НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

Dependency grammar and dependency parsing

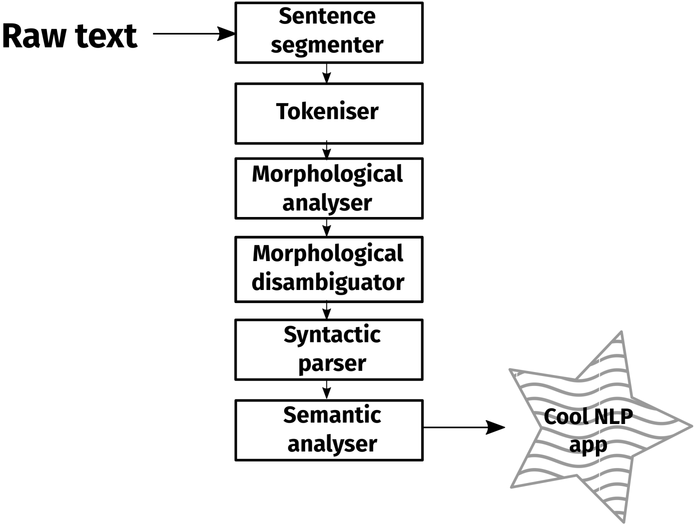
Francis M. Tyers

ftyers@hse.ru
https://www.hse.ru/org/persons/209454856

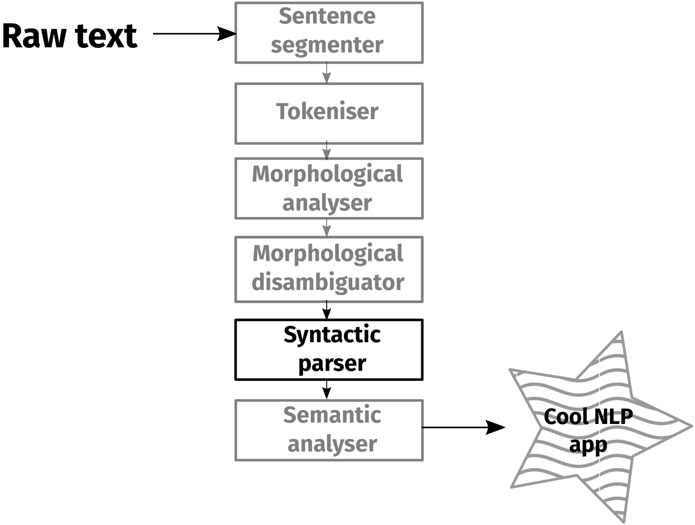
Национальный исследовательский университет
«Высшая школа экономики» (Москва)

26 марта 2018 г.

Pipeline


Raw text
Sentence segmenter
Tokeniser
Morphological analyser
Morphological disambiguator
Syntactic parser
Semantic analyser
Cool NLP app

Pipeline



Raw text

Sentence segmenter

Tokeniser

Morphological analyser

Morphological disambiguator

Syntactic parser

Semantic analyser

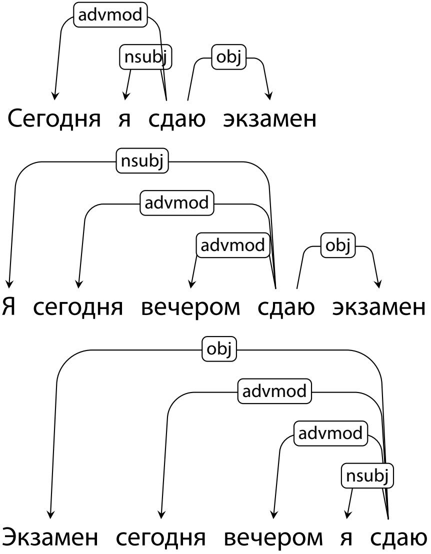
Cool NLP app

Motivating example

	Bigrams
Сегодня я сдаю экзамен	я сдаю, сдаю экзамен
Я сегодня вечером сдаю экзамен	вечером сдаю, сдаю экзамен
Экзамен сегодня вечером я сдаю	я сдаю, сдаю EOS

