обоих.

Например, получить из глубинной формы `spy>s` поверхностную форму `spies` можно двумя правилами:

- `y:i < = > _ θ:e`
- `θ:e < = > y: _ %>:θ`

(Mor)фонология two1

Alphabet

а е я й к н о п ч ъ:θ;

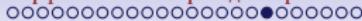
Rules

"чк чн пишется без ъ"

! например, ночьной → ночной или печька → печка

ъ:θ < = > _ н;

 _ к;



(Mor)фонология two1

ночь<nom><sg>:ночь

печь<nom><sg>:печь

ночь<ins><pl>:ночьями

печь<ins><pl>:печьами

ночь<adj><m><nom><sg>:ночной

печь<adj><m><nom><sg>:печной

ночь<adj><f><nom><sg>:ночная

печь<adj><f><nom><sg>:печная

Разрешение морфологической неоднозначности сg3

Парадигма Constraint grammar (CG) [Karlsson et al., 1995, Bick and Didriksen, 2015] — это правиловая процедурная система обработки текста, позволяющая решать достаточно большой набор разнообразных задач, такие как

- разрешение неоднозначности;
- приписывание тэгов, например, синтаксических ролей;
- разрешение анафоры;
- построение деревьев зависимостей;
- chunking — выделение границ синтаксических единиц (без внутренней структуры, отношений вершины-зависимое и т. п.) [Bick, 2013];
- и многие другие.



Что необходимо для создания морфологического анализатора?

- грамматическое описание;
- словарь;
- корпус текстов
 - без морфологической разметки;
 - с морфологической разметкой.

Проблемы моделирования морфологии языка

- Проблема описаний:
 - неполнота: то, что исследователи посчитали достаточным для грамматического описания, может быть недостаточно для моделирования; для некоторого языка может быть доступно грамматическое описание, но отсутствовать словарь и наоборот; словарь может не содержать информации про словоизменительный класс каких-то единиц, которые различаются в грамматическом описании.
 - противоречивые источники: грамматические описания могут противоречить друг другу; грамматические описания могут противоречить словарям
- Проблема вариативности:
 - идиолектная
 - диалектная
 - связанная с какими-нибудь социолингвистическими параметрами (в первую очередь такие как пол и возраст, но можно придумать и другие)

Проблемы моделирования морфологии языка

- Проблема неоднозначности (особенно перекошенной частотно):
 - лексической
 - морфологическая
 - синтаксическая
 - ...
- Проблема идиом (*выйти за муж*) и суплетивизма
- Проблема циклов, возникающих при словообразовании:
 - слуга (N) > служить (V) > услужить (V) > услужение (N), услуга (N)
- Проблема морфологической сложности: для языков с бедной морфологией проще строить простой правиловый морфологический парсер, а не трансдьюсер

Что читать?

- Онлайн материалы нашего с Т. Казаковой курса в НИУ ВШЭ
 - документацию lexd
 - введение в twol в [Beesley and Karttunen, 2003b]
 - документацию cg3
 - ACL Anthology:
 - morphological transducers — около 3440 результатов
 - transducers — около 6450 результатов

Спасибо за внимание!

Ссылки на литературу I

N. V. Arefyev, T. Y. Gratsianova, and K. P. Popov. Morphological segmentation with sequence to sequence neural network. In *Компьютерная лингвистика и интеллектуальные технологии*, pages 85–95, 2018.

K. R. Beesley and L. Karttunen. *Finite State Morphology: Xerox tools and techniques*. Center for Study of Language Information, Stanford, 2003a.

K. R. Beesley and L. Karttunen. Two-level rule compiler, 2003b.

Eckhard Bick. Using constraint grammar for chunking. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*, pages 13–26, 2013.

Eckhard Bick and Tino Didriksen. Cg-3 — beyond classical constraint grammar. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 31–39, 2015.

Ссылки на литературу II

- E. I. Bolshakova and A. S. Sapin. Bi-lstm model for morpheme segmentation of russian words. In *Artificial Intelligence and Natural Language: 8th Conference, AINL 2019, Tartu, Estonia, November 20–22, 2019, Proceedings* 8, pages 151–160. Springer, 2019a.

E. I. Bolshakova and A. S. Sapin. Comparing models of morpheme analysis for russian words based on machine learning. In *Компьютерная лингвистика и интеллектуальные технологии*, pages 104–113, 2019b.

E. I. Bolshakova and A. S. Sapin. An experimental study of neural morpheme segmentation models for russian word forms. In *CMCL*, pages 79–89, 2020.

T. Garipov, D. Morozov, and A. Glazkova. Generalization ability of cnn-based morpheme segmentation. In *2023 Ivannikov Ispras Open Conference (ISPRAS)*, pages 58–62. IEEE, 2023.

Ссылки на литературу III

S.-A. Grönroos, S. Virpioja, P. Smit, and M. Kurimo. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1177–1185, 2014.

C. D. Johnson. *Formal aspects of phonological description*. Mouton, The Hague, Paris, 1972.

Fred Karlsson, Atro Voutilainen, Juha Heikkilae, and Arto Anttila.
Constraint Grammar: a language-independent system for parsing unrestricted text, volume 4. Walter de Gruyter, 1995.

K. Koskenniemi. *Two-level morphology: A general computational model for word-form recognition and production*. PhD thesis, University of Helsenki, Department of General Linguistics, 1983. Publications, 11.

Ссылки на литературу IV

- K. Lindén, E. Axelson, S. Hardwick, T. A. Pirinen, and M. Silfverberg. Hfst—framework for compiling and applying morphologies. In *Systems and Frameworks for Computational Morphology: Second International Workshop, SFCM 2011, Zurich, Switzerland, August 26, 2011. Proceedings* 2, pages 67–85. Springer, 2011.

G. H. Mealy. A method for synthesizing sequential circuits. *The Bell System Technical Journal*, 34(5):1045–1079, 1955.

E. F. Moore. Gedanken-experiments on sequential machines. *Automata studies*, 34:129–153, 1956.

S. Ortiz Rojas, M. L. Forcada, and G. Ramírez Sánchez. Construcción y minimización eficiente de transductores de letras a partir de diccionarios con paradigmas. *Procesamiento del lenguaje natural*, 35: 51–57, 2005.

Ссылки на литературу V

Alexey Sorokin and Anastasia Kravtsova. Deep convolutional networks for supervised morpheme segmentation of russian language. In *Artificial Intelligence and Natural Language: 7th International Conference, AINL 2018, St. Petersburg, Russia, October 17–19, 2018, Proceedings* 7, pages 3–10. Springer, 2018.

Milan Straka. Udpipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies*, pages 197–207, 2018.

Т. А. Архангельский. *Принципы построения морфологического парсера для разноструктурных языков*. PhD thesis, Московский государственный университет им. М. В. Ломоносова, 2012.