- Generalise over linear order
- Generalise long-distance

Motivating example



Bigrams

я сдаю, сдаю экзамен

Я сдаю, сдаю экзамен

я сдаю, сдаю экзамен

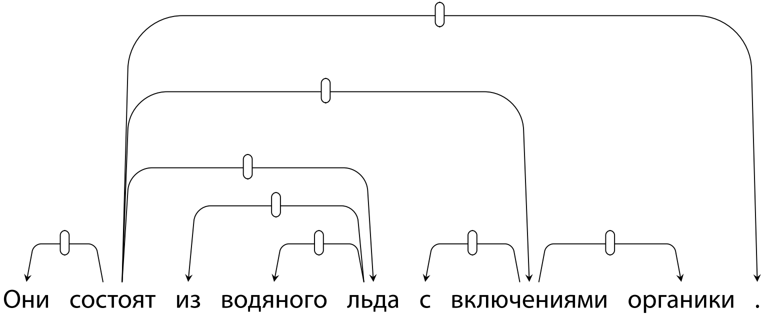
Dependency syntax

- Word based
- No non-terminals
- Words are linked by one-way binary relations
- Relations may be typed or untyped

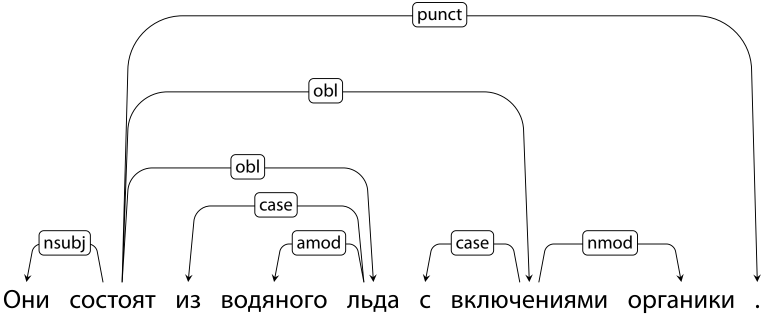
Dependency structure

Они состоят из водяного льда с включениями органики .

Dependency structure



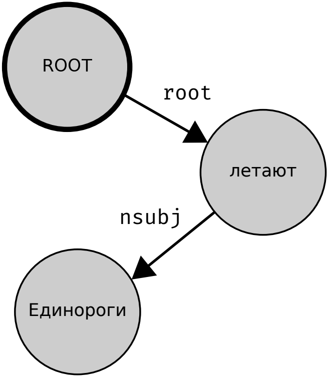
Dependency structure



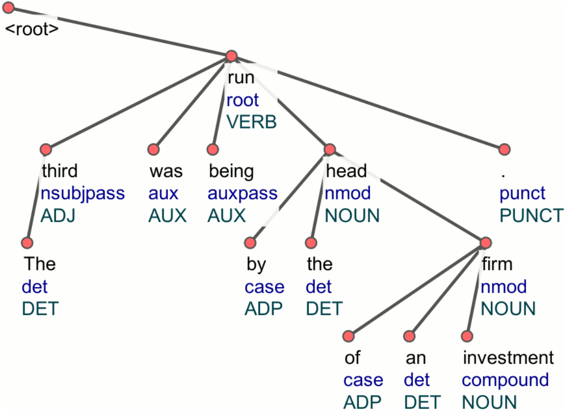
- Labels describe functional relations

Terminology

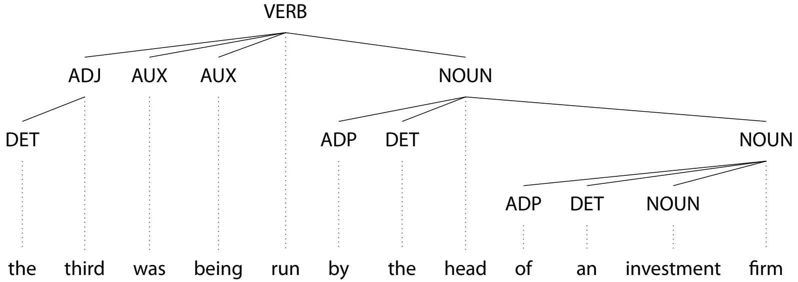
Superior	**Inferior**
Head	Dependent
Governor	Modifier
Regent	Subordinate
Mother	Daughter
Parent	Child



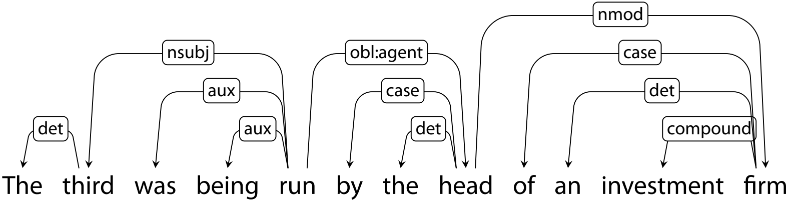
Notational variants



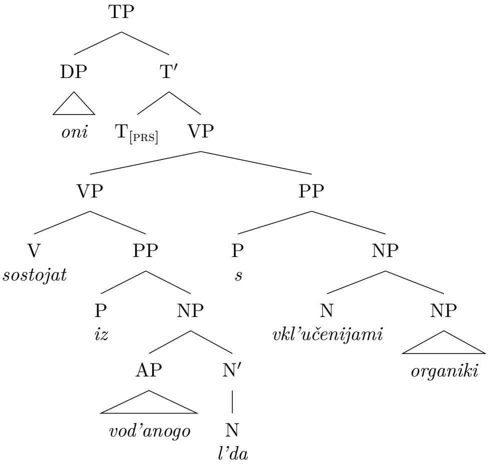
Notational variants



Notational variants



Phrase structure



Comparison

Dependency structures explicitly represent:

- head–dependent relations (directed arcs)
- functional categories (arc labels)

Phrase structures explicitly represent:

- phrases (non-terminal nodes)
- structural categories (non-terminal labels)

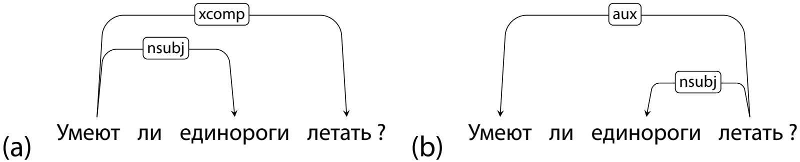
Heads and dependents

- Criteria for a syntactic relation between a head H and a dependent D in a construction C (Zwicky, 1985)[1]
  1. *H* determines the syntactic category of *C*; *H* can replace *C*
  2. *H* determines the semantic category of *C*; *D* specifies *H*
  3. *H* is obligatory, *D* may be optional
  4. *H* selects *D* and determines optionality of *D*
  5. The form of *D* depends on *H* (agreement or government)
  6. Linear position of *D* is specified with reference to *H*
- An issue:
  - Syntactic (and morphological) versus semantic criteria

[1] Zwicky, A. (1985) "Heads"*Journal of Linguistics*, 21:1–29

Some fuzzy cases

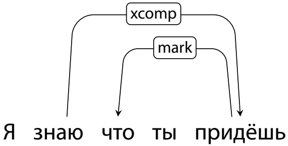
- **Complex verb groups** (auxiliary–main verb)
- Subordinate clauses (complementiser–verb)
- Coordination (coordinator–conjuncts)
- Adpositional phrases (adposition–nominal)
- Punctuation



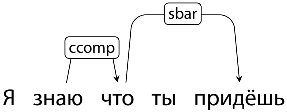
Some fuzzy cases

- Complex verb groups (auxiliary–main verb)
- **Subordinate clauses** (complementiser–verb)
- Coordination (coordinator–conjuncts)
- Adpositional phrases (adposition–nominal)
- Punctuation

(a)

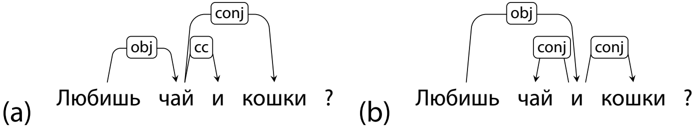


(b)



Some fuzzy cases

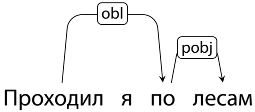
- Complex verb groups (auxiliary–main verb)
- Subordinate clauses (complementiser–verb)
- **Coordination** (coordinator–conjuncts)
- Adpositional phrases (adposition–nominal)
- Punctuation



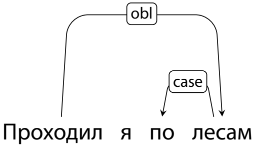
Some fuzzy cases

- Complex verb groups (auxiliary–main verb)
- Subordinate clauses (complementiser–verb)
- Coordination (coordinator–conjuncts)
- **Adpositional phrases** (adposition–nominal)
- Punctuation

(a)



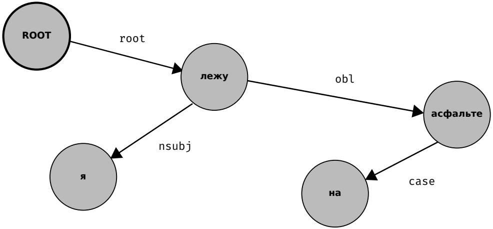
(b)



Some fuzzy cases

- Complex verb groups (auxiliary–main verb)
- Subordinate clauses (complementiser–verb)
- Coordination (coordinator–conjuncts)
- Adpositional phrases (adposition–nominal)
- **Punctuation**

Dependency graphs



A dependency graph, *G*

- a set of *V* nodes, $V = \{0, 1, 2, 3, 4\}$
- a set of *A* arcs, $A = \{0 \rightarrow 2, 2 \rightarrow 1, 2 \rightarrow 4, 4 \rightarrow 3\}$
- a linear precedence order $<$ on *V*

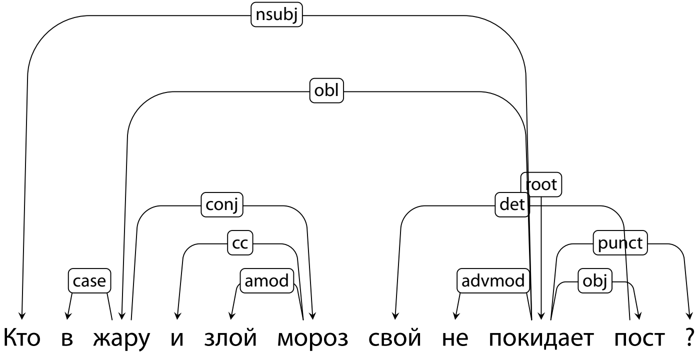
Labelled graphs:

- Nodes in *V* are labelled with word forms (and annotation)
- Arcs in *A* are labelled with dependency types

Conditions

- **Connectedness**: The syntactic structure must be complete, every word must be covered by the structure
- **Acyclicity**: The structure must be hierarchical, no cycles,
- **Single-headedness**: Each word must have at most one head

Projectivity



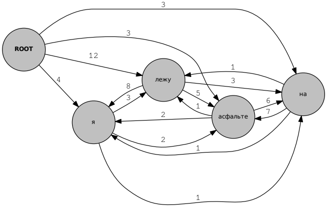
- Projectivity = no crossing arcs
- Non-projective → more complex to parse

Parsing methods

Transition-based

...

Graph-based



Transition-based

General idea

Components

Data structures:

- Stack:
  - Starts as containing only the ROOT
- Buffer
  - Starts as containing the full sentence
- Arcs
  - Starts as empty

Operations:

- SHIFT: Take the word on top of the buffer and put it on the stack
- LEFT-ARC: Make the word at the top of the stack the head of the word below it
  - Then remove the word at the top
- RIGHT-ARC: Make the word second from top the head of the word above it
  - Then remove the second from top word

Example

ROOT   Мы   пошли   домой

Stack	**Buffer**
ROOT	Мы пошли домой

Example

SHIFT

ROOT Мы пошли домой

Stack	**Buffer**
ROOT Мы	пошли домой

Example

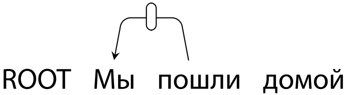
SHIFT

ROOT Мы пошли домой

Stack	**Buffer**
ROOT Мы пошли	домой

Example

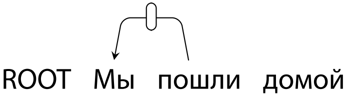
LEFT-ARC



Stack	**Buffer**
ROOT пошли	домой

Example

SHIFT

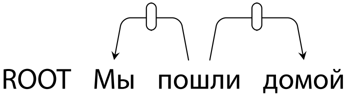


ROOT Мы пошли домой

Stack	Buffer
ROOT пошли домой	

Example

RIGHT-ARC

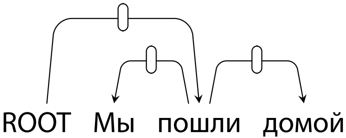


Stack **Buffer**

ROOT пошли

Example

RIGHT-ARC



Stack	**Buffer**
ROOT	

Configurations

A **configuration** is a snapshot of the state of the parser at a given time.

- A stack: Representing the word(s) currently being processed
- A buffer: Representing the remaining words
- A set of arcs representing a (partial) tree

We can conceive parsing as transitioning from one configuration to another via an operation.

Oracle

How do we get the sequence of operations?

Deterministic algorithm:

- LEFT-ARC: Configuration has arc from the top of stack to the word below
- RIGHT-ARC: Configuration has arc from the of the stack to the first word in the input buffer
  - In addition: The dependent must have no dependents of its own
- SHIFT: All other cases

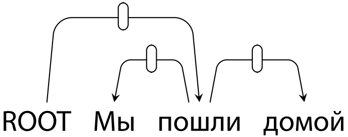
Oracle

How do we get the sequence of operations?

Deterministic algorithm:

- LEFT-ARC: Configuration has arc from the top of stack to the word below
- RIGHT-ARC: Configuration has arc from the of the stack to the **first word in the input buffer**
    - In addition: The **dependent** must have no dependents of its own
- SHIFT: All other cases

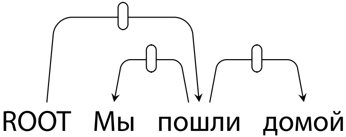
Parsing with an oracle



Stack	**Buffer**
ROOT	Мы пошли домой

- Is there an arc from the first word in the buffer to the top of the stack?
  - (Мы, ROOT)
- Is there an arc from the top of the stack to the first word in the buffer?
  - (ROOT, Мы)
- → then SHIFT

Parsing with an oracle

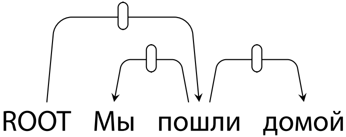


ROOT Мы пошли домой

Stack	Buffer
ROOT Мы	пошли домой

- Is there an arc from the first word in the buffer to the top of the stack?
    - (пошли, Мы) — YES, LEFT-ARC
- Is there an arc from the top of the stack to the first word in the buffer?
    - (Мы, пошли)

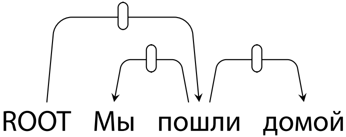
Parsing with an oracle



Stack	Buffer
ROOT	пошли домой

- Is there an arc from the first word in the buffer to the top of the stack?
  - (пошли, ROOT)
- Is there an arc from the top of the stack to the first word in the buffer?
  - (ROOT, пошли) – YES, but *пошли* still has dependents
- → then SHIFT

Parsing with an oracle



ROOT Мы пошли домой

Stack	Buffer
ROOT пошли	домой

- Is there an arc from the first word in the buffer to the top of the stack?
  - (домой, пошли)
- Is there an arc from the top of the stack to the first word in the buffer?
  - (пошли, домой) – YES, and *домой* has no dependents
  - → RIGHT-ARC

Training data

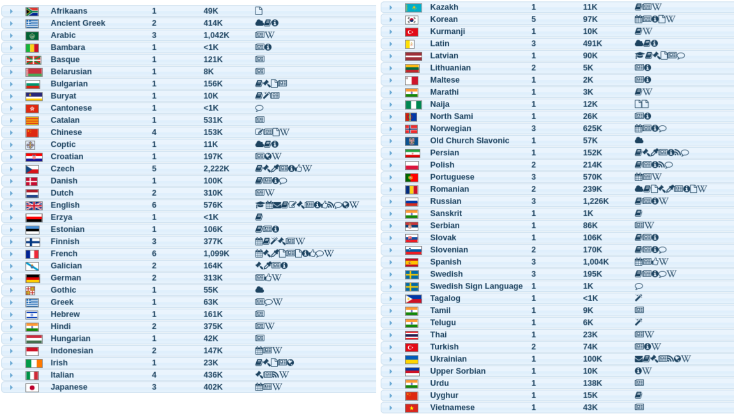
The “only” training data required is a treebank.

- Collection of sentences annotated for dependency structure
- Universal dependencies: 67 languages, 100s of treebanks

Data trains a classifier to predict a transition from a configuration.

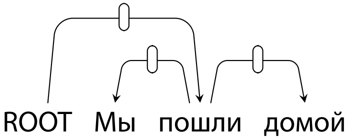
Training data

http://universaldependencies.org



Afrikaans	1	49K
Ancient Greek	2	414K
Arabic	3	1,042K
Bambara	1	<1K
Basque	1	121K
Belarusian	1	8K
Bulgarian	1	156K
Buryat	1	10K
Cantonese	1	<1K
Catalan	1	531K
Chinese	4	153K
Coptic	1	11K
Croatian	1	197K
Czech	5	2,222K
Danish	1	100K
Dutch	2	310K
English	6	576K
Erzya	1	<1K
Estonian	1	106K
Finnish	3	377K
French	6	1,099K
Galician	2	164K
German	2	313K
Gothic	1	55K
Greek	1	63K
Hebrew	1	161K
Hindi	2	375K
Hungarian	1	42K
Indonesian	2	147K
Irish	1	23K
Italian	4	436K
Japanese	3	402K
Kazakh	1	11K
Korean	5	97K
Kurmanji	1	10K
Latin	3	491K
Latvian	1	90K
Lithuanian	2	5K
Maltese	1	2K
Marathi	1	3K
Naija	1	12K
North Sami	1	26K
Norwegian	3	625K
Old Church Slavonic	1	57K
Persian	1	152K
Polish	2	214K
Portuguese	3	570K
Romanian	2	239K
Russian	3	1,226K
Sanskrit	1	1K
Serbian	1	86K
Slovak	1	106K
Slovenian	2	170K
Spanish	3	1,004K
Swedish	3	195K
Swedish Sign Language	1	1K
Tagalog	1	<1K
Tamil	1	9K
Telugu	1	6K
Thai	1	23K
Turkish	2	74K
Ukrainian	1	100K
Upper Sorbian	1	10K
Urdu	1	138K
Uyghur	1	15K
Vietnamese	1	43K

Features



Stack	Buffer
ROOT пошли	домой

Features indexed by address (in stack or buffer) and attribute name.

Traditional:

- (Stack[0], Form) = пошли
- (Buffer[0], Form) = домой
- (Stack[0], UPOS) = VERB
- (Stack[1], Form) = ROOT

Can return NULL.

Indicator:

- Combinations of such features, e.g.
  - (Stack[0], Form) = пошли & (Buffer[0], UPOS) = ADP

Features begone!

Instead of all of those features, what do people do nowadays?

- Use embeddings
    - For words, POS tags, characters, features etc.

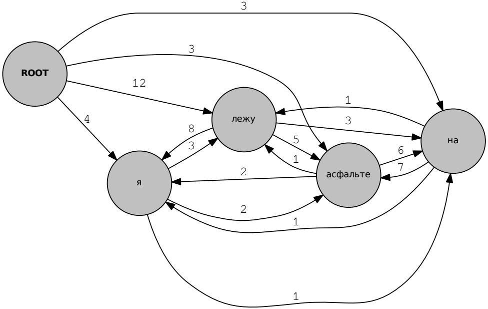
Why? Defining features is easy enough, but defining all those indicator features is tiresome.

Extensions

- **Labelled parsing**: Instead of having three transitions, the LEFT-ARC and RIGHT-ARC transitions are expanded for the number of labels,
    - e.g. LEFT-ARC$_{\text{nsubj}}$
- **Non-projective parsing**: Add an extra transition which “swaps” adjacent nodes

Graph-based

Basic model



- Represent sentence as directed graph
- Score every edge
- Running the spanning tree algorithm

Compared to transition based

Scoring

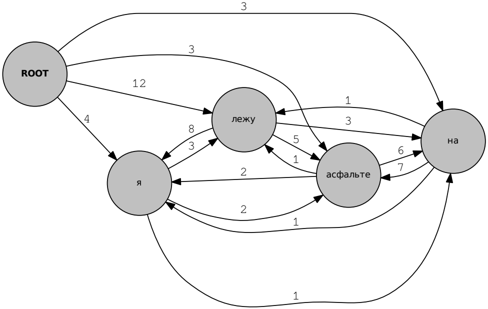
Where do the scores come from ?

- ...

Maximum spanning tree

- Higher score
- Contains all nodes
- Each node has at most one incoming edge
- Originates from a single, predefined root

Dense graph

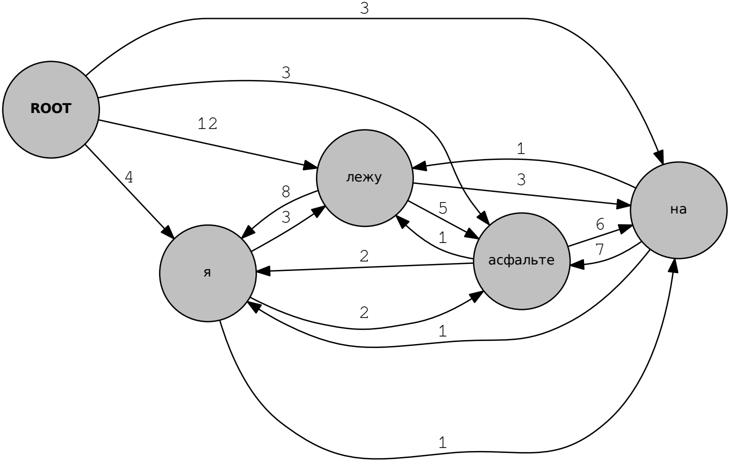


- Links between all nodes
- Except the root

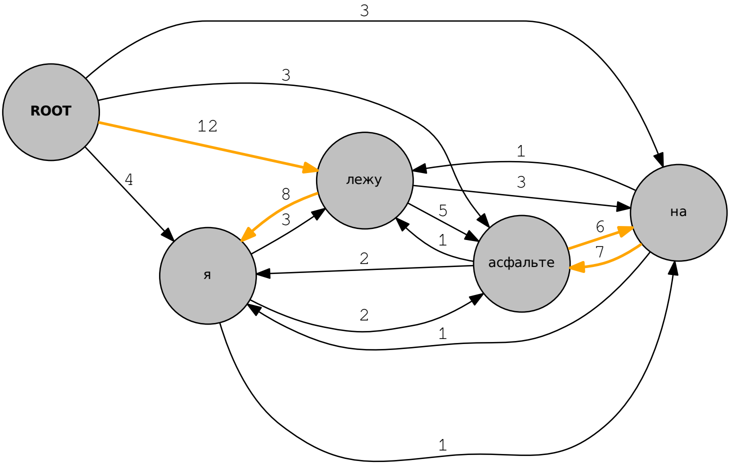
Chu-Liu-Edmonds

- For each node in the graph
  - pick the incoming arc with the highest weight.
  - if this makes a tree → it's the MST
- Otherwise:
  - For each cycle
    - contract the cycle
    - find the incoming arc with highest weight
    - remove incompatible arcs in the cycle
  - repeat

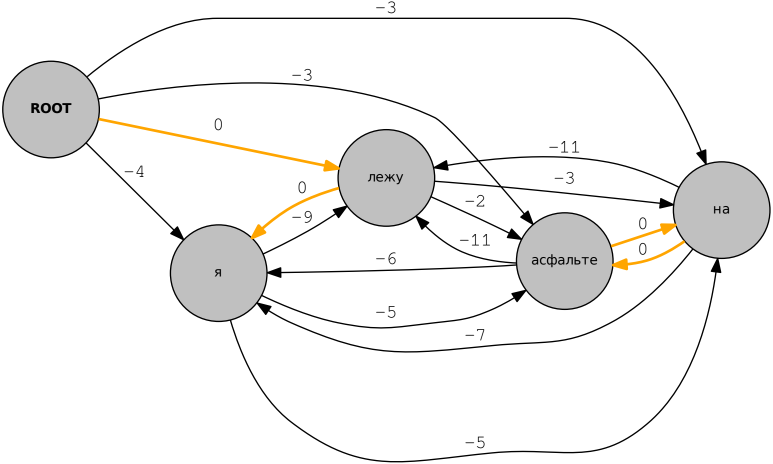
Example



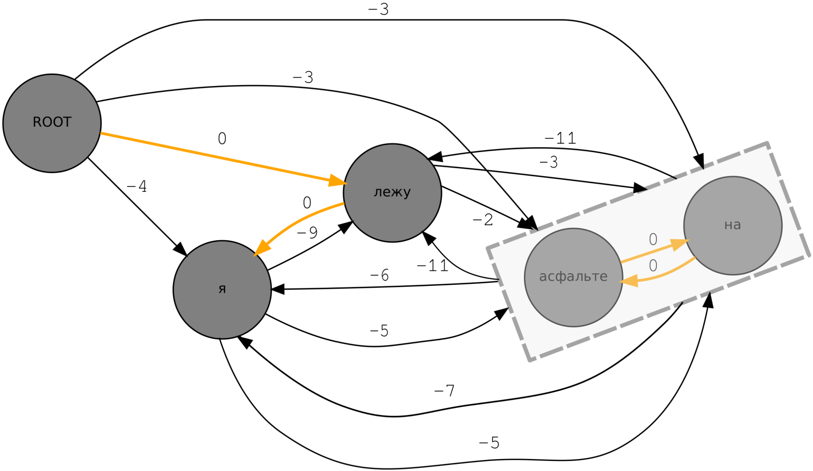
Example



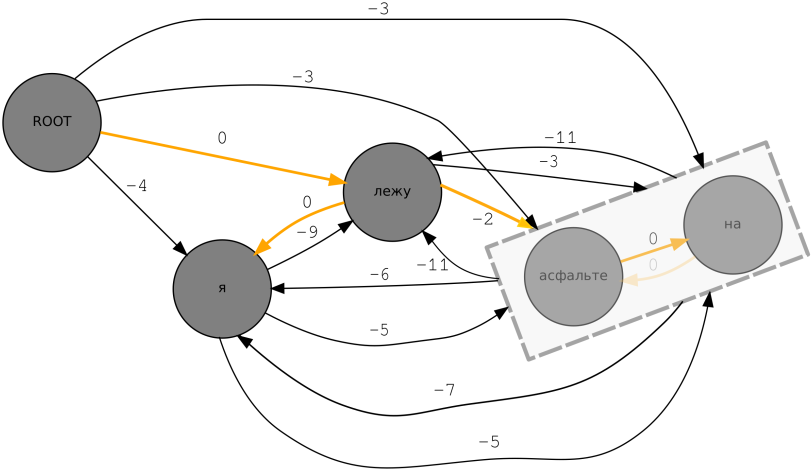
Example



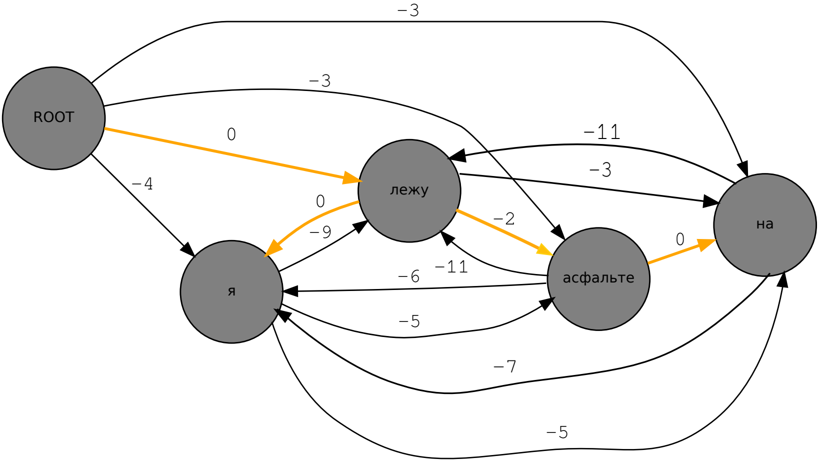
Example



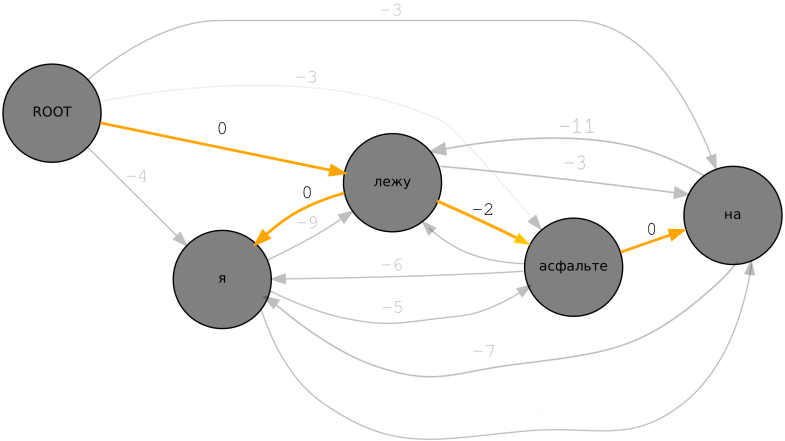
Example



Example



Example



Parsers

Modern:

- UDPipe http://github.com/ufal/udpipe
- SyntaxNet https://github.com/tensorflow/models/tree/master/research/syntaxnet
- BiST https://github.com/elikip/bist-parser
  - Both MST and transition variants
- Stanford Parser https://nlp.stanford.edu/software/nndep.html

Historical:

- MaltParser
- MSTParser

Evaluation

Simple evaluation:

- Unlabelled attachment score, **UAS**: correct heads/total heads
- Labelled attachment score, **LAS**: (correct heads+labels)/total heads

Shared tasks

CoNLL 2018

Practical

Ace the exam!